

**Universidad ORT Uruguay
Facultad de Ingeniería**

Plataforma de Conocimiento Interactivo con LLM para Nichos Especializados: Caso Bonsái

Entregado como requisito para la obtención del título de Ingeniero en Sistemas

María Agustina Lamanna – 223040

Nahuel Larenas – 260079

Melissa Molinari – 229896

Sofía Piñeyro – 243498

Florencia Varela – 254005

Tutor: Ignacio Valle

2025

Declaración de autoría

Nosotros, Agustina Lamanna, Nahuel Larenas, Melissa Molinari, Sofía Piñeyro y Florencia Varela, declaramos que el trabajo que se presenta en esta obra es de nuestra propia mano. Podemos asegurar que:

- La obra fue producida en su totalidad mientras realizábamos Proyecto Final de Ingeniería en Sistemas;
- Cuando hemos consultado el trabajo publicado por otros, lo hemos atribuido con claridad;
- Cuando hemos citado obras de otros, hemos indicado las fuentes. Con excepción de estas citas, la obra es enteramente nuestra;
- En la obra, hemos acusado recibo de las ayudas recibidas;
- Cuando la obra se basa en trabajo realizado conjuntamente con otros, hemos explicado claramente qué fue contribuido por otros, y qué fue contribuido por nosotros;
- Ninguna parte de este trabajo ha sido publicada previamente a su entrega, excepto donde se han realizado las aclaraciones correspondientes.

Agustina Lamanna Villar

Florencia

Agustina Lamanna

Florencia Varela

23-09-2025

23-09-2025

Melissa

Nahuel

Sofía Piñeyro

Melissa Molinari

Nahuel Larenas

Sofía Piñeyro

23-09-2025

23-09-2025

23-09-2025

Agradecimientos

Nos gustaría aprovechar esta oportunidad para expresar nuestra más sincera gratitud a todas las personas que han sido parte fundamental en el camino de este proyecto.

A nuestro tutor, Ing. Ignacio Valle, quien con su guía, paciencia y orientación nos brindó la confianza y las herramientas necesarias para dar forma a esta idea. Su apoyo constante fue clave para transformar un simple concepto en un producto concreto y viable.

A nuestro cliente, Darío Macci, por su paciencia, sus valiosos consejos y su constante disposición. Su visión y la idea original que compartió fueron la base sobre la cual se construyó este proyecto, y su acompañamiento en cada etapa resultó fundamental para poder llevarlo a buen puerto.

A los apasionados del arte del bonsái en Latinoamérica, que nos regalaron su conocimiento, su tiempo y su entusiasmo. Validar la idea y probar el producto con ustedes fue no solo un privilegio, sino también una experiencia enriquecedora que nos inspiró a seguir adelante con más convicción.

A nuestras familias, que con su cariño, comprensión y aliento nos sostuvieron en los momentos de mayor desafío. Su apoyo incondicional fue el motor que nos permitió mantenernos firmes a lo largo de todo este proceso.

Finalmente, a todos aquellos que de alguna manera contribuyeron a este proyecto: amigos que ofrecieron palabras de aliento, colegas que compartieron ideas y cada persona que, directa o indirectamente, dejó su huella en este camino.

Gracias a todos por su apoyo,

Agustina, Melissa, Nahuel, Sofia y Florencia.

Abstract

La plataforma de conocimiento interactivo para nichos especializados surge en respuesta a la necesidad de ofrecer información confiable, verificable y contextualizada en dominios donde los modelos de lenguaje generalistas presentan limitaciones. El caso del bonsái ejemplifica este desafío, al tratarse de una disciplina que requiere conocimientos técnicos específicos y en la cual la información disponible suele ser contradictoria o carecer de sustento.

La solución desarrollada consiste en una plataforma *web* que integra un modelo de lenguaje de gran escala (LLM) bajo una arquitectura de Recuperación de Información Aumentada (RAG). Esta arquitectura combina el procesamiento del lenguaje natural con una base vectorial semántica, permitiendo recuperar contenido relevante y generar respuestas fundamentadas en fuentes validadas del dominio.

Para garantizar la precisión y capacidad de generalización del sistema, se implementó un mecanismo de embeddings semánticos complementado con una organización taxonómica de la información. Esta estrategia permite al LLM extender su comprensión y ofrecer orientación útil incluso cuando carece de información exacta sobre un elemento específico. De este modo, el sistema puede responder con pertinencia y precisión basándose en las características comunes que comparten elementos relacionados jerárquicamente, como especie, género o familia, manteniendo coherencia conceptual frente a consultas inéditas.

El desarrollo siguió un enfoque ágil iterativo, con validaciones continuas junto a bonsaistas y la aplicación de buenas prácticas de ingeniería. La implementación técnica se realizó utilizando *React* y *Next.js* en el *frontend*, con dos *backends* desarrollados en *Node.js* y *Python*, y una integración directa con los servicios de *OpenAI* y la base vectorial *Pinecone* mediante *API REST*. La infraestructura fue definida mediante *Infrastructure as Code* y diseñada bajo una arquitectura *multi-tenant* modular, asegurando escalabilidad y mantenibilidad a futuro.

El resultado es una herramienta estable, precisa y extensible que demuestra la viabilidad técnica de aplicar LLMs especializados mediante arquitecturas RAG, con potencial de expansión hacia otros ámbitos de conocimiento experto.

Palabras clave

LLM; Bonsáis; Inteligencia Artificial; Retrieval Augmentation Generation; Pinecone; Embedding; Vector Database; Kanban; Docker; AWS; Terraform; OpenAI; Vercel; NextJS; Python; Opik; NodeJS; Express; Nicho; Integración Continua; PostgreSQL; Agente; Agentic RAG; Multi Tenancy; Resend

Glosario

Apicultura: Cría y manejo de abejas para producir miel y otros productos, actividad agropecuaria con impacto económico y ambiental. [1]

Bonsái: Arte de cultivar árboles en contenedores pequeños controlando su crecimiento y forma mediante poda y técnicas específicas. [2]

Clasificación taxonómica: Proceso de organizar organismos en categorías jerárquicas (reino, filo, clase, orden, familia, género y especie) según relaciones y características.

Clean Architecture: modelo de diseño de software propuesto por Robert C. Martin que organiza el sistema en capas concéntricas independientes, donde las dependencias siempre apuntan hacia el núcleo, separando la lógica de negocio de los detalles de infraestructura para garantizar mantenibilidad, escalabilidad y testabilidad. [3]

Data pills: Una data pill es una unidad mínima de información coherente, derivada de una fuente textual o estructurada, que se procesa y convierte en un vector mediante un modelo de *embeddings* para su almacenamiento en una base de datos vectorial.

Design thinking: Enfoque de innovación centrado en las personas que combina deseabilidad, viabilidad y factibilidad mediante iteración empática. [4]

Discovery (de producto): Fase continua para comprender problemas y oportunidades del usuario antes/durante el desarrollo, reduciendo riesgo de construir lo equivocado. [5]

Embedding: El embedding es una forma de representar objetos, como texto, imágenes o audio, mediante puntos en un espacio vectorial continuo, donde la posición de esos puntos tiene un significado semántico relevante para los algoritmos de aprendizaje automático. [6]

Figma FigJam: Herramienta colaborativa en línea desarrollada por Figma, diseñada para facilitar la ideación, planificación y trabajo en equipo de manera visual e interactiva. Permite a los usuarios crear pizarras digitales compartidas donde pueden incorporar notas adhesivas, diagramas, flujos, mapas mentales y otros elementos gráficos en tiempo real. [7]

Horticultura: Rama de la agricultura que se enfoca en el cultivo intensivo de plantas para el consumo humano, el embellecimiento del entorno o fines medicinales. Abarca la producción de frutas, hortalizas, hierbas, flores, plantas ornamentales y cultivos especializados. [8]

Infrastructure as Code (IAC): Práctica de definir y gestionar infraestructura mediante archivos de código declarativos/versionados, habilitando automatización y repetibilidad. [9]

LangChain: Marco para desarrollar aplicaciones con LLMs que orquesta *prompts*, herramientas, memoria y cadenas/agentes. [10]

LangGraph: LangGraph, creado por Langchain, es un framework de código abierto para agentes de inteligencia artificial, diseñado para construir, desplegar y gestionar flujos de trabajo complejos basados en IA generativa. [11]

Large Language Model (LLM): Modelo de aprendizaje profundo (usualmente transformer) entrenado a gran escala para procesar y generar lenguaje natural. [12]

Minimum Viable Product (MVP): Versión mínima de un producto que permite validar hipótesis con usuarios con el menor esfuerzo y aprendizaje máximo. [13]

Nomenclátor: Listado/registro estandarizado de nombres aceptados (p. ej., científicos o comunes) regido por códigos de nomenclatura. [14]

Pinecone: Servicio gestionado de base vectorial para indexar, buscar y servir *embeddings* a baja latencia en aplicaciones de IA. [15]

Product Owner (PO): Responsable de maximizar el valor del producto gestionando el Product Backlog y alineando al equipo con la visión. [16]

Procesos enológicos: Conjunto de operaciones científicas y tecnológicas del vino (fermentación, clarificación, crianza, etc.). [17]

Retrieval-Augmented Generation (RAG): Técnica que combina recuperación de documentos con generación por LLM para producir respuestas fundamentadas y actualizadas.

Software Quality Assurance (SQA): Conjunto de actividades planificadas y sistemáticas para asegurar que procesos y productos de software cumplen estándares de calidad. [19]

Spikes (ágiles): Investigaciones acotadas para explorar soluciones o reducir incertidumbre técnica/funcional antes de implementar. [20]

Taxonomía Linneana: Es el sistema normal de clasificación en el cual se asigna a cada organismo un reino, filo, clase, orden, familia, género y especie. Este sistema agrupa los organismos en grupos cada vez más pequeños. [21]

Terraform: Herramienta Infrastructure as Code de HashiCorp para definir, planificar y aplicar infraestructura declarativa en múltiples proveedores. [22]

Timeboxing (timeboxed): Técnica de gestión que fija un límite temporal inamovible para una actividad, priorizando alcance dentro del tiempo acordado. [23]

Tokens: Unidades mínimas (palabras/subpalabras) en que se segmenta el texto para ser procesado por modelos de lenguaje. [24]

User Persona: Es una representación ficticia de un usuario arquetípico, construida a partir de investigación de usuarios, que sintetiza sus necesidades, objetivos, comportamientos y otras características relevantes para guiar las decisiones de diseño centradas en el usuario. [25]

Wishlist (producto): Lista priorizada de ideas o necesidades que podrían incorporarse al producto en el futuro, fuera del alcance comprometido actual. [26]

Índice general

1. Introducción	21
1.1. Elección del proyecto	21
1.2. Contexto del equipo	21
1.3. Descripción del cliente	22
1.4. Descripción del proyecto	23
1.5. Objetivos	24
1.5.1. Objetivos académicos.....	24
1.5.2. Objetivos del proceso.....	25
1.5.3. Objetivos del producto.....	26
2. Del problema a la solución	27
2.1. Contexto del problema.....	27
2.2. Validación de la propuesta de valor.....	30
2.2.1. Resultados de la validación.....	31
2.3. Definición de usuario objetivo.....	32
2.4. Solución.....	33
2.5. Comunidad.....	35
2.6. Actores.....	36
2.6.1. Administrador.....	36
2.6.2. Usuario.....	38
3. Marco metodológico	39
3.1. Metodología.....	39
3.2. Roles internos y relación con el cliente.....	39
3.3. Ciclo de vida.....	40
3.3.1. Fase Inicial.....	41
3.3.2. Fase de desarrollo.....	42
3.3.3. Fase de documentación.....	44
3.4. Herramientas utilizadas.....	44
4. Ingeniería de requerimientos	46
4.1. Adaptación del proceso de ingeniería de requerimientos.....	46
4.1.1. Listado de requerimientos iniciales del cliente.....	48
4.1.2. Análisis y consolidación.....	49

4.1.3.	Negociación y alineación.....	51
4.1.4.	Documentación y prototipación.....	53
4.1.5.	Validación con usuarios finales.....	56
4.2.	Requerimientos.....	60
4.2.1.	Funcionales.....	60
4.2.2.	Requerimientos funcionales del sistema de IA.....	72
4.2.3.	No funcionales.....	75
5.	Investigación del sistema de IA.....	81
5.1.	Investigaciones iniciales.....	81
5.1.1.	Proveedores de modelo de IA.....	81
5.1.2.	Javascript vs Python.....	83
5.1.3.	RAG vs. Fine Tuning.....	84
5.1.4.	Proveedor de base de datos.....	86
5.2.	Investigaciones posteriores.....	88
5.2.1.	RAG vs Agentic RAG.....	88
5.2.2.	Generalización.....	91
6.	Arquitectura e implementación del proyecto.....	94
6.1.	Desafíos para lograr la solución.....	94
6.2.	Descripción de la arquitectura.....	96
6.3.	Atributos de calidad.....	102
6.4.	Desarrollo de la arquitectura.....	106
6.4.1.	Backend Lógico.....	107
6.4.2.	Backend del Sistema de IA.....	115
6.4.3.	Frontend.....	136
6.5.	Infraestructura como Código.....	138
7.	Gestión del proyecto.....	140
7.1.	Cronograma e hitos.....	140
7.2.	Metodología de trabajo.....	142
7.2.1.	Adaptación de SCRUM.....	143
7.2.2.	Adaptación de Kanban.....	145
7.3.	Métricas de gestión.....	147
7.3.1.	Métricas en la etapa Scrum.....	148
7.3.2.	Métricas en la etapa Kanban.....	152

7.3.3.	Distribución de throughput.....	157
7.3.4.	Análisis y aprendizajes	159
7.4.	Gestión de la comunicación.....	161
7.4.1.	Comunicación entre el equipo	161
7.4.2.	Comunicación con el cliente.....	162
7.4.3.	Comunicación con el tutor.....	162
8.	Gestión de riesgos	163
8.1.	Estrategia de gestión de riesgos	163
8.2.	Identificación	163
8.3.	Análisis cuantitativo	165
8.4.	Seguimiento	167
9.	Gestión de calidad.....	170
9.1.	Objetivos del producto.....	170
9.2.	Objetivos del proceso	170
9.3.	Plan de calidad	171
9.4.	Aseguramiento de calidad	174
9.4.1.	Validación de requerimientos con usuarios	174
9.4.2.	Estudio de tecnologías y pruebas de concepto	174
9.4.3.	Definición de criterios de aceptación	174
9.4.4.	Revisión de arquitectura	175
9.4.5.	Revisión sistemática de código.....	175
9.4.6.	Implementación de pruebas automatizadas e integración continua.....	176
9.4.7.	Pruebas específicas para sistemas de IA.....	178
9.4.8.	Monitoreo continuo y feedback de usuarios	178
9.4.9.	Gestión proactiva de riesgos de calidad	179
9.4.11.	Auditorías y revisiones de calidad	180
9.5.	Métricas de calidad	181
10.	Gestión de la configuración	188
10.1.	Desarrollo de software	188
10.1.1.	Control de versiones y cambios.....	188
10.1.2.	Nombramiento de ramas.....	188
10.1.3.	Procedimientos de release	189
10.1.4.	Gestión de dependencias	190

10.2.	Documentación y gestión del proceso	191
10.2.1.	Linear	191
10.2.2.	Google Drive	192
10.2.3.	Documentación generada	193
11.	Conclusiones del proyecto	195
11.1.	Conclusiones sobre Objetivos	195
11.1.1.	Objetivos Académicos	195
11.1.2.	Objetivos del Producto	196
11.1.3.	Objetivos del Proceso	197
11.2.	Estado actual	198
11.3.	Lecciones aprendidas	198
11.4.	Proyección a futuro	200
12.	Referencias bibliográficas	201
13.	Anexos	210
13.1.	Validación de la propuesta de valor	210
13.2.	Usuario objetivo	213
13.3.	Listado de requerimientos iniciales del cliente	213
13.3.1.	Glosario/Definiciones	213
13.3.2.	Actores	214
13.3.3.	Funcionalidades	214
13.3.4.	Ejemplos de preguntas a soportar	220
13.4.	Validación del prototipo con usuarios finales	222
	Resultados de las entrevistas de usabilidad del prototipo	222
13.5.	Listado de Requerimientos Funcionales	224
13.6.	Investigaciones del sistema de IA	234
13.6.1.	Proveedores de modelos de IA	234
13.6.2.	Javascript vs Python	240
13.6.3.	Fine Tuning vs RAG	247
13.6.4.	Retrieval Augmented Generation	249
13.6.5.	Bases de datos vectoriales	253
13.6.6.	Comparativa entre Knowledge Graphs y Bases de Datos Vectoriales ..	256
13.6.7.	LangChain	260
13.7.	Evaluaciones del sistema	261

13.7.1.	Resultados de pruebas	261
13.7.2.	Alcance y configuración.....	264
13.8.	Cronogramas del proyecto.....	265
13.8.1.	Cronograma versión 1	265
13.8.2.	Cronograma versión 2	266
13.8.3.	Cronograma versión 3	267
13.9.	Gestión de riesgos: notas extraídas de reuniones de análisis.....	268
13.10.	Gestión de calidad	301
13.10.1.	Plan de calidad.....	301
13.10.2.	Casos de prueba	304
13.10.3.	Script de Tareas para validacion con usuarios.....	307
13.10.4.	Resumen de feedback recibido en las entrevistas de validación	310
13.10.5.	Resumen de las notas extraídas de devolución en revisiones.....	313
13.10.6.	Análisis de heurísticas de Nielsen	317
13.10.7.	Encuesta de satisfacción del cliente.....	323

Índice de tablas

Tabla 1: Ejemplo comparativo: respuesta modelo generalista vs. respuesta esperada por un bonsaista	29
Tabla 2: Comparación entre OpenAI y Google Gemini.....	82
Tabla 3: Comparación RAG tradicional vs Agentic RAG	90
Tabla 4: Comparación detallada de las implementaciones RAG tradicional y Agentic RAG.....	132
Tabla 5: Riesgos identificados.....	165
Tabla 6: Valor establecido para el riesgo 1.5 durante la revisión de mayo de 2025. ...	167
Tabla 7: Registro de bugs y tech-debt.	182
Tabla 8: Medición de tiempo por tarea.....	183
Tabla 9: Resultados y objetivos de la encuesta en Sprig.	185
Tabla 10: Documentación generada durante el proyecto.	194
Tabla 11: Validación de la propuesta de valor	212
Tabla 12: Criterios de decisión de proveedores de modelo de IA.....	235
Tabla 13: ChatGPT (OpenAI) vs. Google Gemini (Google AI)	236
Tabla 14: Modelos y versiones	236
Tabla 15: Esquema de precios (alto nivel)	237
Tabla 16: Perspectiva desarrollador(a)	238
Tabla 17: Opciones alternativas de proveedores	238
Tabla 18: Gestión de datos personales y cumplimiento.	239
Tabla 19: Comparación de métodos de indexación.....	255

Tabla 20: Comparación entre Knowledge Graphs y Bases de Datos Vectoriales	258
Tabla 21: Tecnologías recomendadas.....	259
Tabla 22: Riesgos	270
Tabla 23: Riesgos	281
Tabla 24: Riesgos	295
Tabla 25: Riesgos	299
Tabla 26: Plan de calidad.....	303
Tabla 27: Casos de prueba	307

Índice de ilustraciones

Ilustración 1: Visualización del feed principal en una computadora portátil.	34
Ilustración 2: Interfaz destinado a la creación de preguntas.	35
Ilustración 3: Dashboard administrativo.	37
Ilustración 4: Gestión de preguntas y respuestas para administrador de la comunidad.	38
Ilustración 5: Ciclo de Vida del proyecto con enfoque iterativo-incremental.	41
Ilustración 6: Ciclo de ingeniería de requerimientos.	48
Ilustración 7: Diagrama de entendimiento de los requerimientos iniciales con preguntas para el cliente.	50
Ilustración 8: Diagrama de los requerimientos de un administrador y como se da la configuración y moderación de la comunidad con comentarios, ideas y preguntas.	51
Ilustración 9: Diagrama de estado de una pregunta generado en la etapa de Análisis y consolidación con preguntas realizadas al cliente y su respuesta.	52
Ilustración 10: Priorizaciones realizadas en la etapa de negociación y alineación inicial	53
Ilustración 11: Pantalla inicial del pretotipo	54
Ilustración 12: Pantalla inicial del prototipo.	55
Ilustración 13: Resultados de la validación del problema – Frecuencia de búsqueda de cuidado de bonsáis.	57
Ilustración 14: Resultados de la validación del problema – Dificultad del cuidado de bonsáis.	57
Ilustración 15: Resultados de la validación del problema – Problemas más frecuentes al buscar información sobre bonsáis.	58
Ilustración 16: Comentarios de un administrador.	64

Ilustración 17: Máquina de estados con preguntas para el cliente - Versión 1	65
Ilustración 18: Máquina de estados - Versión 2	66
Ilustración 19: Máquina de estados - Versión 3	67
Ilustración 20: Flujo de una respuesta	68
Ilustración 21: Ejemplo de visualización de respuestas aprobadas, pendientes y seleccionadas como mejor respuesta.	71
Ilustración 22: Ejemplo de visualización de respuesta rechazada.	72
Ilustración 23: Visualización de comentario de un administrador en una respuesta rechazada.	72
Ilustración 24: Visualización de aceptar/rechazar respuesta.	72
Ilustración 25: Visualización desde dispositivo iPhone 15.	76
Ilustración 26: Diagrama de flujo de proceso de Fine-Tuning. Tomado de [30]	84
Ilustración 27: Diagrama de flujo de proceso RAG. Tomado de [30]	85
Ilustración 28: Flujo jerárquico del RAG tradicional	89
Ilustración 29: Diagrama de Arquitectura de Alto Nivel. Elaborado a partir de [31]. ...	96
Ilustración 30: Logs del Niche-API filtrados por la palabra “RAG”.	108
Ilustración 31: Diagramas de capas del backend Niche-API	108
Ilustración 32: Diagrama de secuencia de la autenticación del primer nivel	112
Ilustración 33: Diagrama de secuencia de la autenticación del segundo nivel.....	113
Ilustración 34: Diagrama de secuencia de realizar una pregunta	114
Ilustración 35: Flujo general del proceso y estructura del estado compartido del backend.	118

Ilustración 36: Diagrama de componentes y conectores del backend del sistema de inteligencia artificial.	119
Ilustración 37: Objeto taxonómico de un árbol	122
Ilustración 38: Grafo de estado completo para la implementación RAG tradicional... ..	124
Ilustración 39: Arquitectura del sistema agéntico	128
Ilustración 40: Captura de Opik para una ejecución del grafo RAG.	134
Ilustración 41: Métricas de la última evaluación	135
Ilustración 42: Encuesta realizada a través de Sprig luego de obtener una respuesta de IA.	138
Ilustración 43: Cronograma final del proyecto simplificado	141
Ilustración 44: Action items resultados de la Retrospectiva.....	144
Ilustración 45: Organización del ciclo de desarrollo y asignación de tareas por integrante en Linear	146
Ilustración 46: Velocidad del equipo por sprint.	149
Ilustración 47: Compromiso y cumplimiento por sprint (Commitment Reliability)....	151
Ilustración 48: Throughput del equipo.	153
Ilustración 49: Cycle Time a lo largo del proyecto.	155
Ilustración 50: Distribución del esfuerzo entre el módulo de inteligencia artificial y el desarrollo del resto de la aplicación.	159
Ilustración 51: Matriz de riesgos	166
Ilustración 52: Evolución de la severidad de los riesgos a través del tiempo.	168
Ilustración 53: Ejemplo de tarea en el backlog con sus criterios de aceptación.....	175
Ilustración 54: Porcentaje de cobertura de los tests unitarios.....	177

Ilustración 55: Template para Bugs	179
Ilustración 56: Bug creado con el template.	180
Ilustración 57: Resultados de la encuesta en Sprig.....	184
Ilustración 58: Resultados de la encuesta en Sprig.....	184
Ilustración 59: Documentación generada para el release de la versión 1.0.4 de Niche-API	190
Ilustración 60: Análisis del ciclo 22 en Linear.	192
Ilustración 61: Usuario objetivo	213
Ilustración 62: Métricas de iteración 1	262
Ilustración 63: Métricas de iteración 2	262
Ilustración 64: Métricas de iteración 3	263
Ilustración 65: Métricas de iteración 4.	263
Ilustración 66: Visión general del último experimento realizado.....	264
Ilustración 67: Ejemplo representativo del formato de prueba empleado en Opik.	265
Ilustración 68: Cronograma del proyecto versión 1	265
Ilustración 69: Cronograma del proyecto versión 2	266
Ilustración 70: Cronograma del proyecto versión 3	267
Ilustración 71: Matriz de riesgos	271
Ilustración 72: Visualización de riesgos.....	271
Ilustración 73: Matriz de riesgos	282
Ilustración 74: Severidad de los riesgos a través del tiempo del proyecto.	296
Ilustración 75: Severidad de los riesgos a través del tiempo	299

Ilustración 76: Visualización de la comunidad.....	317
Ilustración 77: Visualización de mensajes de carga	317
Ilustración 78: Visualización de acciones completadas exitosamente	317
Ilustración 79: Visualización de iconos utilizados	317
Ilustración 80: Cambio de comunidad.....	318
Ilustración 81: Visualización de "Mis Preguntas"	318
Ilustración 82: Configuración de información personal	319
Ilustración 83: Visualización de "Hacer una pregunta".....	320
Ilustración 84: Iconos	320
Ilustración 85: Validación en formularios.	320
Ilustración 86: Visualización de error.	322

1. Introducción

1.1. Elección del proyecto

La elección del proyecto estuvo determinada por una propuesta realizada por un cliente externo, quien planteó la necesidad de contar con una herramienta tecnológica capaz de responder preguntas especializadas en dominios de conocimiento muy concretos. Esta demanda surgió de la falta de soluciones existentes en el mercado que pudieran ofrecer respuestas fiables y contextualizadas en áreas de nicho, como por ejemplo el arte del bonsái.

A partir de esta propuesta, se evaluó la pertinencia académica y técnica del proyecto, considerando su viabilidad, grado de innovación y el potencial impacto que podría generar en usuarios con intereses específicos. El análisis confirmó que se trataba de una oportunidad adecuada para aplicar los conocimientos adquiridos a lo largo de la carrera, al mismo tiempo que se respondía a una necesidad real planteada por un cliente.

De este modo, la elección del proyecto no solo obedeció a un interés personal del equipo, sino también a la posibilidad de trabajar en un encargo concreto, con expectativas definidas y orientado a la aplicación práctica de modelos de lenguaje de gran escala. Este enfoque permitió vincular el entorno académico con el profesional, garantizando que el resultado final tuviera un valor tangible para el cliente.

1.2. Contexto del equipo

El equipo se encuentra conformado por cinco estudiantes de la carrera de Ingeniería en Sistemas. Sus integrantes se conocieron durante el tercer y cuarto semestre de la carrera, trabajando en conjunto en diversas propuestas académicas dentro de las asignaturas de dichos semestres. Hacia el quinto semestre, con miras a comenzar a planificar este proyecto final de carrera, decidieron conformar un grupo de tesis, dado que durante los semestres previos habían logrado desarrollar una dinámica colaborativa sólida.

A nivel profesional, el grupo reúne experiencia en desarrollo de software, respaldada por la formación académica y práctica adquirida a lo largo de la carrera. En lo referente al desarrollo de Inteligencia Artificial en general, y al uso de Large Language Models

(LLMs) en particular, la experiencia inicial era más limitada, aunque se complementó mediante la formación especializada obtenida en cursos como Inteligencia Artificial y Machine Learning de la Universidad ORT.

Complementariamente, Agustina y Florencia cursaron la asignatura electiva Infraestructura en la Nube 1, lo que aportó al equipo una comprensión profunda sobre los principios de *Infrastructure as Code (IaC)*, despliegue automatizado y administración de entornos en la nube de Amazon. Este conocimiento resultó clave para diseñar la arquitectura técnica del proyecto, definir la infraestructura necesaria para el funcionamiento del sistema y asegurar la escalabilidad y mantenibilidad de la solución propuesta.

Asimismo, aproximadamente cuatro meses después del inicio de la tesis, Nahuel asumió un nuevo puesto en su trabajo orientado al desarrollo de agentes de IA, lo que aportó al equipo un conocimiento más profundo y aplicado en esta área, fortaleciendo significativamente el avance técnico del proyecto.

De esta manera, la combinación de perfiles académicos y profesionales del equipo, en áreas como el desarrollo de software, inteligencia artificial e infraestructura en la nube, generó una sinergia clave para afrontar los desafíos técnicos y metodológicos del proyecto.

1.3. Descripción del cliente

El cliente de este proyecto es el Ingeniero Darío Macchi, bonsaista con una reconocida trayectoria y participación activa dentro de la comunidad bonsaista en Uruguay, así como en espacios internacionales vinculados a esta disciplina. Su experiencia en el cultivo, cuidado y enseñanza del bonsái le ha permitido establecer una red de contactos significativa y consolidarse como referente en este ámbito especializado.

En el plano profesional, se desempeña como Ingeniero en Sistemas, formación que le ha brindado una sólida base en el área tecnológica. Además, cumple la función de tutor de proyectos finales de carrera en la Universidad ORT Uruguay, lo que le otorga experiencia en el acompañamiento de iniciativas académicas con aplicación práctica. Esta combinación de conocimientos especializados en bonsái y en sistemas constituye el punto

de partida para la propuesta de este proyecto, en la que convergen interés temático y enfoque técnico.

Durante el desarrollo del proyecto, el cliente desempeñó un rol central en la definición de objetivos y requerimientos, aportando su conocimiento experto y validando las necesidades de los usuarios finales. Asimismo, facilitó el acceso a otros bonsaistas expertos con los que se pudo contrastar y validar tanto la idea inicial como el producto desarrollado, asegurando así que la solución respondiera de manera adecuada a las expectativas del nicho de usuarios y cumpliera con criterios de pertinencia y utilidad práctica.

1.4. Descripción del proyecto

En la actualidad, los usuarios que buscan información especializada en nichos concretos, como el arte del bonsái, enfrentan dificultades para obtener respuestas rápidas, precisas y contextualizadas. Este proyecto surge para abordar esa necesidad mediante el desarrollo de una plataforma digital interactiva, capaz de brindar asesoramiento experto de manera accesible y eficiente.

El núcleo de la plataforma está basado en un modelo de lenguaje de gran escala (Large Language Model, LLM, por su sigla en inglés), que permite mantener interacciones naturales con los usuarios y responder a sus consultas de manera comprensible y relevante. A diferencia de foros o sistemas de búsqueda tradicionales, la plataforma integra conocimiento especializado del dominio, ofreciendo respuestas adaptadas al contexto y a las preferencias del usuario.

Uno de los elementos distintivos de la solución es su flexibilidad y escalabilidad. La plataforma se diseñó como un MVP, listo para producción, que permite integrar diferentes modelos de lenguaje según las necesidades futuras y expandirse a nuevos dominios de conocimiento. Además, facilita la conexión con comunidades digitales existentes, permitiendo validar y mejorar el producto con retroalimentación de usuarios y expertos.

El desarrollo del proyecto combina la aplicación de conocimientos avanzados en ingeniería de software, gestión de proyectos e inteligencia artificial, con un entendimiento profundo del dominio del bonsái y de las necesidades de los usuarios finales. Mediante un proceso iterativo de diseño, pruebas y validaciones, el equipo busca asegurar que la

plataforma no solo satisfaga los requerimientos del cliente, sino que también ofrezca funcionalidades relevantes y una experiencia de uso enriquecedora. De esta manera, se presenta como una solución útil, versátil y confiable, diseñada para aportar un valor distintivo dentro de su nicho especializado.

1.5. Objetivos

Para orientar el desarrollo de este proyecto, se definieron de manera clara y estructurada los objetivos que guiaron tanto el aprendizaje académico del equipo como el diseño y la implementación de la solución tecnológica. Esta definición inicial estableció un marco de trabajo común, garantizando que todas las etapas del proyecto se enfoquen en cumplir con las expectativas del cliente y los criterios académicos.

Los objetivos se estructuran en tres categorías principales: académicos, de proceso y de producto. Los objetivos académicos están orientados al fortalecimiento de competencias técnicas, metodológicas y de gestión del equipo, mientras que los objetivos de proceso buscan asegurar una planificación y ejecución eficiente de todas las fases del proyecto. Por su parte, los objetivos de producto se centran en garantizar que la plataforma satisfaga los requerimientos del cliente, sea funcional, medible y aporte valor real a los usuarios finales.

A continuación, se presentan y describen los objetivos específicos de cada categoría, detallando cómo contribuyen al logro de los resultados esperados y los criterios que permitirán evaluar su cumplimiento.

1.5.1. Objetivos académicos

OA1: Aplicación de conocimientos adquiridos en un proyecto de contexto real

Demostrar la capacidad de aplicar conceptos avanzados de ingeniería de software en un proyecto real y desafiante. Para el cumplimiento de este objetivo, se apuntó a que el equipo integre conocimientos adquiridos referentes a ingeniería de requerimientos, gestión ágil de proyectos, patrones de diseño de arquitectura, tácticas y patrones para cumplir con atributos de calidad, procesos de desarrollo, gestión de riesgos, gestión de la configuración y aseguramiento de la calidad.

OA2: Conocimiento del estado del arte en el desarrollo de aplicaciones con LLMs

Conocer en profundidad el estado del arte en el desarrollo de aplicaciones con LLMs, incluyendo frameworks, buenas prácticas, principios y patrones utilizados en este tipo de sistemas. Esto abarca el uso de Retrieval-Augmented Generation (RAG), bases de datos vectoriales y evaluación de la respuesta mediante *testing*. Para el cumplimiento de este objetivo se apuntó a que el sistema final diseñado contemplase el conocimiento adquirido.

OA3: Contribuir al conocimiento académico sobre la aplicación de LLMs

Aportar conocimiento académico sobre la aplicación de LLMs en soluciones a problemas reales. Para cumplir con este objetivo nos propusimos documentar de manera detallada todo lo aprendido durante el desarrollo de la plataforma, de modo que pueda servir como referencia para futuros desarrollos académicos y profesionales.

OA4: Alcanzar un desempeño académico sobresaliente

Una prioridad fundamental fue ejecutar un proyecto que lograra resultados destacados en términos académicos. Por ello, el equipo se fijó como meta obtener una calificación final igual o superior a 95 sobre 100.

1.5.2. Objetivos del proceso

OPC1: Aplicar un marco de gestión ágil y planificación eficiente

Establecer un plan de trabajo claro y estructurado, utilizando un marco de gestión ágil que brinde al equipo la flexibilidad para adaptarse a cambios en las especificaciones del proyecto. Esto se cumplirá definiendo hitos, entregables y haciendo un seguimiento sistemático de los avances.

OPC2: Coordinación y trabajo en equipo

Garantizar una comunicación fluida y una asignación clara de responsabilidades, de manera que los problemas se puedan resolver de forma oportuna y se mantenga un flujo de trabajo ordenado. Este objetivo apunta a asegurar que las tareas se cumplan según lo planificado y fomentar la eficiencia en la ejecución de todas las fases del proyecto.

OPC3: Reflexión y mejora continua en el flujo de trabajo

Analizar de manera reflexiva cada paso del desarrollo del proyecto, evaluando las decisiones tomadas, los resultados obtenidos y las posibles alternativas de mejora. Este objetivo busca fortalecer la toma de decisiones informadas, fomentar el aprendizaje continuo y asegurar que cada etapa contribuya de manera efectiva al cumplimiento de los objetivos generales del proyecto.

1.5.3. Objetivos del producto

OPD1: Desarrollo de una plataforma funcional con LLM

Crear una plataforma que utilizara un modelo de lenguaje para responder consultas especializadas de manera precisa y confiable, cubriendo las necesidades definidas por el cliente y asegurando su aplicabilidad en un contexto real.

OPD2: Medición de la efectividad y calidad de la solución

Garantizar que la IA responda de manera precisa a los cinco tipos de preguntas inicialmente definidos por el cliente. Este enfoque nos permitirá valorar de forma objetiva la efectividad, calidad y pertinencia de la plataforma, asegurando que cumpla con las expectativas tanto del cliente como de los usuarios finales.

OPD3: Desarrollo de un MVP escalable y versátil

Desarrollar un producto mínimo viable (MVP) listo para entrar en producción, que permita validar las funcionalidades esenciales de la plataforma. Se buscó que la plataforma pueda adaptarse a nuevos dominios de conocimiento sin requerir desarrollos adicionales complejos, garantizando su evolución y expansión futura.

OPD4: Validación con expertos y usuarios del nicho

El equipo se propuso validar la idea y el producto desarrollado mediante la interacción directa con bonsaistas expertos y usuarios especializados del nicho. Este objetivo buscó contrastar los resultados obtenidos con el conocimiento práctico de la comunidad, evaluar la precisión y relevancia de las respuestas generadas por la IA, y recopilar retroalimentación cualitativa y cuantitativa que permitiera ajustar la plataforma antes de su puesta en producción.

2. Del problema a la solución

En este capítulo se presenta el desarrollo de la propuesta, enmarcado en un proceso que parte de la identificación del problema y avanza hacia el diseño de una solución que lo aborda de manera sistemática.

En primer lugar, se analiza el contexto en el que surge la problemática y se valida su existencia a través de instancias de investigación con usuarios reales. Luego, se caracteriza al público objetivo y se define la propuesta de valor, lo que permite sentar las bases conceptuales de la solución planteada. Finalmente, se describen los actores que intervienen en el sistema, estableciendo con claridad los roles y responsabilidades que cada uno cumple dentro de la plataforma.

2.1. Contexto del problema

El bonsái es una práctica artística milenaria que combina conocimientos de horticultura, diseño y estética japonesa. Su propósito no es únicamente mantener árboles en miniatura, sino recrear a través de técnicas de poda, alambrado y cultivo, la esencia de la naturaleza de este tipo de árboles en un espacio reducido. Este arte exige paciencia y un conocimiento técnico muy específico sobre especies, clima, estaciones y cuidados particulares.

A medida que nuevos aficionados se incorporan a esta disciplina, se enfrentan a una dificultad recurrente: la falta de información confiable, estructurada y adaptada a su contexto particular. Los bonsaistas suelen recurrir a fuentes en línea, como foros, redes sociales o blogs, donde los consejos son genéricos, las respuestas contradictorias y los datos que varían según el país o la fuente. Esta situación genera un problema concreto, la toma de decisiones incorrectas en el cuidado del bonsái, especialmente entre quienes están dando sus primeros pasos en este arte. Lo que aparenta ser una buena recomendación (por ejemplo, la frecuencia de riego o el tipo de sustrato a utilizar para nutrir un ejemplar) muchas veces resulta perjudicial para la salud del árbol y compromete años de trabajo y cuidado. La falta de criterios técnicos unificados y la ausencia de verificación en las fuentes consultadas conducen a errores costosos, frustración y, en algunos casos, al abandono de esta disciplina.

Asimismo, la información disponible en los foros o redes sociales consultadas, no suele adaptarse a las particularidades locales en la que se encuentran los bonsaistas. Factores como el clima, la humedad, la especie del árbol y la estación en la que se encuentran, influyen directamente en las decisiones de cuidado. Sin embargo, la mayoría de los recursos en línea no contemplan estas variables, presentando contenidos generalizados que ignoran el contexto del usuario. Esto afecta no solo a quienes buscan aprender, sino también a quienes desean compartir su conocimiento de manera sistemática y confiable dentro de la comunidad.

En los últimos años, los modelos de lenguaje de gran escala, como ChatGPT, han demostrado ser herramientas versátiles capaces de responder sobre una amplia variedad de temas con distintos niveles de especificidad. Sin embargo, en dominios de conocimiento altamente especializados, como es el caso de los bonsái, tienden a mostrar limitaciones importantes: sus respuestas resultan genéricas, poco precisas y, en ocasiones, contradictorias.

En el cuadro a continuación (Tabla 1), se ilustra la diferencia entre una respuesta genérica, típica de un foro o modelo de lenguaje no especializado y la respuesta que brindaría un experto en bonsái. Mientras que la primera ofrece una orientación superficial y general, que podría ser válida para algunas especies pero erróneas para otras, la segunda integra variables contextuales (especie, clima, humedad, tamaño del recipiente y tipo de sustrato) y refleja el conocimiento experto necesario para una recomendación precisa.

Consulta del usuario: “¿Cada cuánto debo regar mi bonsái de pino?”

Respuesta típica de un modelo generalista (o foro en línea):

“Los bonsáis deben regarse dos veces por semana o cuando la tierra se vea seca. Evítá que el sustrato se seque completamente y asegúrate de que tenga buen drenaje.” [27]

Respuesta esperada de un experto bonsaista:

“La frecuencia de riego del pino bonsái depende de factores como el tipo de sustrato, el tamaño de la maceta, la temperatura y la humedad ambiental. En climas templados como el de Uruguay, durante el verano puede requerir riego diario, mientras que en invierno basta con hacerlo cada dos o tres días. Es fundamental observar el nivel de humedad del sustrato antes de regar: si los primeros 2 cm están húmedos, es preferible esperar. El exceso de agua puede provocar pudrición radicular.” [28]

Tabla 1: Ejemplo comparativo: respuesta modelo generalista vs. respuesta esperada por un bonsaista

Este contraste evidencia la principal dificultad que enfrentan los bonsaistas al buscar información en línea. Se encuentran en un punto intermedio entre la sobreabundancia de información general, que no distingue contextos particulares, y una falta de herramientas específicas que ofrezcan respuestas confiables, verificadas y adaptadas a sus necesidades reales.

El problema trasciende el ámbito del bonsái y plantea un interrogante de mayor alcance: ¿en cuántos otros nichos de conocimiento los usuarios están expuestos a información incompleta o errónea sin ser plenamente conscientes de ello? De esta reflexión surge la interrogante central que motiva este proyecto: ¿cómo podemos ofrecer un espacio que brinde respuestas precisas, fundamentadas y fáciles de entender en temas de nicho donde los modelos generales y las fuentes informales suelen fallar?.

A modo de ejemplificación, en la apicultura, una misma especie de abeja puede recibir diferentes nombres comunes en función del país, como *Yateí* en Argentina o *Angelita* en Colombia. Esto genera confusión al momento de realizar búsquedas o compartir información sobre su cuidado. En viticultura, un mismo término como *Malbec* describe variedades con características y procesos enológicos distintos según el contexto geográfico.

Estos ejemplos reflejan que el desafío no es exclusivo del mundo del bonsái, sino que forma parte de un problema más amplio, la falta de precisión contextual en la información disponible para dominios de conocimiento de nicho.

2.2. Validación de la propuesta de valor

El problema que dio origen a este proyecto fue inicialmente identificado por el cliente, quien a partir de su experiencia propia en el ámbito del bonsái, planteó la necesidad de una herramienta capaz de ofrecer respuestas confiables y contextualizadas en dominios de conocimiento especializados. Sin embargo, con el propósito de sustentar esta hipótesis inicial, el equipo consideró pertinente llevar a cabo un proceso de validación de la propuesta de valor, orientado a confirmar la relevancia del problema y a profundizar en la comprensión de las necesidades reales de los usuarios finales.

Para ello, se diseñó y difundió una encuesta exploratoria dirigida a potenciales usuarios finales, con el fin de establecer un primer contacto con ellos. El objetivo principal fue obtener información demográfica que permitiera segmentar a la audiencia y construir perfiles detallados. Este análisis resultó esencial para caracterizar tanto a los usuarios interesados específicamente en el cuidado de bonsáis, como a aquellos que, en un futuro, podrían emplear la herramienta en otros nichos de conocimiento.

El público objetivo de la encuesta estuvo conformado por potenciales usuarios finales interesados en el cuidado del bonsái. Se incluyeron aficionados principiantes, entusiastas con experiencia, expertos y profesionales vinculados al área, como ingenieros agrónomos, con especial atención en aquellos que ya buscan activamente información en internet o que se muestran receptivos a nuevas soluciones tecnológicas. Estos perfiles también serán de utilidad en etapas posteriores, cuando se validen prototipos de diseño de interfaz y experiencia de usuario (UI/UX).

La encuesta tuvo además un rol fundamental en la confirmación de la relevancia del problema identificado. A través de las respuestas se buscó determinar si las dificultades observadas en el acceso a información especializada sobre bonsáis eran efectivamente significativas para los usuarios, y comprender cuáles eran las principales limitaciones que encontraban en las fuentes disponibles. De esta forma, la validación no solo permitió

reforzar la pertinencia del problema planteado, sino también identificar patrones en la forma en que los usuarios perciben y enfrentan estas carencias.

Otro aspecto considerado fue la percepción de un asistente basado en inteligencia artificial como parte de la solución. La encuesta permitió medir el nivel de confianza y disposición de los usuarios hacia una herramienta tecnológica capaz de responder preguntas en lenguaje natural. Los resultados de esta validación temprana ofrecieron un fundamento sólido para justificar la incorporación de un modelo de IA como componente central de la propuesta.

2.2.1. Resultados de la validación

La encuesta, que obtuvo un total de 111 respuestas, permitió confirmar que la problemática identificada es real y afecta tanto a bonsaistas de Uruguay como del exterior. El detalle del cuestionario aplicado se presenta en el Anexo 13.1, mientras que entre los principales hallazgos se destacan:

- **Frecuencia de búsqueda de información:** el 79% de los encuestados manifestó buscar con frecuencia información sobre el cuidado de bonsáis, mientras que un 16% lo hace ocasionalmente.
- **Problemas al acceder a información especializada:**
 - 66% señaló la falta de profundidad en las explicaciones.
 - 62% identificó respuestas inconsistentes o contradictorias.
 - 45% destacó la presencia de información poco confiable o sin respaldo.
 - 36% expresó dificultades para encontrar datos específicos.

Además de la encuesta, se realizaron entrevistas con algunos participantes que voluntariamente proporcionaron sus datos de contacto. El propósito de las mismas fue complementar los resultados de la encuesta, permitiendo explorar en mayor profundidad las experiencias de los participantes. A través de estos intercambios cualitativos se buscó comprender con más detalle las dificultades prácticas que enfrentan en el cuidado de sus bonsáis, así como relevar expectativas y criterios que posteriormente sirvieron de insumo para orientar el diseño de la solución.

En conjunto, los resultados obtenidos respaldan la hipótesis inicial: los aficionados al bonsái enfrentan barreras significativas para acceder a información confiable y

especializada, lo que justifica el desarrollo de una solución centrada en sus necesidades reales.

2.3. Definición de usuario objetivo

Con base en la información recolectada a través de las encuestas y entrevistas, se elaboraron *user personas* (perfil de usuario objetivo) que representan los perfiles predominantes de los potenciales usuarios de la solución. Este enfoque se apoyó en elementos de la metodología de *design thinking*, particularmente en la identificación de características, motivaciones y necesidades, con el objetivo de orientar el diseño de un producto centrado en el usuario. El detalle de los *user personas* construidos se presenta en el Anexo 13.2.

Los resultados indican que una proporción significativa de las personas interesadas en el cuidado de bonsáis pertenece a un rango etario avanzado, compuesto en muchos casos por individuos jubilados o próximos a serlo. Este grupo presenta, en general, habilidades tecnológicas limitadas y únicamente cierto grado de familiaridad con herramientas digitales de uso común, como Facebook o Quora. En consecuencia, cualquier solución propuesta debe priorizar la simplicidad, la accesibilidad, una curva de aprendizaje reducida y una experiencia de uso que les resulte cercana y comprensible.

Asimismo, se observó que varios de estos usuarios han tenido experiencias negativas derivadas de información incorrecta o incompleta obtenida en internet, lo que en ocasiones resultó en un deterioro de sus bonsáis. Este aspecto refuerza la importancia de generar confianza en la herramienta, ofreciendo contenido verificado y contextualizado.

Del análisis de la información se desprenden tres necesidades centrales del usuario objetivo:

- **Acceso a información confiable:** los usuarios requieren datos respaldados por fuentes sólidas, que reduzcan la incertidumbre frente a la abundancia de consejos contradictorios disponibles en la web.
- **Respaldo teórico:** además de recomendaciones prácticas, valoran explicaciones fundamentadas que les permitan comprender el porqué de cada acción en el cuidado de sus árboles.

- **Adaptación al contexto personal:** la información debe ajustarse a variables propias de cada usuario, como el tipo de bonsái, el entorno en el que vive y su nivel de experiencia.

En suma, el usuario objetivo puede caracterizarse como una persona mayor, con limitaciones en el uso de tecnología, que busca una herramienta confiable, accesible y personalizada para mejorar sus conocimientos y prácticas en el cuidado de bonsáis.

2.4. Solución

Este proyecto busca abordar este desafío mediante la creación de un asistente especializado que proporcione respuestas más específicas y fundamentadas sobre temas de nichos específicos, comenzando con el cuidado de bonsáis. Este asistente permitirá a los usuarios realizar preguntas en lenguaje natural, y utilizando tanto la información provista dentro del perfil del usuario como el contexto específico de cada pregunta, proporcionará respuestas precisas y fundamentadas. Además, los usuarios podrán utilizar etiquetas para clasificar las preguntas según su área de interés y tendrán la opción de aceptar o rechazar las respuestas. Por otra parte, los usuarios podrán contribuir a las respuestas brindadas por el asistente, generando una dinámica colaborativa similar a otras plataformas de preguntas y respuestas.

A nivel lógico, la solución se estructura en torno al concepto de comunidad, entendido como una unidad de conocimiento independiente. Cada comunidad representa un dominio temático o ámbito de especialización (por ejemplo, bonsáis o apicultura) y dispone de un entorno aislado para la gestión de sus datos, documentos y miembros.

El concepto de comunidad surge como una abstracción funcional más que como un actor del sistema. No representa a un usuario individual, sino a un espacio autónomo de información, que puede estar gestionado por una persona, institución u organización interesada en preservar y organizar su propio conocimiento especializado. Este modelo de aislamiento, basado en una arquitectura *multi-tenant*, facilita la escalabilidad de la solución y la extensión a nuevos dominios de conocimiento.

Por último, se busca que la solución sea extensible para otros nichos, lo que permitirá expandir su utilidad a otros dominios de conocimiento para los que también se requieren respuestas especializadas. Algunos ejemplos son la apicultura, la astronomía amateur, la

fermentación casera, así como áreas profesionales como las políticas laborales o la normativa en materia de seguridad ocupacional. En todos estos ámbitos, los usuarios valoran contar con información confiable, clara y adaptada a sus contextos particulares.

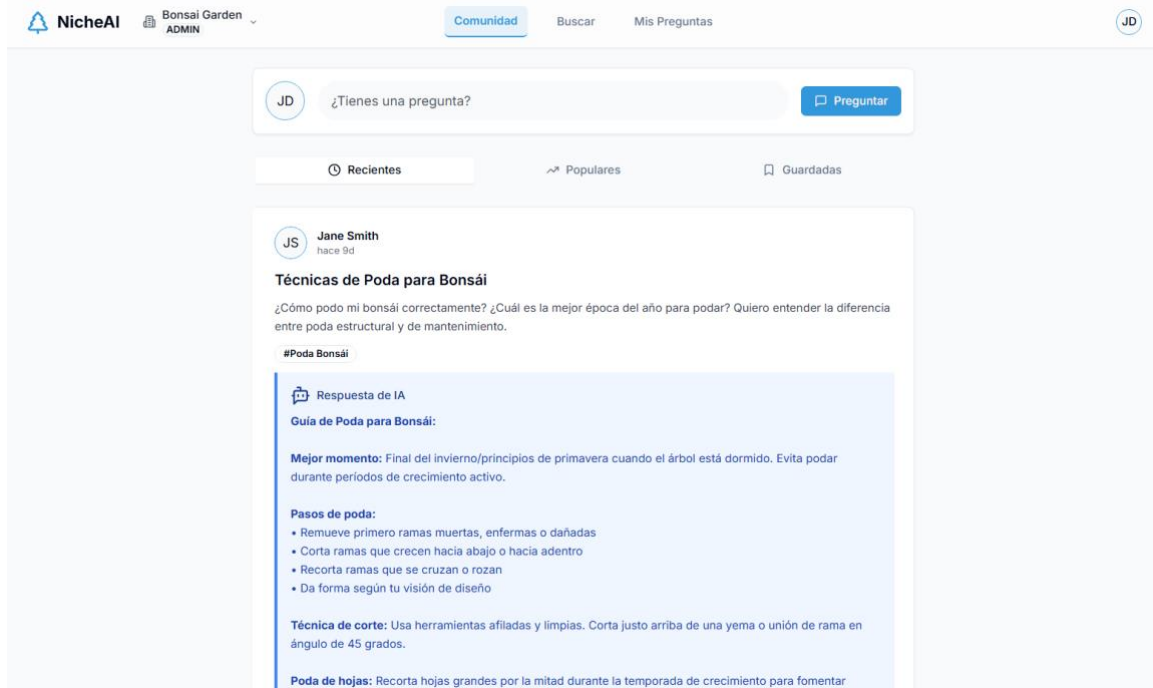


Ilustración 1: Visualización del feed principal en una computadora portátil.

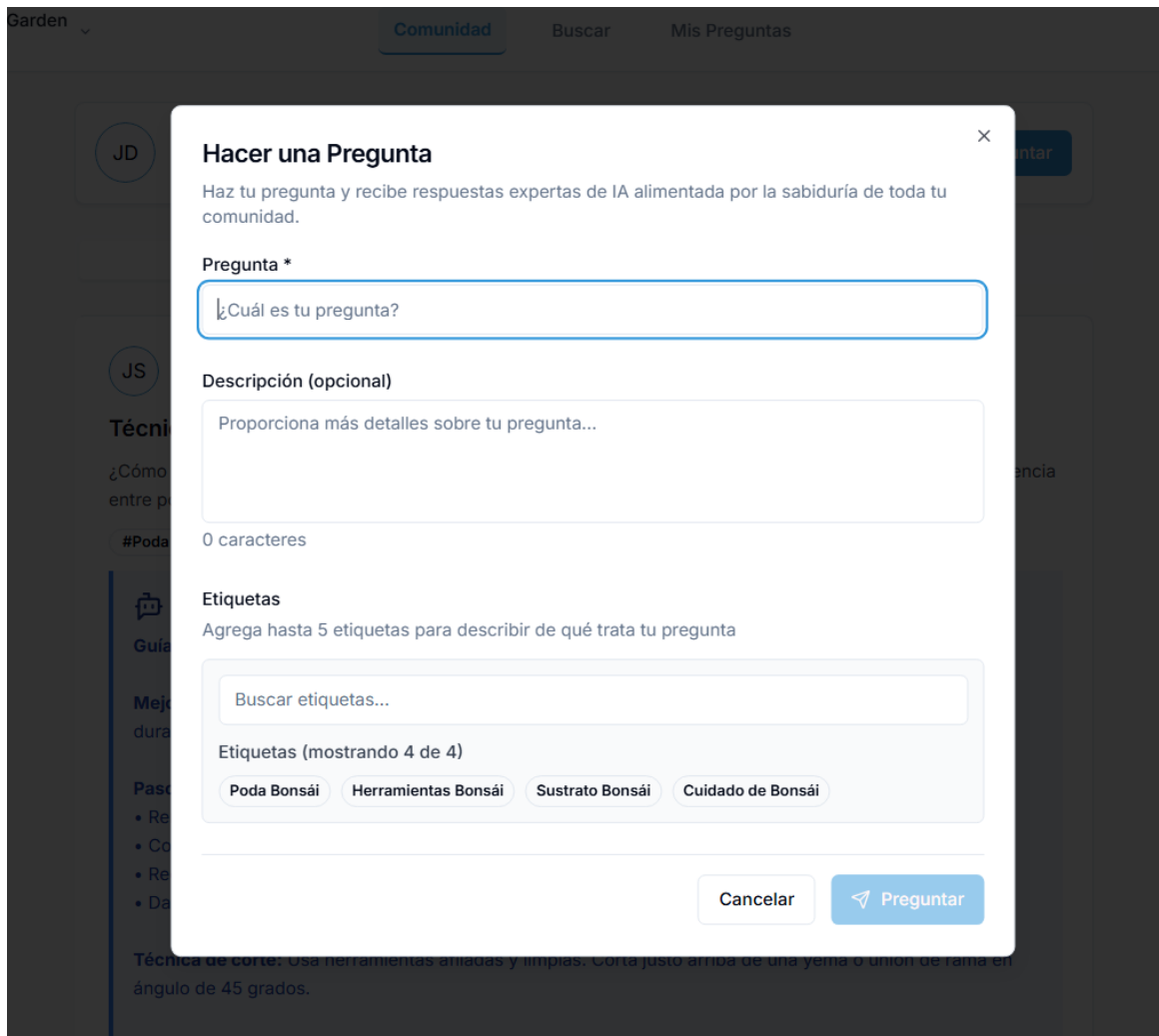


Ilustración 2: Interfaz destinado a la creación de preguntas.

2.5. Comunidad

La comunidad constituye la unidad central de la solución propuesta. Se entiende como un dominio de conocimiento independiente, o bien un silo de datos, que puede estar conformado por una institución, una organización, o una persona que desee gestionar y preservar su propio espacio de información.

Durante el desarrollo del proyecto, el concepto de comunidad atravesó un proceso de evolución. En un primer momento, se distinguían dos nociones: por un lado, el nicho, entendido como un tema o tópico específico (por ejemplo, el cuidado de bonsáis); y por otro, el *tenant*, que representaba un dominio de conocimiento particular dentro de ese nicho. Sin embargo, con el avance del diseño se identificó que esta diferenciación

resultaba innecesaria y, en algunos casos, podía introducir complejidad adicional a la experiencia de uso.

Como resultado, se optó por unificar ambos términos bajo el concepto de comunidad. Esta decisión respondió a dos motivos principales: en primer lugar, simplificar la comprensión del sistema a través de un concepto más claro y cercano para los usuarios; y en segundo lugar, mantener una denominación menos técnica que favorezca la usabilidad y la adopción de la plataforma.

Cada comunidad dispone de un entorno propio y aislado, lo que asegura la preservación de la privacidad, la identidad y la coherencia de los datos que gestiona. Esta arquitectura, de tipo *multi-tenant*, permite la coexistencia de múltiples comunidades sin comprometer la seguridad ni la independencia de cada una, y habilita la escalabilidad de la solución.

2.6. Actores

En el diseño del sistema resulta fundamental identificar a los distintos actores que interactúan con la plataforma, ya que cada uno de ellos cumple funciones específicas y tiene necesidades particulares. La definición clara de estos roles permite estructurar la solución de manera que responda de forma adecuada a los distintos perfiles de usuarios y garantice tanto la usabilidad como la seguridad de la información.

2.6.1. Administrador

El administrador es el usuario responsable de gestionar una comunidad. Su función principal consiste en asegurar el correcto funcionamiento del entorno, manteniendo la coherencia de los datos y facilitando la interacción de los usuarios dentro de ese espacio.

En la concepción inicial, el cliente definió el administrador como un perfil estrictamente técnico, encargado de la gestión de un *tenant*. Este rol estaba pensado como un “super usuario” con control total sobre el sistema, capaz de ejecutar tareas a través de interfaces avanzadas como la línea de comandos (CLI) y de tener la autoridad exclusiva para administrar todo lo que sucediera en su dominio.

No obstante, este enfoque presentaba dos limitaciones importantes. Por un lado, exigía un nivel de conocimientos técnicos que restringía el acceso y reducía la usabilidad del

sistema; por otro, concentraba demasiadas funciones en un único rol, lo que lo volvía difícil de escalar y poco flexible.

A partir de estas observaciones, el concepto de administrador fue redefinido con el objetivo de hacerlo más accesible y práctico. En esta nueva versión, el administrador conserva las capacidades de un usuario común, es decir, puede realizar todas las acciones disponibles para los demás miembros de la comunidad, pero cuenta además con herramientas de gestión que le permiten administrar la comunidad de forma integral como subir la información que consumirá el sistema de IA, invitar usuarios o moderar las respuestas. De esta manera, se elimina la necesidad de un perfil altamente técnico, se favorece la usabilidad y se asegura que el sistema pueda escalar sin depender un único usuario omnipotente.

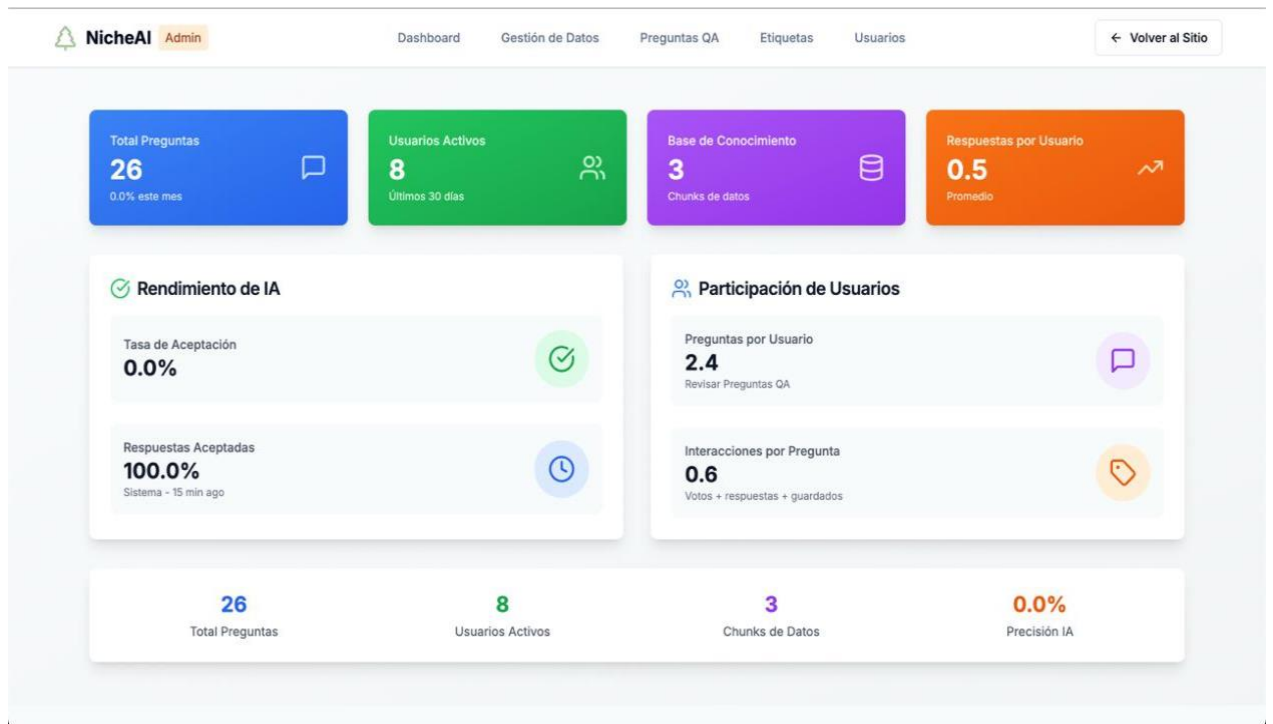


Ilustración 3: Dashboard administrativo.

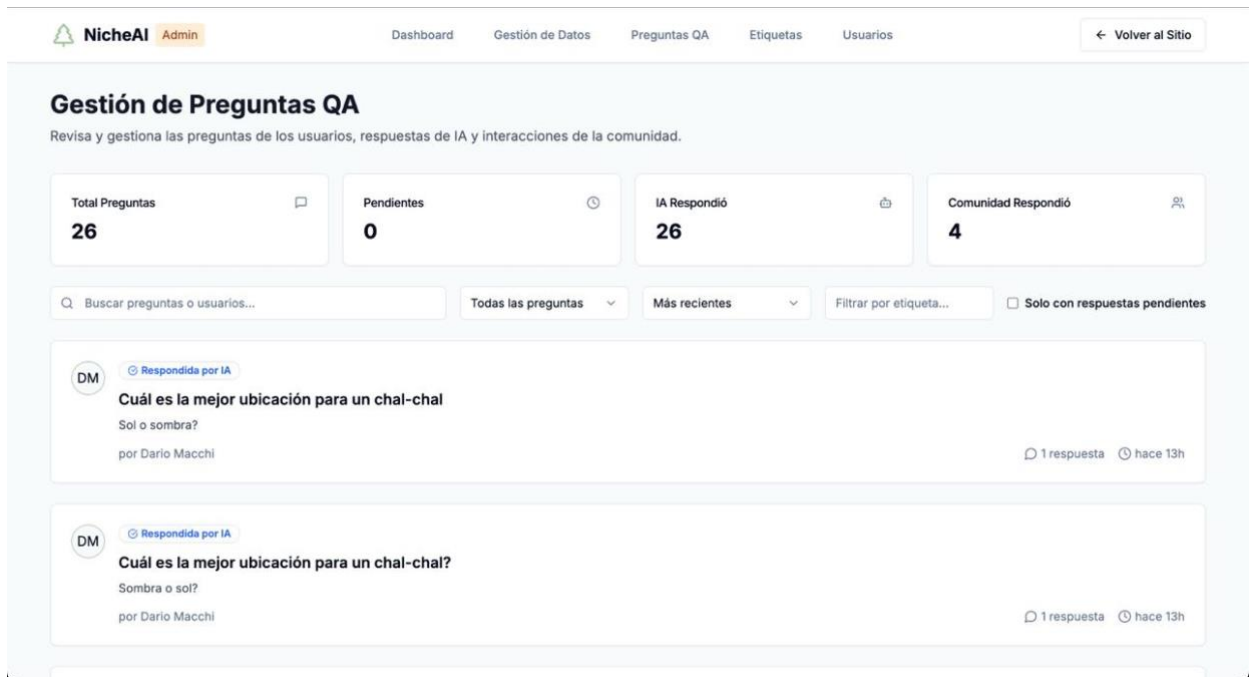


Ilustración 4: Gestión de preguntas y respuestas para administrador de la comunidad.

2.6.2. Usuario

El usuario representa al actor final de la solución, cuya motivación principal es acceder a información confiable y participar en la comunidad. En el inicio del proyecto, este rol se definía como “bonsaista junior”, en alusión al perfil de un aficionado que comienza en la práctica del bonsái y busca resolver dudas específicas.

Con el tiempo, este concepto evolucionó para generalizarse en “usuario”. Ya no se limita únicamente a formular preguntas, sino que también contribuye activamente en la comunidad, compartiendo conocimientos, experiencias y prácticas. De esta forma, el rol de usuario refleja un espectro más amplio de participación, donde la interacción va desde la consulta inicial hasta la colaboración en el crecimiento colectivo del sistema.

3. Marco metodológico

El marco metodológico describe de manera integrada los procesos y decisiones que guiaron el desarrollo del proyecto. Incluye una introducción a la metodología aplicada, el ciclo de vida adoptado, los roles del equipo y del cliente, y las distintas fases de trabajo, desde la investigación inicial hasta la documentación final.

3.1. Metodología

Se resolvió utilizar metodologías ágiles debido a que el equipo ya se encontraba familiarizado con este tipo de enfoque y se adapta correctamente a la naturaleza exploratoria y cambiante del proyecto. Las metodologías ágiles facilitaron la obtención de un MVP temprano el cual fue testeado por el cliente y usuarios reales del nicho del bonsái, permitiendo recibir feedback continuo y validar las decisiones técnicas tomadas.

En el capítulo 7, se profundiza en la implementación de estas metodologías en el contexto de este proyecto.

3.2. Roles internos y relación con el cliente

La organización interna del equipo y la dinámica de trabajo con el cliente se diseñaron en coherencia con el enfoque metodológico adoptado. Dado que el proyecto se desarrolló bajo un modelo ágil e incremental, la estructura de roles se orientó a maximizar la colaboración, la flexibilidad y la autonomía de los integrantes, al tiempo que se garantizaba una comunicación efectiva con el cliente.

Debido al tamaño del equipo, todos los integrantes participaron activamente en las tareas de desarrollo, organizándose en dos subequipos especializados: Nahuel y Sofia se encargaron del desarrollo del sistema de inteligencia artificial, mientras que Florencia, Melissa y Agustina se enfocaron en el desarrollo de la aplicación web y el producto en su conjunto.

Además, cada integrante asumió un rol específico dentro de la gestión del proyecto, con el objetivo de asegurar el seguimiento y cumplimiento de los aspectos críticos. Florencia fue responsable de las áreas de *DevOps* y arquitectura de software; Agustina se ocupó de la gestión del proceso y los riesgos; Melissa de la gestión de la calidad; y Sofia junto con

Nahuel de la gestión del desarrollo del sistema de IA. De esta forma se logró mejorar la organización y permitió mantener un control integral sobre los componentes clave del desarrollo.

La relación con el cliente, Darío Macchi, se estructuró para garantizar una comunicación continua y efectiva. Se llevaron a cabo reuniones internas semanales para revisar avances y planificar tareas, complementadas con reuniones quincenales con el cliente, donde se evaluaron progresos, ajustaron objetivos y reordenaron prioridades según la evolución del proyecto. Esta dinámica de interacción constante, junto con la gestión mediante las metodologías ágiles, aseguró transparencia, trazabilidad y alineación permanente entre las decisiones técnicas del equipo y las expectativas del cliente.

3.3. Ciclo de vida

El ciclo de vida adoptado en este proyecto se enmarca dentro del enfoque iterativo-incremental, ampliamente reconocido en la ingeniería de software por su capacidad de combinar desarrollo progresivo con mejora continua. Este tipo de enfoque permite construir el producto a través de sucesivos incrementos funcionales, donde cada iteración incorpora retroalimentación y nuevos aprendizajes obtenidos durante la ejecución del proyecto.

De acuerdo con el PMBOK Guide [29], los enfoques iterativos e incrementales resultan especialmente adecuados en contextos donde los requerimientos evolucionan o se refinan a lo largo del proyecto. Cada iteración implica la planificación, diseño, construcción y validación de un conjunto de funcionalidades, generando versiones parciales del producto que pueden ser revisadas y ajustadas según la respuesta del cliente o hallazgos técnicos.

El ciclo de vida se estructuró en tres fases interrelacionadas, Inicial, Desarrollo y Documentación. Estas fases constituyen el nivel macro del proceso y se vinculan directamente con las actividades operativas descritas en el Plan de calidad (capítulo 9.2). Cada fase integró iteraciones de análisis, diseño, implementación y validación, reflejando la naturaleza iterativa del proyecto y su orientación hacia la entrega progresiva de valor.

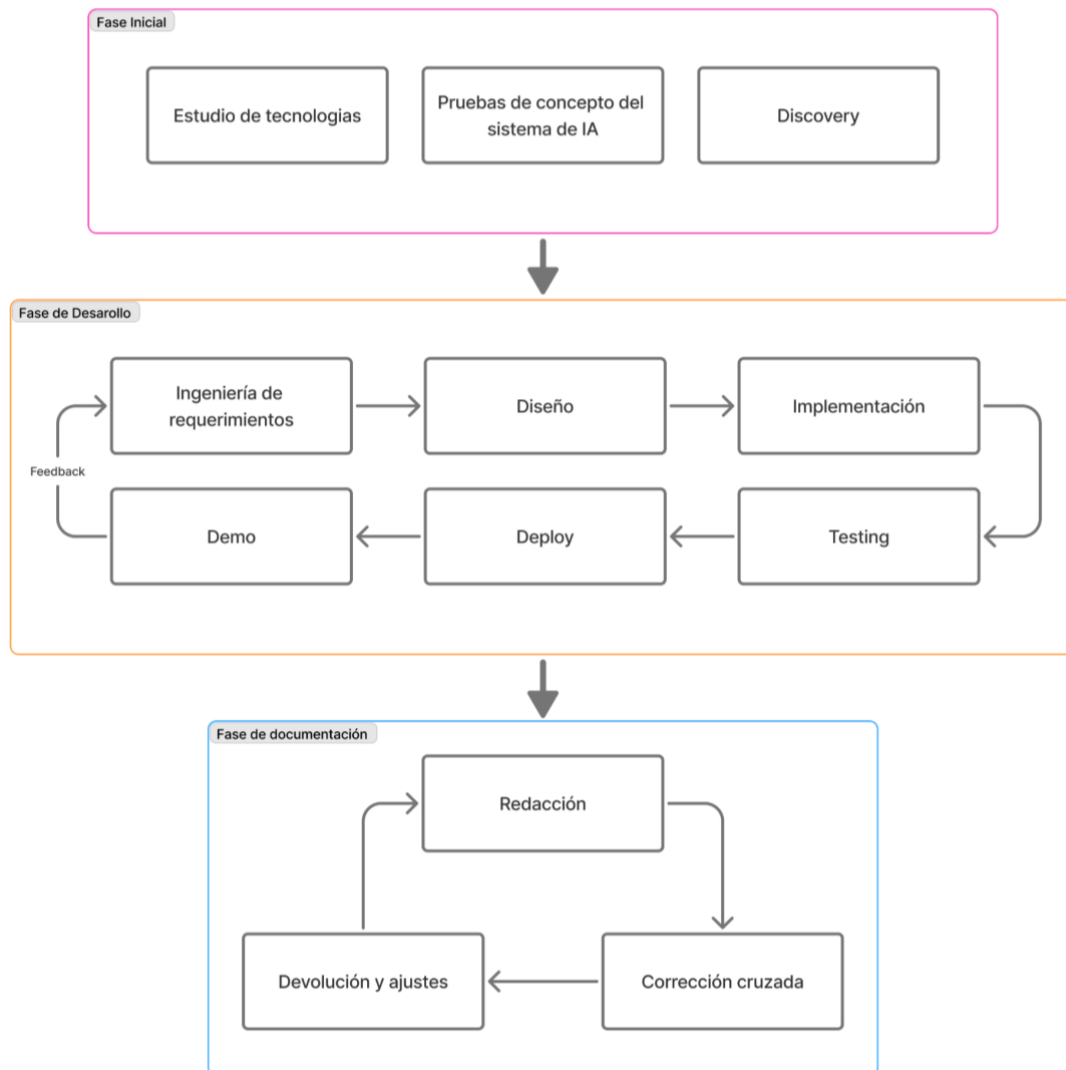


Ilustración 5: Ciclo de Vida del proyecto con enfoque iterativo-incremental.

3.3.1. Fase Inicial

En la fase inicial se establecieron las bases conceptuales, técnicas y funcionales del proyecto. En esta etapa se abordaron simultáneamente el estudio de tecnologías emergentes, las pruebas de concepto del módulo RAG, orientadas a validar la factibilidad y el desempeño del enfoque propuesto, y las actividades de discovery para comprender profundamente el dominio del problema y las necesidades de los usuarios finales.

Estudio de tecnologías

Durante esta fase se evaluaron y seleccionaron las tecnologías fundamentales para el desarrollo del sistema, incluyendo la arquitectura general, lenguajes de programación,

frameworks y servicios de infraestructura. Las decisiones técnicas y su justificación se detallan en el capítulo 6 (*Arquitectura e implementación del proyecto*).

Estudio de tecnologías de LLM

Se llevó a cabo una investigación profunda sobre proveedores de modelos de IA, estrategias de incorporación de conocimiento especializado, bases de datos vectoriales y mecanismos de generalización taxonómica. El proceso completo de investigación, pruebas de concepto y decisiones fundamentadas se presenta en el capítulo 5 (*Investigación del sistema de IA*).

Discovery

En este proceso de *discovery*, el equipo tuvo como objetivo comprender en profundidad el problema y validar los supuestos iniciales del proyecto. Para ello, se realizaron encuestas y entrevistas con potenciales usuarios que permitieron caracterizar sus necesidades, expectativas y dificultades reales. A partir de esta información, se definieron de forma estructurada los requerimientos funcionales y no funcionales del sistema.

El análisis detallado del problema se presenta en el capítulo 2 (*Del problema a la solución*), mientras que el proceso completo de ingeniería de requerimientos, junto con las técnicas aplicadas y los resultados obtenidos, se desarrolla en el capítulo 4 (*Ingeniería de requerimientos*).

3.3.2. Fase de desarrollo

La fase de desarrollo siguió un esquema iterativo e incremental, en el que se implementaron, evaluaron y ajustaron funcionalidades de manera continua a través de ciclos que incluyeron ingeniería de requerimientos, diseño, implementación, *testing*, despliegue y demostración con el cliente. Cada ciclo representó una versión funcional del sistema, lo que permitió obtener retroalimentación temprana, refinar los componentes existentes y redefinir prioridades según los resultados obtenidos.

Ingeniería de requerimientos

Incluyó la recepción y extracción de necesidades del cliente, el análisis y la estructuración de requerimientos, y la validación mediante prototipos y pruebas con usuarios objetivo. Los detalles completos se desarrollan en el capítulo 4 (*Ingeniería de requerimientos*)

Diseño

Durante esta etapa se definió la arquitectura del sistema, se tomaron decisiones técnicas fundamentales y se establecieron los patrones y tácticas para garantizar el cumplimiento de los atributos de calidad. El diseño de la arquitectura, las decisiones tomadas y su justificación se detallan en el capítulo 6 (*Arquitectura e implementación del proyecto*).

Implementación

La implementación se realizó de manera incremental, desarrollando funcionalidades en iteraciones que permitían validación continua. Se establecieron estándares de código, procesos de revisión entre pares y pruebas automatizadas para asegurar la calidad del producto desarrollado.

Testing

Cada incremento fue sometido a pruebas unitarias, de integración, de usabilidad y específicas para el sistema de IA. Los resultados de estas pruebas permitieron identificar y corregir defectos tempranamente.

Despliegue

Los incrementos funcionales se publicaron mediante procesos automatizados de release que aseguraron la consistencia y trazabilidad entre versiones. Estas automatizaciones facilitaron la implementación controlada de nuevas funcionalidades en los entornos de producción, dentro de comunidades creadas con fines demostrativos, reduciendo errores humanos y manteniendo la reproducibilidad del sistema.

Demostración con el cliente

Al finalizar cada ciclo, se realizaron demostraciones con el cliente donde se presentaron las funcionalidades implementadas, se recopiló feedback y se ajustaron las prioridades para el siguiente ciclo.

Gestión del desarrollo

El trabajo se organizó inicialmente bajo una adaptación del marco *Scrum*, con sprints de dos semanas que permitieron una entrega incremental de valor y retroalimentación temprana. A medida que el proyecto avanzó hacia etapas más experimentales, particularmente en el desarrollo del módulo de inteligencia artificial, se migró hacia *Kanban*. Esto facilitó un flujo de trabajo más flexible y continuo, adecuado para la naturaleza exploratoria de las tareas.

Las decisiones metodológicas y las adaptaciones específicas realizadas durante la gestión del proyecto se desarrollan con mayor detalle en el capítulo 7, *Gestión de proyecto*, donde se explican los fundamentos de la transición entre marcos de trabajo y su impacto en la dinámica del equipo.

3.3.3. Fase de documentación

Finalmente, la fase de documentación también se llevó a cabo de manera iterativa. Las tareas de redacción, corrección cruzada y revisión se realizaron en paralelo, integrando sugerencias del equipo para asegurar la coherencia y la calidad de los entregables. A diferencia de las fases previas, donde se aplicaron marcos metodológicos ágiles formales, esta etapa adoptó un enfoque colaborativo con revisiones cruzadas, distribuyendo la redacción según las áreas de especialización de los integrantes.

3.4. Herramientas utilizadas

La selección de herramientas respondió a la necesidad de sostener un flujo de trabajo ágil, colaborativo y transparente, alineado con los principios del enfoque iterativo-incremental adoptado en el proyecto.

Linear se empleó para la gestión de tareas, permitiendo visualizar el progreso, priorizar actividades de manera dinámica y facilitar la trazabilidad de los entregables. *GitHub* se

utilizó como repositorio central para el control de versiones y la colaboración en el código fuente, integrando *Github Actions* para la automatización de procesos de integración continua y validaciones previas a los *releases*.

Por su parte, *Slack* fue la principal herramienta de comunicación interna, favoreciendo la coordinación diaria del equipo y la resolución ágil de incidencias, mientras que *Postman* se utilizó para la verificación y prueba de las API desarrolladas, asegurando la coherencia de las respuestas y la estabilidad de los servicios.

En conjunto, estas herramientas establecieron un entorno de trabajo estructurado, promoviendo la eficiencia operativa, la transparencia del proceso y la calidad de los resultados obtenidos.

4. Ingeniería de requerimientos

El proceso de ingeniería de requerimientos se desarrolló bajo un enfoque ágil e incremental, cuidadosamente adaptado a las particularidades del cliente, los usuarios finales y la dinámica del entorno. A diferencia de un proceso lineal, se concibió como un ciclo iterativo de descubrimiento, validación y ajuste, en el que cada avance generó nuevo conocimiento sobre el dominio, permitió validar hipótesis y orientó las decisiones posteriores del equipo.

Este enfoque permitió mantener un equilibrio constante entre la visión estratégica del cliente y la realidad operativa de los usuarios, considerando que los requerimientos son dinámicos y están sujetos a revisión continua según los hallazgos obtenidos durante el análisis y la validación.

Si bien el cliente contaba con una lista preliminar de requerimientos, resultado de su experiencia previa y de una comprensión sólida de los objetivos funcionales, el equipo optó por no limitar el proceso a la recopilación inicial, sino abordarlo como un proceso de descubrimiento continuo. Bajo esta perspectiva, la ingeniería de requerimientos se convirtió en un mecanismo de aprendizaje y validación permanente, en el cual la identificación, análisis y priorización de requerimientos evolucionaron de forma colaborativa junto al cliente y los usuarios.

En consecuencia, el proceso trascendió a la fase inicial de relevamiento y se consolidó como un ciclo continuo de análisis, validación y ajuste, plenamente alineado con los principios de la ingeniería de requerimientos ágil y orientado a la construcción de un producto coherente con las necesidades reales del entorno de uso.

4.1. Adaptación del proceso de ingeniería de requerimientos

El proceso de ingeniería de requerimientos de este proyecto se ejecutó de forma evolutiva a través de iteraciones sucesivas que combinaron análisis, negociación, validación y reformulación de los requerimientos definidos inicialmente por el cliente. El punto de partida, como se mencionaba anteriormente, fue un listado de requerimientos provisto por el cliente, que reflejaba una primera visión del alcance funcional del sistema y sus objetivos estratégicos. A partir del mismo, el equipo realizó un análisis detallado con el

propósito de interiorizarse en el dominio del problema, hasta entonces desconocido, y así estructurar un backlog inicial que sirviera como base de trabajo.

Durante esta fase, el equipo detectó que varios requerimientos planteados por el cliente necesitaban una validación más profunda desde la perspectiva del usuario final. En consecuencia, se llevó adelante un proceso de *discovery*, orientado a comprender cómo se manifestaban las necesidades reales de los usuarios finales en la práctica, y a verificar si los requerimientos definidos efectivamente las resolvían.

Para ello se realizaron consultas exploratorias e interacciones con potenciales usuarios finales lo que permitió identificar discrepancias entre la visión inicial del cliente y las expectativas o dificultades concretas de quienes utilizarían la herramienta. Este intercambio resultó clave para reformular o descartar requerimientos, garantizando que la solución propuesta respondiera a un uso realista y contextualizado.

En paralelo, se identificaron ciertos requerimientos asociados a funcionalidades administrativas, donde el cliente asumía el rol de usuario principal al operar inicialmente como administrador del sistema. En estos casos, el proceso adoptó un carácter iterativo e incremental, con ciclos de definición por parte del equipo, prueba y ajuste continuo junto con el cliente. De esta forma, se validaron los comportamientos esperados y se optimizaron las interacciones administrativas antes de avanzar hacia el desarrollo.

Este enfoque mixto, que combinó actividades de *discovery* con usuarios finales y refinamiento iterativo con el cliente, dió lugar a un proceso altamente colaborativo y adaptativo, en el cual cada hallazgo o validación retroalimentó al conjunto de requerimientos. En consecuencia, el backlog inicial evolucionó en múltiples iteraciones, priorizando aquellos requerimientos de mayor impacto y reformulando aquellos que no se alineaban con las necesidades reales de uso final.

La Ilustración 6 resume este ciclo de ingeniería de requerimientos, que integró las etapas de análisis y consolidación, negociación y alineación, documentación y prototipación, y validación con usuarios finales. Este diagrama ilustra la naturaleza iterativa y cíclica del proceso, en el que cada fase aportó información que permitió revisar y mejorar las demás, favoreciendo de esta forma la construcción de una base sólida y validada para el posterior desarrollo del sistema. Por otra parte, permite ilustrar aquellos casos en dónde no fue

necesario realizar una validación con usuarios ya que se consideraban requerimientos base del sistema.

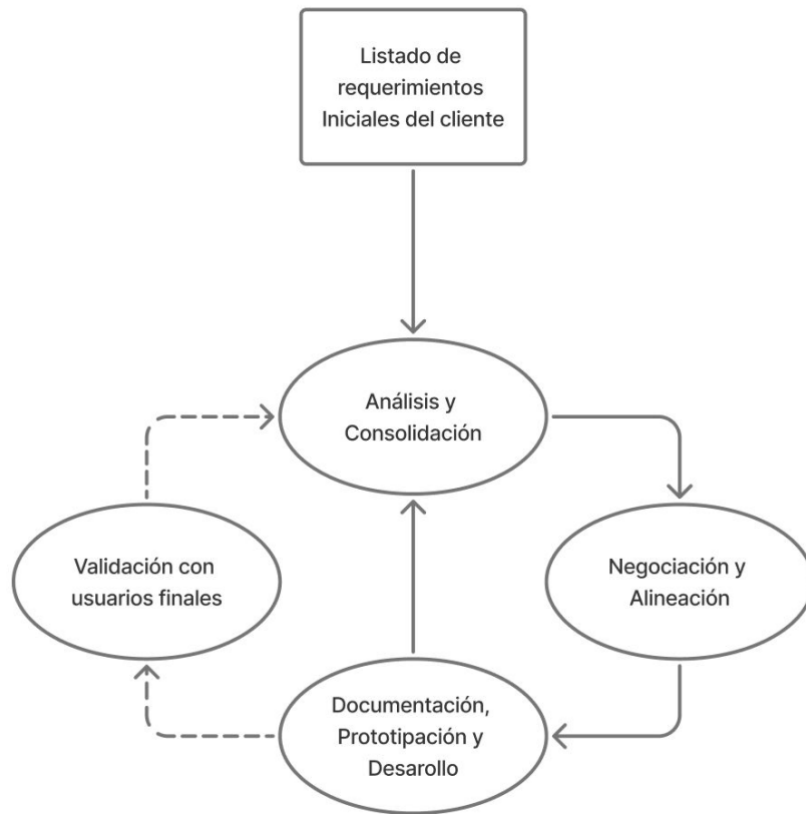


Ilustración 6: Ciclo de ingeniería de requerimientos.

A continuación se detallan los procesos realizados en cada una de estas etapas.

4.1.1. Listado de requerimientos iniciales del cliente

A partir del listado inicial de requerimientos provisto por el cliente, el equipo llevó adelante una primera fase de análisis orientada a comprender en profundidad el alcance funcional propuesto y a detectar posibles ambigüedades o superposiciones entre los puntos planteados. Dado que el dominio del problema resultaba novedoso para el equipo, esta instancia fue clave para construir una comprensión compartida sobre las expectativas del cliente y los objetivos estratégicos del cliente.

Durante este proceso, se clasificaron los requerimientos según su naturaleza, funcionales, no funcionales y aquellos deseados (*wishlist*). Esto permitió estructurar un backlog inicial

que serviría de base para las siguientes iteraciones. Esta clasificación permitió anticipar dependencias, áreas críticas y definir prioridades preliminares que luego serían validadas con el cliente y los usuarios finales.

El listado se consideró preliminar y sujeto a validación, ya que varias de las funcionalidades planteadas requerían contraste con la experiencia práctica de uso de los correspondientes usuarios finales, y con las necesidades reales detectadas. A medida que avanzó este proceso de ingeniería de requerimientos, parte de los elementos de dicha lista fueron reformulados, fusionados o descartados, en función de la evidencia obtenida en las sucesivas rondas de validación.

El detalle completo de esta lista puede consultarse en el Anexo 13.3.

4.1.2. Análisis y consolidación

La etapa de análisis y consolidación tuvo como objetivo revisar en profundidad los requerimientos identificados en cada iteración, con el fin de depurar, clarificar, y documentar la comprensión compartida entre el equipo y el cliente. Esta fase constituyó un momento clave del proceso, ya que permitió transformar los elementos del listado original, muchos de ellos en estado preliminar o ambiguo, en requerimientos comprensibles, priorizados y técnicamente viables.

Para esta tarea el equipo utilizó la herramienta colaborativa *Figma FigJam*, que facilitó el trabajo visual e iterativo del equipo. A través de esta herramienta se representaron los requerimientos, se agruparon por categorías funcionales, se identificaron dependencias entre ellos y se formularon preguntas abiertas destinadas a resolver inconsistencias o dudas conceptuales con el cliente. Este enfoque permitió detectar redundancias, refinar definiciones y construir una base común de entendimiento antes de avanzar hacia la negociación y validación con el cliente.

La Ilustración 7, presenta un ejemplo del resultado obtenido durante la primera iteración de esta fase. En ella se observa el proceso de análisis y mapeo inicial de los requerimientos provistos por el cliente, junto con las preguntas que surgieron durante la revisión interna. Este ejercicio fue fundamental para guiar la siguiente instancia de trabajo colaborativo con el cliente.

Asimismo, la Ilustración 8 muestra un segundo ejemplo correspondiente al análisis de los requerimientos asociados al rol de administrador, en el cual se representa cómo se configuró la gestión de comunidades, moderación de contenidos y administración de ideas, preguntas y comentarios dentro del sistema. Este tipo de diagramas permitió al equipo visualizar interacciones complejas, anticipar posibles conflictos funcionales y definir los límites operativos de cada rol dentro de la plataforma.

Si bien la Ilustración 7 muestra el resultado de la primera iteración, la Ilustración 8 refleja un trabajo realizado en una etapa posterior del proceso, cuando ya se habían incorporado nuevos requerimientos y el conocimiento del dominio era más profundo. Ambas ilustraciones evidencian la evolución del enfoque aplicado en esta fase, que se mantuvo constante a lo largo de las iteraciones, un ciclo de análisis, revisión y depuración destinado a afinar la comprensión funcional y técnica del sistema. De esta manera, cada iteración permitió construir un conjunto progresivamente más preciso y validado de requerimientos, alineado con la maduración del proyecto y con las necesidades reales de los usuarios finales.

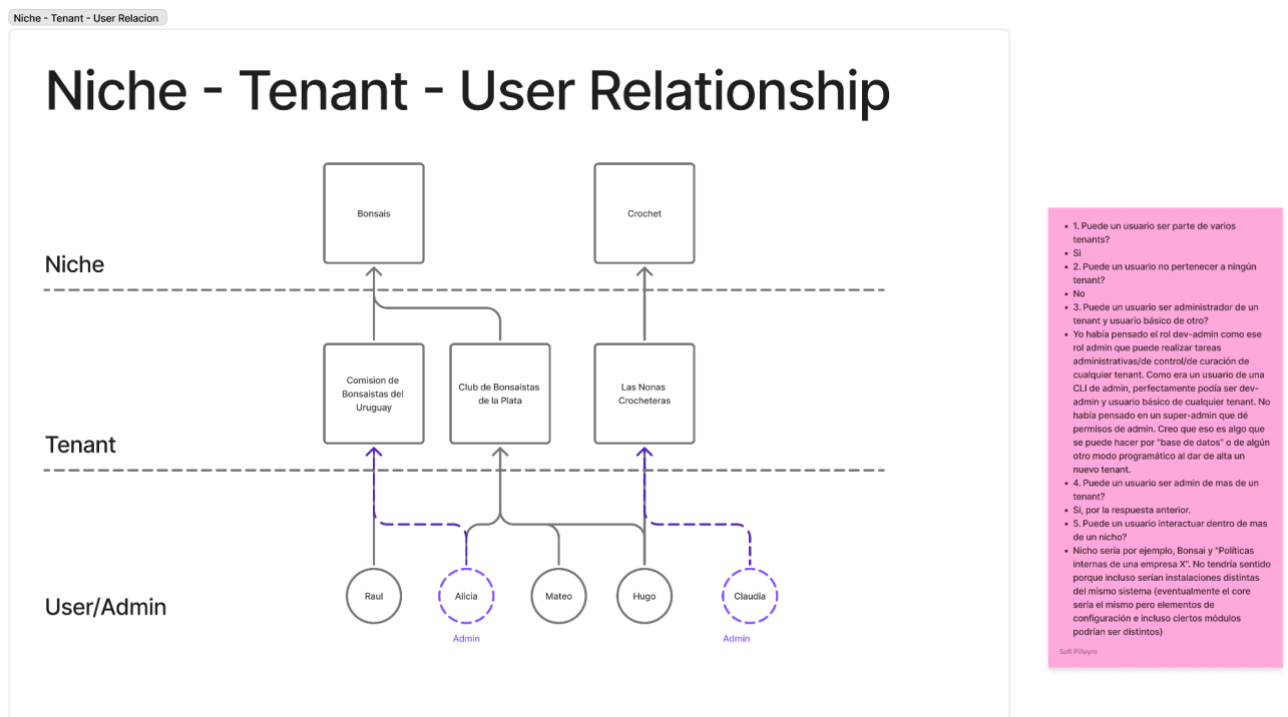


Ilustración 7: Diagrama de entendimiento de los requerimientos iniciales con preguntas para el cliente.



Ilustración 8: Diagrama de los requerimientos de un administrador y como se da la configuración y moderación de la comunidad con comentarios, ideas y preguntas.

4.1.3. Negociación y alineación

En esta etapa, el equipo buscó alcanzar una comprensión compartida junto con el cliente respecto al alcance, las prioridades y el significado funcional de los requerimientos en cada iteración. Esta fase fue esencial para garantizar la coherencia entre la visión estratégica del cliente y las decisiones técnicas tomadas por el equipo, así como para asegurar que cada requerimiento validado respondiera efectivamente a una necesidad real y alcanzable dentro del contexto del sistema.

Durante esta etapa, el equipo presentó al cliente los resultados del análisis previo, incluyendo observaciones, preguntas abiertas y dependencias detectadas. El intercambio permitió resolver ambigüedades, redefinir funcionalidades y establecer prioridades, evitando malentendidos que pudieran derivar en retrabajo en fases posteriores. Pese al intento del equipo, no se logró evitar completamente el retrabajo en ciertas tareas. Estas instancias de negociación se realizaron de manera iterativa y colaborativa, remitiéndose a lo largo de todo el proyecto conforme se refinaban los requerimientos.

En algunas iteraciones, la negociación adoptó una dinámica con una primera etapa asíncrona y luego sesiones vía *Google Meets*. Primero, se enviaban al cliente las preguntas y observaciones derivadas del análisis realizado por el equipo en *FigJam*, lo que permitía al cliente reflexionar sobre las respuestas con mayor profundidad. Posteriormente, se realizaban reuniones de trabajo conjunto para discutir las respuestas, acordar reformulaciones y ajustar la priorización de los requerimientos para la siguiente fase.

La Ilustración 9 muestra un ejemplo de este proceso, en el que se visualizan el estado de una de las preguntas formuladas al cliente y cómo su respuesta permitió clarificar el requerimiento correspondiente. Este tipo de instancias fue clave para reducir la ambigüedad y consolidar el entendimiento mutuo sobre el funcionamiento esperado del sistema.

Maquina de estado de una pregunta

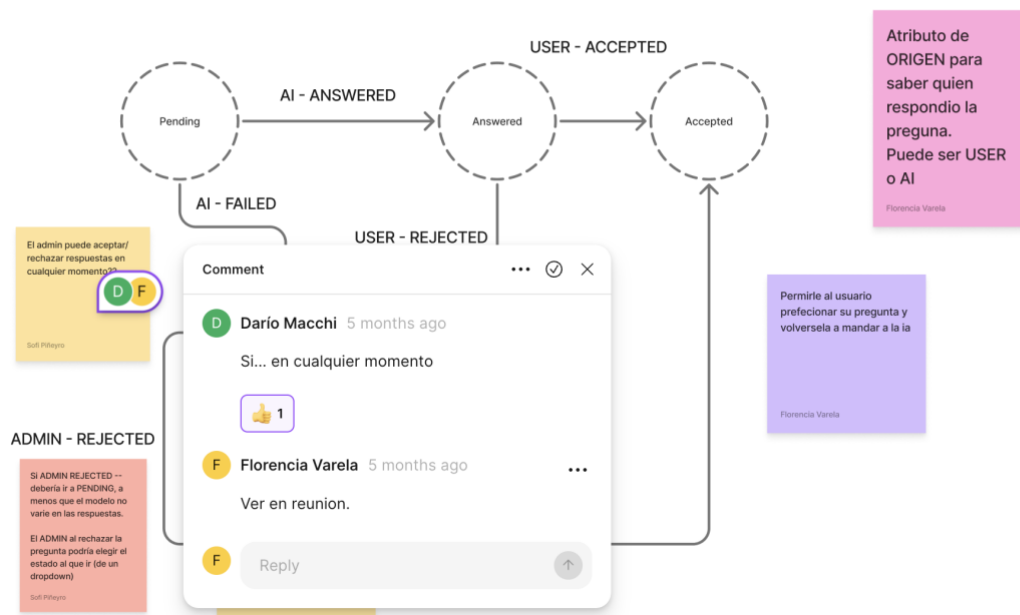


Ilustración 9: Diagrama de estado de una pregunta generado en la etapa de Análisis y consolidación con preguntas realizadas al cliente y su respuesta.

Por otra parte, la Ilustración 10 ejemplifica uno de los tableros de priorización generados durante esta fase en una de sus iteraciones.



Ilustración 10: Priorizaciones realizadas en la etapa de negociación y alineación inicial

4.1.4. Documentación y prototipación

El proceso de documentación y prototipación acompañó de forma transversal a todas las etapas de la ingeniería de requerimientos. Desde las primeras ideas del sistema se elaboró un pretotipo inicial, que sirvió como primera materialización de la solución y permitió validar tempranamente los flujos principales y la lógica de interacción con los usuarios. A pesar de su baja fidelidad, este pretotipo resultó clave para obtener retroalimentación concreta, ya que se utilizó en las primeras sesiones de validación con usuarios reales. Dichas pruebas posibilitaron identificar necesidades, expectativas y oportunidades de mejora tanto a nivel funcional como de navegación.

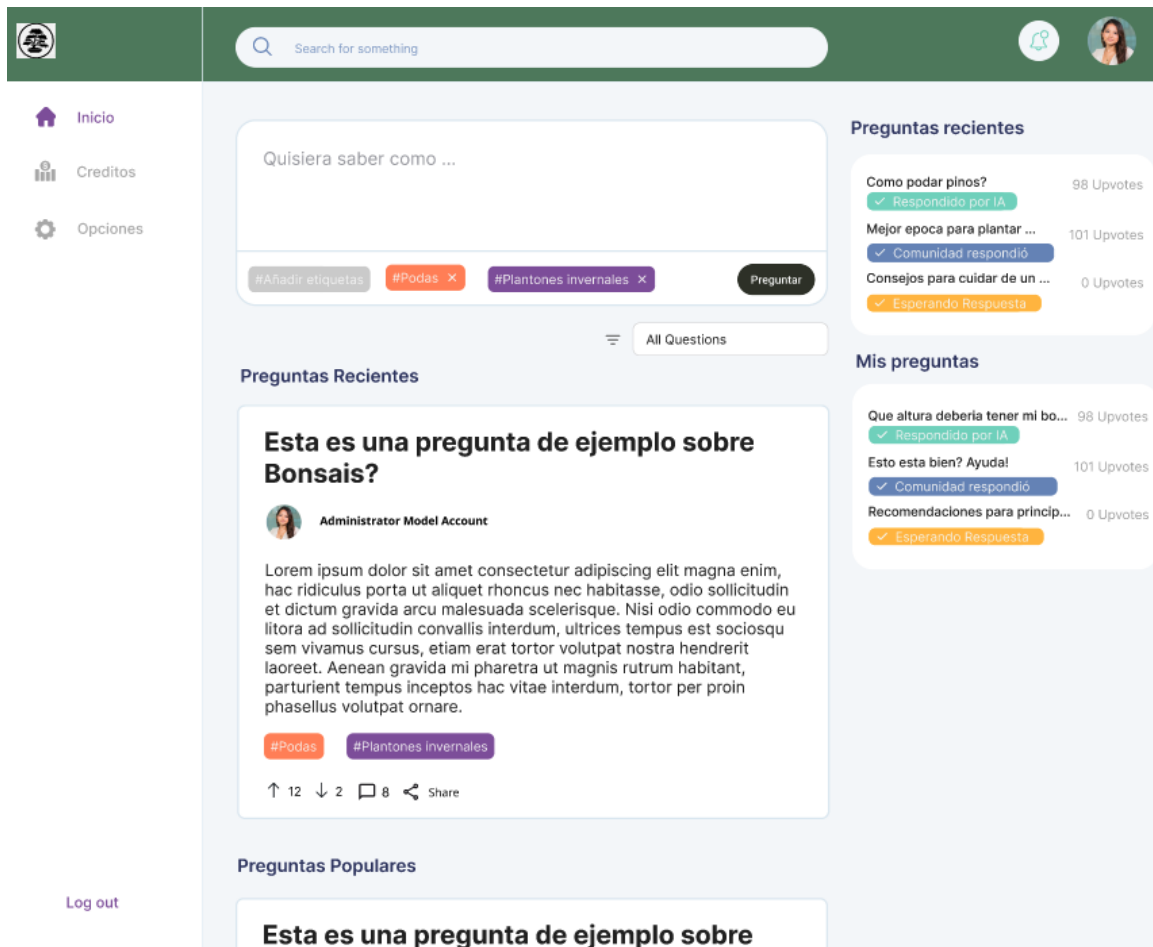


Ilustración 11: Pantalla inicial del prototipo

A partir de las observaciones recogidas en esa instancia, el equipo desarrolló un prototipo de mayor fidelidad, incorporando los ajustes derivados del *feedback* y profundizando en los aspectos visuales y de usabilidad. En esta etapa, las iteraciones se centraron en refinar la experiencia de usuario (UX) y fortalecer la coherencia estética e identidad visual del sistema (UI). Este proceso permitió consolidar una versión más madura del diseño, alineada con la visión del cliente y las necesidades detectadas durante la validación.

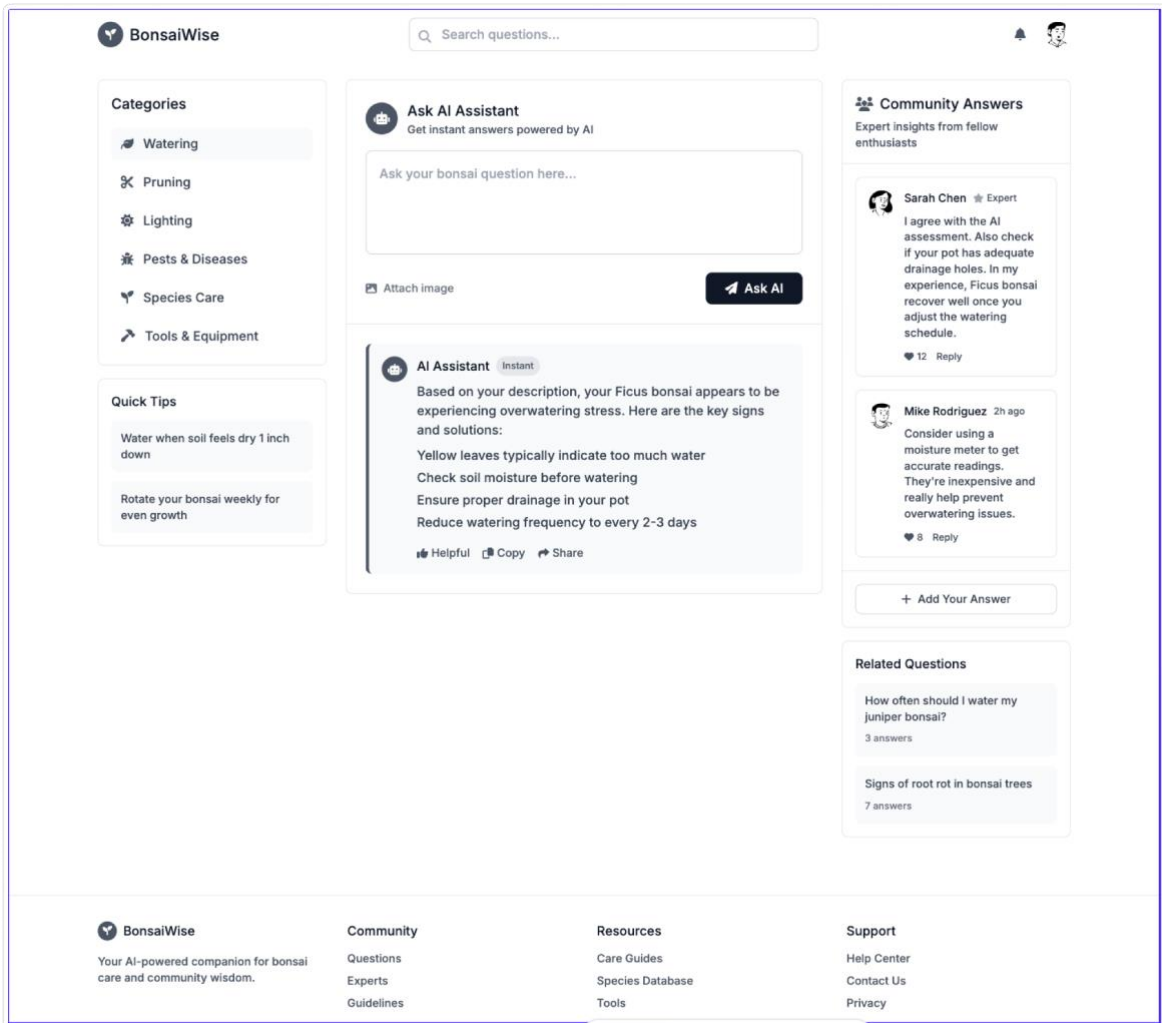


Ilustración 12: Pantalla inicial del prototipo

Por otra parte, la documentación de los requerimientos se concibió como un proceso continuo y colaborativo, orientado a mantener la trazabilidad entre las decisiones de diseño, las validaciones realizadas y los objetivos del sistema. En lugar de optar por una documentación tradicional en formato estático, el equipo priorizó Figma como entorno central de documentación, tomando como ventaja su capacidad para combinar elementos visuales y textuales en un mismo espacio de trabajo.

Cada requerimiento identificado o ajustado durante las iteraciones quedaba registrado directamente sobre los flujos y pantallas del prototipo, acompañado por descripciones que especificaban su propósito, condiciones y dependencias. De esta forma, el equipo podía visualizar no solo qué debía hacerse, sino también por qué se tomaba cada decisión.

Este enfoque facilitó la colaboración entre el equipo y garantizó que todos trabajaran sobre una fuente única de verdad actualizada. Asimismo, la centralización de la información en un entorno visual redujo la fricción entre el diseño conceptual y la implementación técnica, evitando la pérdida de contexto entre etapas.

De manera complementaria, la documentación y los prototipos desempeñaron un rol esencial en la ingeniería de requerimientos, al sistematizar el avance del proyecto y las decisiones adoptadas, y al mismo tiempo facilitar la comunicación bidireccional y la validación continua con el cliente y los usuarios..

4.1.5. Validación con usuarios finales

A partir del conjunto inicial de requerimientos, se llevaron a cabo distintas sesiones de validación con usuarios representativos del público objetivo. Estas actividades tuvieron como propósito contrastar las suposiciones iniciales con evidencia empírica y garantizar que el sistema respondiera efectivamente a las necesidades reales de sus futuros usuarios. En conjunto, las validaciones permitieron ajustar la dirección del proyecto, fortalecer su usabilidad y reafirmar la pertinencia funcional de la solución.

Validación del problema

La primera validación se enfocó en confirmar la existencia y relevancia del problema planteado. Para ello, se elaboró una encuesta dirigida a potenciales usuarios finales, con el objetivo de explorar sus hábitos de búsqueda de información y las principales dificultades experimentadas al intentar acceder a contenidos confiables sobre el cuidado de bonsáis.

Los resultados obtenidos fueron concluyentes: el 79% de los encuestados manifestó buscar información con frecuencia (varias veces por semana), y un 86% consideró que el cuidado de bonsáis requiere conocimientos especializados.

Frecuencia de búsqueda de cuidado de bonsáis

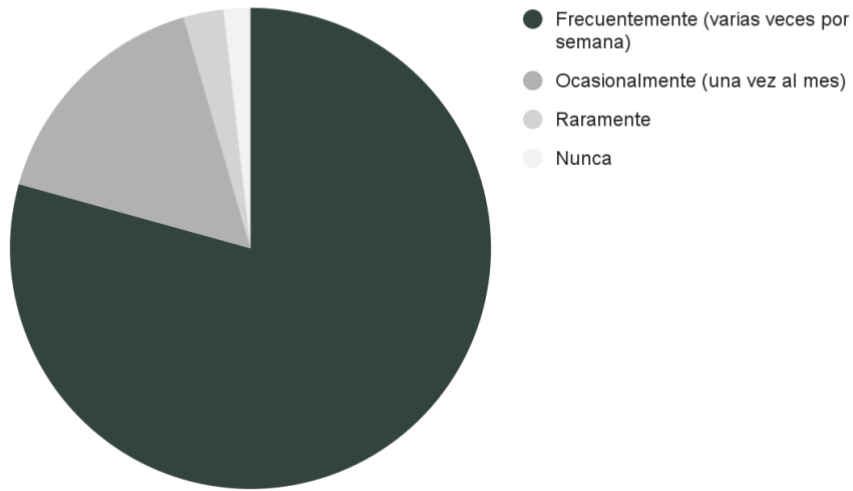


Ilustración 13: Resultados de la validación del problema – Frecuencia de búsqueda de cuidado de bonsáis.

¿Consideras que cuidar bonsáis requiere conocimientos especializados?

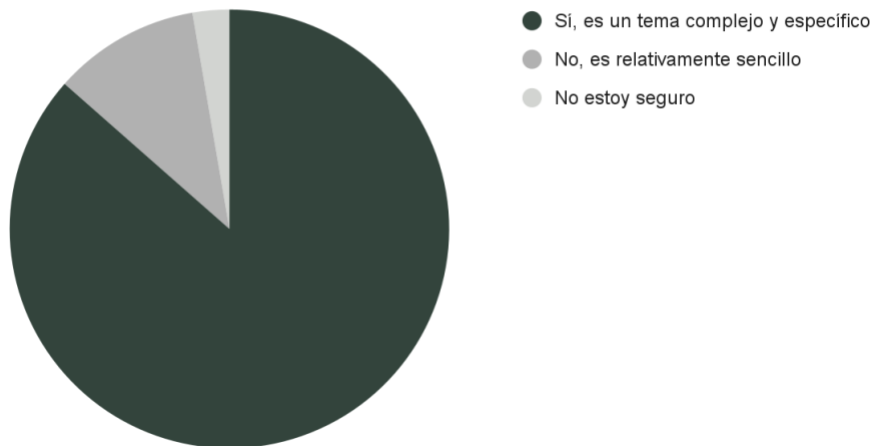


Ilustración 14: Resultados de la validación del problema – Dificultad del cuidado de bonsáis.

Entre los problemas más señalados, se destacaron la falta de profundidad en las explicaciones, las respuestas inconsistentes o contradictorias, y la información poco confiable o sin respaldo.

Problemas más frecuentes al buscar información

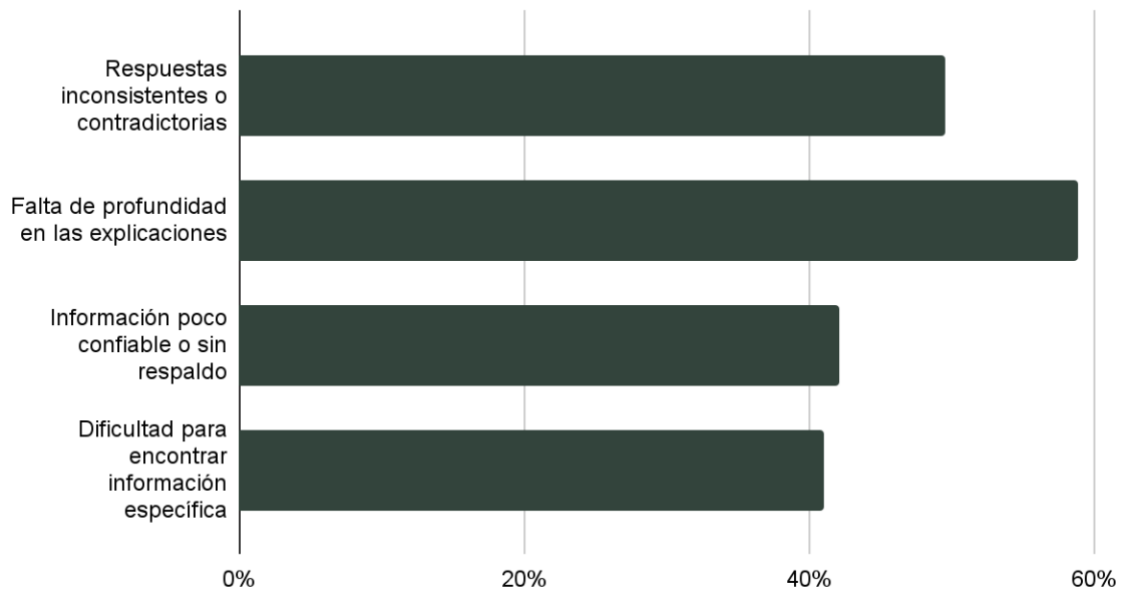


Ilustración 15: Resultados de la validación del problema – Problemas más frecuentes al buscar información sobre bonsáis.

Estos datos confirmaron empíricamente la hipótesis inicial: existe una brecha significativa entre la información disponible y las necesidades reales de los aficionados, lo que justifica el desarrollo de una herramienta especializada que ofrezca respuestas contextualizadas, verificables y adaptadas al nivel de conocimiento del usuario.

La encuesta completa y sus resultados se presentan en el Anexo 13.1.

Validación del prototipo

En una segunda instancia, el equipo buscó validar la adecuación de la propuesta inicial mediante entrevistas y sesiones de prueba con usuarios finales. Se presentó un prototipo que permitió observar de forma práctica cómo se traducen los requerimientos en una experiencia de uso concreta.

Participaron usuarios con perfiles técnicos, pedagógicos y prácticos, lo que enriqueció el proceso al aportar perspectivas complementarias. En general, los participantes valoraron

positivamente la propuesta, destacando su utilidad como espacio centralizado de información confiable y su potencial para distintos niveles de experiencia. La interfaz fue percibida como clara e intuitiva, aunque se sugirió mostrar la pregunta publicada tras el envío y generar etiquetas automáticamente.

Entre las recomendaciones más relevantes surgieron la incorporación de información contextual (clima o región), la personalización del nivel de detalle de las respuestas, la identificación de fuentes, el enfoque en que fuese una plataforma colaborativa, y la inclusión de sistemas de reputación y notificaciones.

En conjunto, las entrevistas confirmaron que el prototipo cumplía con su objetivo principal: ofrecer una experiencia de interacción fluida y significativa. Las observaciones obtenidas sirvieron de guía para refinar la navegación, la personalización de respuestas y la transparencia del contenido generado por la IA.

Los detalles completos de las entrevistas, junto con los resultados individuales y observaciones registradas, se presentan en el Anexo 13.4.

Validación de la solución final

Con el desarrollo avanzado, se realizaron sesiones de validación centradas en la usabilidad y efectividad del sistema. Estas pruebas permitieron observar la interacción real de los usuarios con la plataforma, validar los flujos principales y medir la satisfacción general.

Participaron usuarios de Argentina, Uruguay y Ecuador, con niveles de experiencia variables: desde principiantes que recurrían a foros o grupos de mensajería, hasta aficionados avanzados y docentes especializados. Esta diversidad permitió evaluar la adaptabilidad del sistema a distintos contextos y niveles de conocimiento.

Las pruebas finales confirmaron la madurez y coherencia del producto. El sistema fue percibido como accesible, confiable y con una curva de aprendizaje mínima, validando las decisiones de diseño tomadas en etapas previas. Las observaciones recabadas orientan los próximos pasos hacia la personalización contextual, el fortalecimiento de la comunidad como fuente de conocimiento colectivo y la mejora continua de la experiencia de usuario.

El detalle de las pruebas de usabilidad y satisfacción de los usuarios puede leerse en el capítulo 9, Gestión de calidad.

4.2. Requerimientos

Los requerimientos se organizaron según los distintos roles del sistema: tenant, administrador y usuarios, lo que permitió identificar claramente las responsabilidades y necesidades de cada actor. Además, se definieron requisitos no funcionales y restricciones tecnológicas para garantizar el rendimiento y la alineación con las condiciones del cliente.

4.2.1. Funcionales

En esta sección se presentan algunos de los requerimientos funcionales del sistema donde se debió indagar más sobre el dominio y encontrar un balance entre las necesidades de los usuarios finales y los requerimientos presentados por el cliente. La lista completa de requerimientos funcionales se encuentra en el Anexo 13.5.

4.2.1.1. Requerimientos del Administrador

CLI vs. Sección en la Web

Dentro de los requerimientos iniciales presentados por el cliente se incluía un conjunto de funcionalidades específicas asociadas a una interfaz de línea de comandos (CLI) administrativa (ver Anexo 13.5 y capítulo 4). Sin embargo, durante las conversaciones iniciales mantenidas en la etapa de anteproyecto, el cliente aclaró que la necesidad de una CLI no respondía a una restricción técnica, sino a su propia preferencia personal, dado su perfil técnico.

Durante el proceso de discovery, en el proceso de análisis y consolidación de los requerimientos, surgió la propuesta de permitir que el administrador pudiera realizar sus tareas directamente desde la misma plataforma web. De esta forma, se buscaba unificar la experiencia de administración e interacción dentro de un mismo entorno, evitando la complejidad innecesaria de una herramienta separada basada en línea de comandos.

Cuando esta alternativa fue presentada al cliente durante la etapa de negociación y alineación, si bien no manifestó un interés particular en invertir esfuerzos significativos en el desarrollo de una sección administrativa compleja, comprendió la preocupación del

equipo respecto a que una CLI podría resultar poco accesible para otros usuarios. En consecuencia, se acordó incorporar una sección administrativa dentro de la web app, manteniendo su alcance simple y funcional. Paralelamente, se despriorizaron algunos requerimientos vinculados al rol de administrador, como los ítems 4.1.6.2. *Generar QR para especie* y la administración de créditos, debido a que el cliente no tenía del todo claro su propósito y se consideró que no aportaban valor sustancial al producto en la etapa de MVP.

Finalmente los requerimientos del administrador se plantearon todos para una aplicación web como el que se encuentra a continuación.

RFA1. Carga de datos

El sistema deberá brindar al administrador la posibilidad de realizar la carga de datos a la comunidad, contemplando distintos formatos de entrada según la naturaleza de la información. Por un lado, se permitirá la incorporación de archivos en formato Markdown, utilizados para alimentar la base de conocimiento específica de cada comunidad. Por otro lado, se habilitará la carga de datos estructurados en formato Excel, destinados principalmente al nomenclátor provisto por el cliente.

Una vez cargados, los datos deberán integrarse a la base de conocimiento de la comunidad correspondiente, asegurando la correcta segregación de la información entre distintos dominios.

Este requerimiento funcional se encuentra alineado con el documento de requerimientos iniciales del cliente, específicamente en el apartado 4.1.1. *Ingest de datos*, el cual establece la necesidad de que los administradores dispongan de un mecanismo seguro y flexible para abastecer de datos al sistema.

Feedback sobre preguntas realizadas por usuarios

Uno de los requerimientos iniciales planteados por el cliente establecía lo siguiente:

4.1.4. Responder pregunta como si fuera la AI.

“Para no dejar preguntas sin responder correctamente o completamente, como dev-admin quiero poder editar la respuesta dada por la AI o responder a una pregunta que no pudo ser respondida automáticamente (estado AI MISS).”

Desde el primer análisis, este requerimiento generó debate dentro del equipo. La posibilidad de que un administrador pudiera responder en nombre de la inteligencia artificial despertó dudas éticas y técnicas. Por un lado, se consideró que esto podía afectar la transparencia del sistema, haciendo creer al usuario que la respuesta provenía del modelo cuando en realidad había sido escrita por una persona. Por otro lado, se entendió que esta lógica complicaba el mantenimiento y el control de versiones, al perder claridad sobre qué contenido era generado automáticamente y cuál había sido intervenido.

Durante una primera iteración, el equipo propuso al cliente una alternativa: permitir que las respuestas editadas se marcaran con una etiqueta visible, por ejemplo “editada”. El cliente, sin embargo, consideró que bastaba con registrar los cambios internamente y no veía necesario mostrar esa información en la interfaz.

Aun con ese registro interno, el equipo mantuvo su preocupación por la posible pérdida de confianza del usuario. Por ese motivo, el tema volvió a discutirse con el cliente en una revisión posterior, esta vez poniendo el foco en la experiencia de usuario y la transparencia del sistema. Tras analizarlo en conjunto, ambas partes coincidieron en que el enfoque original podía resultar confuso.

Como resultado, el requerimiento evolucionó: en lugar de permitir que el administrador modificara o reemplazara respuestas, se optó por que pudiera agregar comentarios asociados a ellas. De esta forma, si la IA no respondía correctamente o su respuesta era incompleta, el administrador podía complementar la información sin alterar el contenido original.

RFA5. Dejar comentario en una respuesta de IA

El sistema deberá permitir al administrador registrar comentarios sobre las respuestas generadas por la IA, con el objetivo de aportar retroalimentación o contexto adicional. Esta funcionalidad asegura que las respuestas puedan ser evaluadas y mejoradas de manera ética, sin modificar directamente la salida de la IA.

Este requerimiento funcional se origina a partir del requerimiento *4.1.4. Responder pregunta como si fuera la AI* del documento de requerimientos iniciales del cliente, que originalmente permitía editar respuestas de la IA. Tras conversaciones con el cliente, se acordó que modificar las respuestas no era éticamente adecuado, por lo que se definió esta solución alternativa de dejar comentarios como mecanismo de retroalimentación.

Esta solución resolvió las preocupaciones éticas iniciales y mejoró la trazabilidad, asegurando que cada intervención humana quedara claramente diferenciada del comportamiento autónomo de la IA. En la Ilustración 16 se muestra cómo se implementaron estos comentarios dentro de la interfaz.

 **Respuesta de IA** 👍 Útil (0) 🚩 No me resultó útil (0)

hace 1d

Parece que tu bonsái está teniendo problemas, y aunque no tengo información específica sobre tu árbol, puedo ofrecerte algunos consejos generales sobre el cuidado de los bonsáis.

Revisar el riego: Asegúrate de que tu bonsái esté recibiendo la cantidad adecuada de agua. Un sustrato demasiado seco o demasiado húmedo puede causar que el árbol se vea apagado. Es importante que el agua drene bien y que las raíces no se encharquen.

Luz adecuada: Verifica que tu bonsái esté recibiendo suficiente luz. La mayoría de los bonsáis requieren luz indirecta brillante. Si está en un lugar muy oscuro, puede debilitarse.

Nutrientes: Considera fertilizar tu bonsái con un abono adecuado. Los bonsáis pueden necesitar nutrientes adicionales, especialmente durante la temporada de crecimiento.

Transplante: Si las raíces están saliendo por los agujeros de drenaje o si el sustrato se seca rápidamente, puede ser un signo de que necesita ser trasplantado. Esto ocurre cuando las raíces han llenado todo el espacio disponible.

Revisar plagas y enfermedades: Inspecciona tu bonsái en busca de signos de plagas o enfermedades, que también pueden causar que se vea triste.

Si sigues estos consejos y observas cambios, tu bonsái podría recuperarse. Si la situación no mejora, podría ser útil consultar a un experto en bonsáis.

 **Administrator Comment** Admin

Para poder identificar correctamente la causa del estado de tu bonsái es mejor indicar la especie.

Ilustración 16: Comentarios de un administrador

4.2.1.2. Proceso de realizar una pregunta y obtener una respuesta

Como parte de los requerimientos iniciales, el cliente había planteado la necesidad de contar con un proceso de pregunta y respuesta asistido por inteligencia artificial, el cual contemplara distintos estados y ofreciera a los usuarios no administradores la posibilidad de intentar responder o mejorar las preguntas publicadas.

Durante la etapa de *discovery*, y con el objetivo de comprender con mayor precisión las expectativas del cliente, se elaboró un diagrama en formato de máquina de estados, que

permitió visualizar y analizar el flujo general del proceso. Esta primera versión se construyó principalmente a partir de la interpretación del equipo sobre los requerimientos iniciales presentados por el cliente.

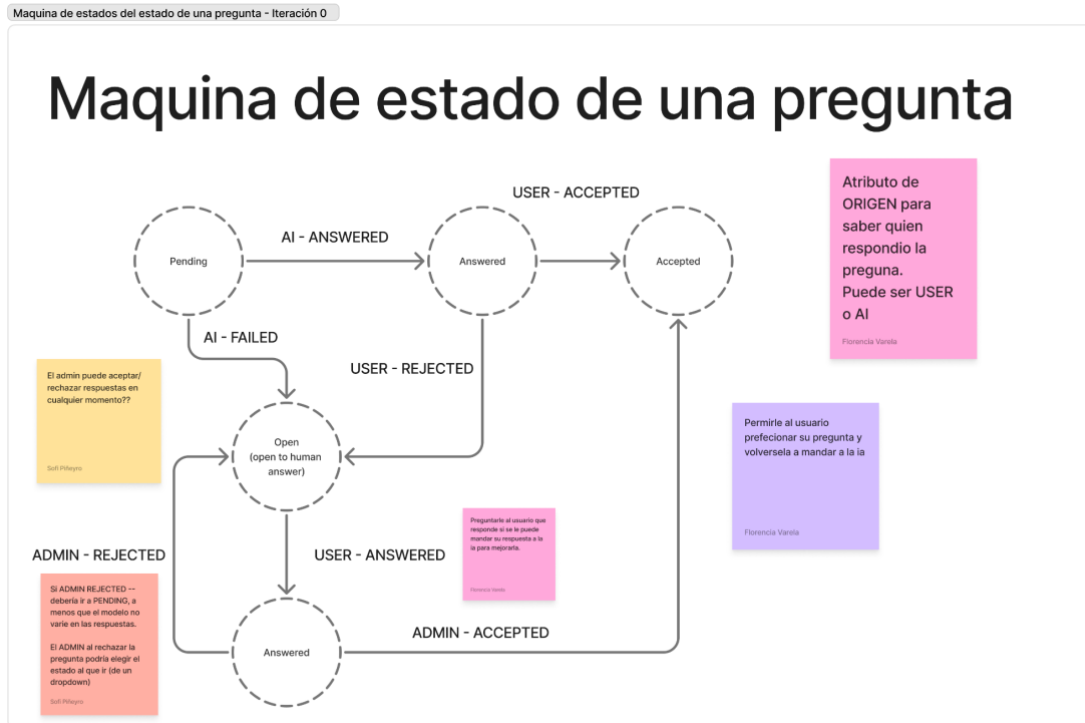


Ilustración 17: Máquina de estados con preguntas para el cliente - Versión 1

A partir de esta versión preliminar, se mantuvo una reunión de negociación con el cliente, donde se revisaron las suposiciones y se discutieron los resultados obtenidos. Como resultado de ese intercambio, se elaboró una segunda versión de la máquina de estados, incorporando las conclusiones derivadas de dicha negociación. En esta versión, el cliente propuso ampliar el modelo de estados para incluir un mecanismo de feedback por parte del administrador, que permitiera retroalimentar al modelo de IA y favorecer su aprendizaje continuo. El propósito de esta segunda versión fue, por tanto, comprender de forma más precisa el flujo funcional que el cliente tenía en mente.

Maquina de estado de una pregunta - V2

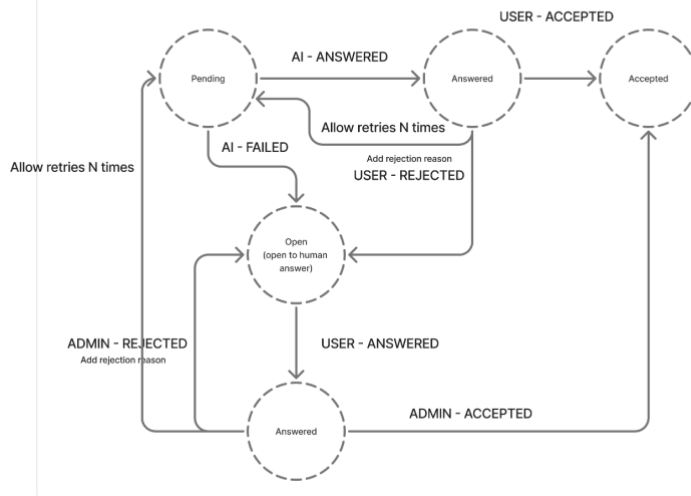


Ilustración 18: Máquina de estados - Versión 2

Una vez consolidada esta comprensión, durante las entrevistas de validación de la solución surgió un nuevo hallazgo: los usuarios percibían el proceso como excesivamente complejo, y manifestaban dificultades para comprender cuándo aceptar o rechazar una respuesta impactaba en el sistema. Considerando este feedback y el hecho de que la comunidad de bonsaistas es pequeña y valora fuertemente el sentido de comunidad y participación, el equipo propuso una versión más simplificada del flujo. Esta nueva propuesta otorgaba un rol más central al administrador y eliminaba la retroalimentación directa al modelo, dado que este proceso requería consentimiento explícito por parte de los usuarios, quienes en las entrevistas expresaron incomodidad con la posibilidad de que sus aportes fueran utilizados para entrenar la IA.

Maquina de estado de una pregunta - V3

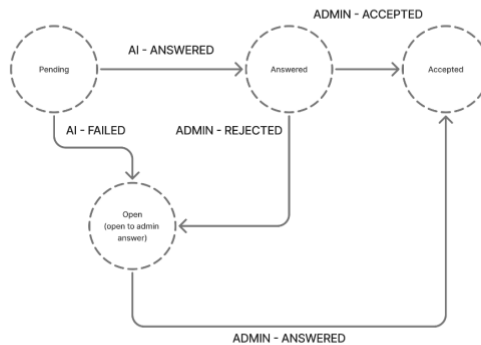


Ilustración 19: Máquina de estados - Versión 3

En una reunión posterior, el cliente manifestó su conformidad con la simplificación y acordó que, por motivos de tiempo y priorización, no resultaba necesario incorporar el mecanismo de retroalimentación en el MVP.

Durante la etapa de prototipado y las entrevistas de validación correspondientes, se detectó un nuevo patrón de comportamiento: los usuarios no comprendían del todo el rol del administrador y expresaban preferencia por un modelo donde todas las preguntas permanecieran abiertas a la participación de la comunidad.

A partir de estos hallazgos, el equipo decidió realizar una nueva sesión con el cliente para profundizar en las razones detrás de la centralidad del rol del administrador. En dicha reunión, el cliente explicó que necesitaba mantener control sobre las respuestas generadas por el sistema de IA y verificar la validez tanto de las respuestas automáticas como de las aportadas por otros usuarios. Asimismo, argumentó que en otros nichos (posibles comunidades), por ejemplo en áreas de Recursos Humanos dentro de una empresa, no sería deseable permitir respuestas comunitarias, dado el riesgo de desinformación o confusión.

Con base en estas aclaraciones, el equipo desarrolló una cuarta versión, introduciendo el concepto de “modo de operación”. Se definieron dos modalidades: modo moderado y modo libre, junto con la capacidad del administrador para cambiar el modo operativo de la comunidad. Además, se incorporó la posibilidad de que los usuarios pudieran brindar

feedback sobre las respuestas recibidas, y, en coherencia con las preocupaciones del cliente, sobre la validez de una respuesta, se añadió una configuración para permitir o no que el autor de la pregunta seleccione la respuesta que considere más útil o adecuada, incluyendo las generadas por la IA.

En base a estas modificaciones se descartó el concepto de máquina de estado y se planteó los dos flujos que puede recorrer una respuesta luego de ser creada independientemente de si fue un usuario o el sistema de IA el autor.

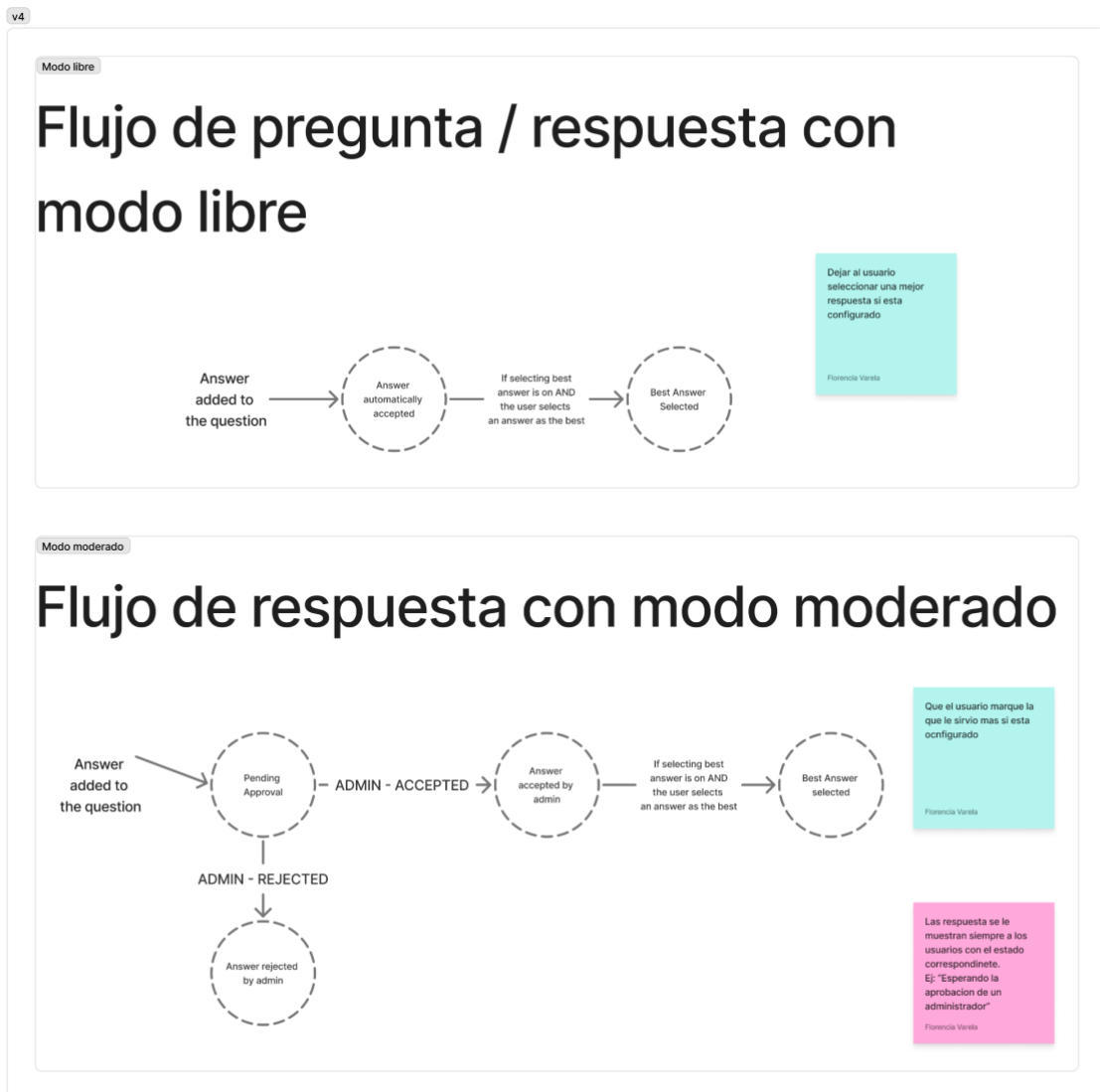


Ilustración 20: Flujo de una respuesta

El Modo Libre opera bajo un principio de confianza en la comunidad y rapidez: una vez que una respuesta es añadida a la pregunta, es automáticamente aceptada y queda inmediatamente visible para los usuarios. El flujo concluye cuando el autor de la pregunta

decide finalizar el proceso al seleccionar una de las respuestas como la mejor, asumiendo la responsabilidad total de la validación y permitiendo la autonomía en el cierre de la pregunta, siempre y cuando esta funcionalidad esté activada.

El Modo Moderado establece un control riguroso de calidad sobre cada respuesta, incorporando la validación obligatoria del administrador como etapa intermedia del flujo. Una vez enviada, la respuesta ingresa a un estado de aprobación pendiente, donde debe ser revisada y evaluada. Si el administrador la aprueba (*ADMIN – ACCEPTED*), la respuesta se considera validada y se muestra al resto de los usuarios con la correspondiente indicación de aprobación. En cambio, si la respuesta es rechazada (*ADMIN – REJECTED*), esta no puede ser votada ni seleccionada como la mejor, aunque permanece visible para los usuarios, claramente marcada como rechazada. Solo tras completarse esta validación administrativa, y siempre que la configuración lo habilite, el autor de la pregunta puede seleccionar la respuesta aprobada como la mejor, cerrando así el ciclo del proceso.

A partir de esta última iteración se definieron requerimientos funcionales del administrador como los siguientes:

RFA6. Rechazar respuesta brindada por la IA en modo moderado

El sistema deberá permitir al administrador rechazar respuestas generadas por la IA cuando estas no cumplan con los criterios de calidad o precisión establecidos para la comunidad. Al rechazar una respuesta, la pregunta asociada será marcada como RECHAZADA, asegurando que los usuarios no interpreten información incorrecta como válida. Esta funcionalidad es aplicable únicamente cuando la comunidad opera en modo moderado, en el cual cada respuesta recibirá inicialmente el estado PENDIENTE hasta ser ACEPTADA o RECHAZADA por un administrador, garantizando la depuración y calidad de la información.

Este requerimiento funcional se adapta del requerimiento 4.1.4. *Responder pregunta como si fuera la AI* del documento de requerimientos iniciales del cliente, que originalmente permitía al administrador editar respuestas de la IA. Tras analizar aspectos éticos y de consistencia, se

determinó que rechazar respuestas inapropiadas en lugar de modificarlas ofrece un mecanismo más transparente de control y retroalimentación, alineado con las prácticas de moderación de la comunidad.

RFA10. Configuración de comunidad

El sistema deberá permitir al administrador definir aspectos clave del funcionamiento de cada comunidad, incluyendo el modo de operación y la posibilidad de seleccionar la mejor respuesta. Las comunidades podrán operar en modo moderado, donde las respuestas son revisadas y validadas por un administrador antes de mostrarse, o en modo libre, donde las respuestas son aceptadas directamente, favoreciendo una interacción más ágil.

Además, se podrá habilitar la funcionalidad de selección de la mejor respuesta, que permitirá a los usuarios resaltar la opción más relevante para cada pregunta. En comunidades moderadas, únicamente podrán marcarse como mejores las respuestas previamente validadas por un administrador, mientras que en comunidades libres cualquier respuesta podrá ser seleccionada. Esta configuración podrá modificarse en cualquier momento según las necesidades de la comunidad.

También se extrajeron algunos requerimientos funcionales para los usuarios:

RFU5. Agregar respuesta

El sistema deberá permitir que un usuario agregue respuestas a preguntas, complementando o corrigiendo la información proporcionada por la IA. Esto asegura que todas las preguntas puedan ser respondidas de manera completa, incluso cuando la IA no genere una respuesta adecuada, y permite a los usuarios involucrarse activamente en la comunidad.

Este requerimiento funcional se adapta del requerimiento *4.2.10.2. Responder preguntas* del documento inicial, garantizando la continuidad del ciclo pregunta-respuesta y la participación de los miembros.

RFU6. Seleccionar mejor respuesta

El sistema deberá permitir que el usuario seleccione una única mejor respuesta para su pregunta, siempre considerando las restricciones del modo de operación. En modo moderado, sólo se podrá seleccionar entre respuestas previamente ACEPTADAS; en modo libre, cualquier respuesta disponible podrá ser marcada como la mejor.

Este requerimiento funcional se adapta de 4.2.10.3. *Aceptar respuesta* y deriva del RFT3. *Selección de mejor respuesta*, integrando la dependencia del modo de operación de la comunidad para garantizar que solo contenido validado pueda ser reconocido como la mejor respuesta.

Finalmente luego de desarrollar la funcionalidad el usuario puede ver las respuestas (Ilustración 21 y 22) y el administrador puede rechazar o aceptar la respuesta (Ilustración 23 y 24).

The screenshot displays three response cards in a list. Each card includes a user profile picture and name, a status badge, a timestamp, utility votes, and the response text.

- Response 1:** User **Jhoanna Smith** (JS) posted 7 days ago. Status: **Mejor Respuesta** (Best Answer). Utility: 1 useful, 0 not useful. Text: "Para APIs REST profesionales recomiendo: 1) Usar códigos de estado HTTP correctos, 2) Implementar HATEOAS, 3) Versionado en URL (/api/v1/), 4) Documentación con OpenAPI/Swagger, 5) Rate limiting y caching." Accepted by Jhoanna Smith.
- Response 2:** User **Alice Johnson** (AJ) posted 7 days ago. Status: **Aprobada por Administrador** (Approved by Administrator). Utility: 1 useful, 0 not useful. Text: "Estructura consistente de respuestas es clave: { \"data\": {}, \"meta\": {}, \"errors\": [] }. También implementa filtrado, paginación y sorting estándar."
- Response 3:** User **John Doe** (JD) posted 7 days ago. Status: **Pendiente** (Pending). Utility: 1 useful, 0 not useful. Text: "GraphQL > REST. Cambio mi voto, las APIs REST están obsoletas en 2024."

Ilustración 21: Ejemplo de visualización de respuestas aprobadas, pendientes y seleccionadas como mejor respuesta.

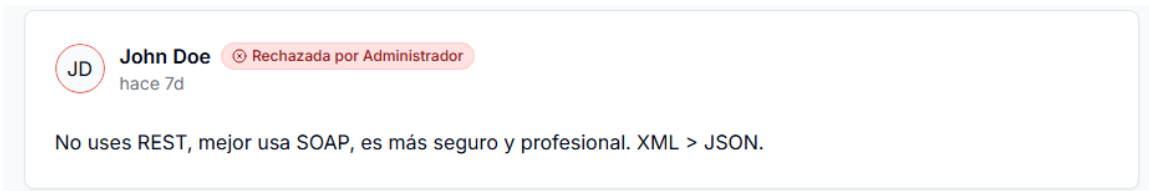


Ilustración 22: Ejemplo de visualización de respuesta rechazada.

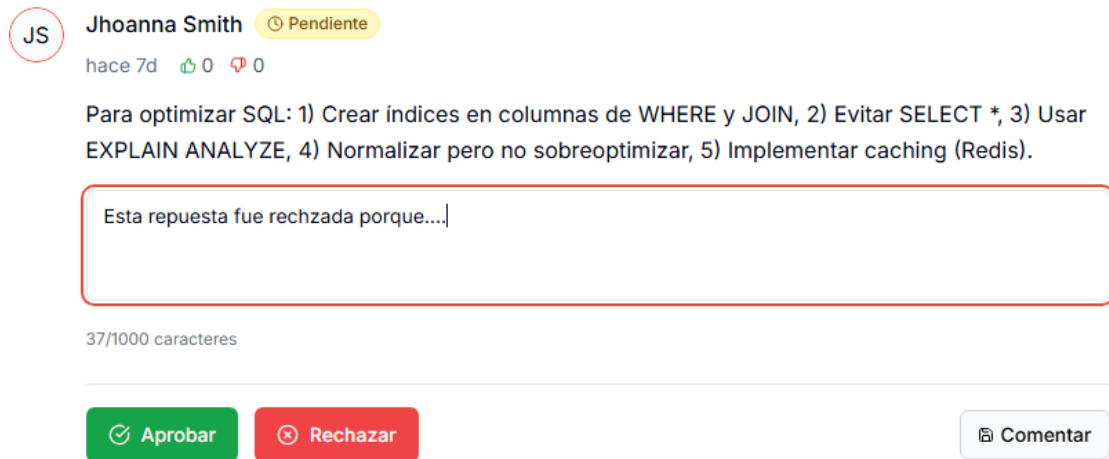


Ilustración 23: Visualización de comentario de un administrador en una respuesta rechazada.

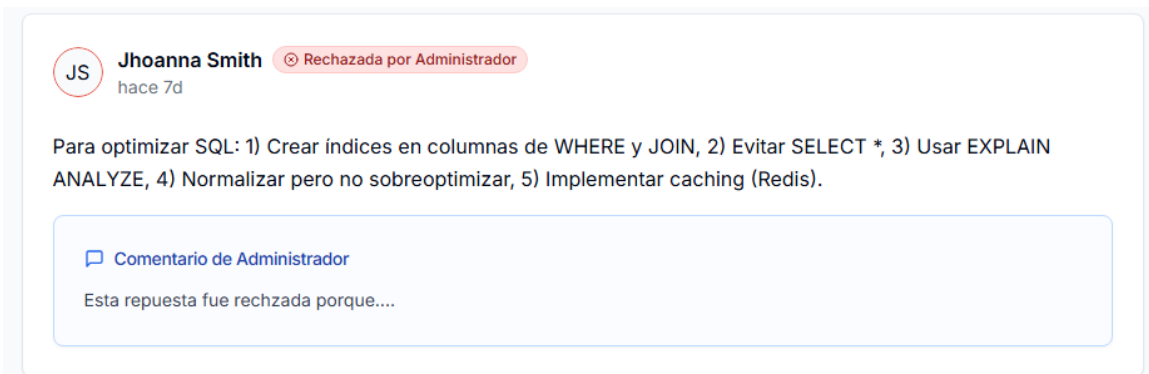


Ilustración 24: Visualización de aceptar/rechazar respuesta.

4.2.2. Requerimientos funcionales del sistema de IA

Esta sección describe las capacidades del sistema de inteligencia artificial para generar respuestas precisas, coherentes y contextualizadas, adaptándose a distintos nichos de conocimiento. Se consideran aspectos como el manejo de términos equivalentes, la interpretación de conceptos según el contexto y la provisión de información general o buenas prácticas cuando no se dispone de datos específicos, garantizando así la consistencia y calidad de las respuestas.

Es importante destacar que el requerimiento RFSIA4 correspondiente al *Caso 4 Tipo de clima* (ver Anexo 13.3), relativo a la provisión de información dependiente de las condiciones climáticas seleccionadas por el usuario, fue descartado, dado que el cliente lo consideró difícil de generalizar a distintos dominios.

RFSIA1. Mismo concepto, distintos términos locales o nombres comunes

El sistema debe reconocer que un mismo concepto puede expresarse con distintos nombres, términos locales o sinónimos según el contexto, región o idioma, y debe mapear correctamente esos términos equivalentes para ofrecer respuestas consistentes basadas en el concepto central.

Ejemplo: Nicho de Apicultura

- *Tetragonisca angustula* → “Yateí” en Argentina
- *Tetragonisca angustula* → “Angelita” en Colombia

Preguntas y respuestas esperadas:

P (Argentina): ¿Cómo alimento a las yateí en invierno?

P (Colombia): ¿Cómo alimento a las angelitas en invierno?

R: La *Tetragonisca angustula* se alimenta con jarabe de azúcar (2:1) en invierno cuando no hay floración.

RFSIA2. Mismo término, distintos conceptos según contexto

El sistema debe identificar que un mismo término puede referirse a distintos conceptos según el contexto, país o área de conocimiento, y generar respuestas precisas adaptadas a la interpretación correcta del término en cada caso.

Ejemplo: Nicho de Viticultura

- *Malbec* en Francia → *Côt* (busca estructura y longevidad)
- *Malbec* en Argentina → *Malbec* (busca fruta y sedosidad)

Preguntas y respuestas esperadas:

P: ¿Qué técnica de vinificación es más recomendable para el *Malbec*?

R (Francia): Para el *Côt* francés, se recomienda la maceración prolongada post-fermentación y el añejamiento en roble francés. Esto estabiliza el color y ablanda los taninos firmes, produciendo un vino de gran estructura y apto para una larga guarda.

R (Argentina): Para el *Malbec* argentino, la clave es la maceración suave y la micro-oxigenación controlada. Se busca el equilibrio, utilizando a menudo el concreto para resaltar la pureza de la fruta y conseguir esa textura aterciopelada y redonda que lo caracteriza.

RFSIA3. Información general cuando no hay datos específicos

Cuando no se dispone de información precisa sobre un elemento específico, el sistema debe generalizar utilizando conocimientos de un nivel superior o categoría más amplia, proporcionando información útil y relevante basada en el grupo o categoría correspondiente.

Ejemplo: Nicho de Arqueoastronomía

Los siguientes lugares son todos considerados observatorios para la arqueoastronomía.

- Observatorio de Chankillo (Perú)
- Caracol de Chichén Itzá (México)
- Stonehenge (Inglaterra)

Pregunta y respuesta esperada:

El sistema no tiene información específica del observatorio de Chankillo pero sí de otros observatorios.

P: ¿Para qué servía el observatorio de Chankillo?

R: No tengo información específica del observatorio de Chankillo. Pero la mayoría de los observatorios prehispánicos estaban diseñados para seguir el movimiento solar y lunar, regulando calendarios agrícolas y ceremoniales.

RFSIA5. Generalidades y buenas prácticas

Cuando no se dispone de información concreta, el sistema debe poder proporcionar recomendaciones generales, buenas prácticas o lineamientos universales aplicables a la mayoría de los casos, ofreciendo orientación útil aun sin datos específicos indicando que la respuesta puede no ser exacta a falta de información.

Ejemplo: Nicho de Terrarios de Reptiles

Pregunta y respuesta esperada:

El sistema no tiene conocimiento de cuidados luego de trasladar un camaleón pantera, pero sí de otros camaleones.

P: ¿Qué cuidados debo tener luego de trasladar un camaleón pantera a un nuevo terrario?

R: No tengo información concreta de los cuidados a tener luego de trasladar un camaleón pantera. Pero si te puedo dar información sobre cuidados a tener para la mayoría de los reptiles: conviene reducir la manipulación, mantener la humedad estable y ofrecer refugios visuales para reducir el estrés post-traslado.

4.2.3. No funcionales

Los requerimientos no funcionales de esta plataforma se centran en garantizar que el sistema de IA especializado en nichos de conocimiento cumpla con los estándares de calidad necesarios para su adopción y escalabilidad. Dado el contexto del proyecto, una plataforma extensible a múltiples dominios de conocimiento especializado, estos requerimientos abordan aspectos críticos como la flexibilidad para permitir la expansión a nuevos nichos, la adecuación funcional para generar confianza en las respuestas de la IA, y la usabilidad para garantizar la adopción por parte de usuarios con diferentes niveles de experiencia tecnológica.

RNF1. Implementación como aplicación web

Este requerimiento surge de un requerimiento explícito del cliente, quien especificó que la solución debe implementarse como aplicación web accesible a través de navegadores estándar, proporcionando una interfaz de usuario intuitiva y responsiva que funcione

correctamente en dispositivos móviles, tablets y computadoras de escritorio. La aplicación debe ser compatible con los principales navegadores web y no requerir instalación de software adicional por parte del usuario.

En la Ilustración 25 a continuación, se deja una imagen ilustrativa de cómo el sistema se adapta a las diferentes pantallas de diversos dispositivos móviles.

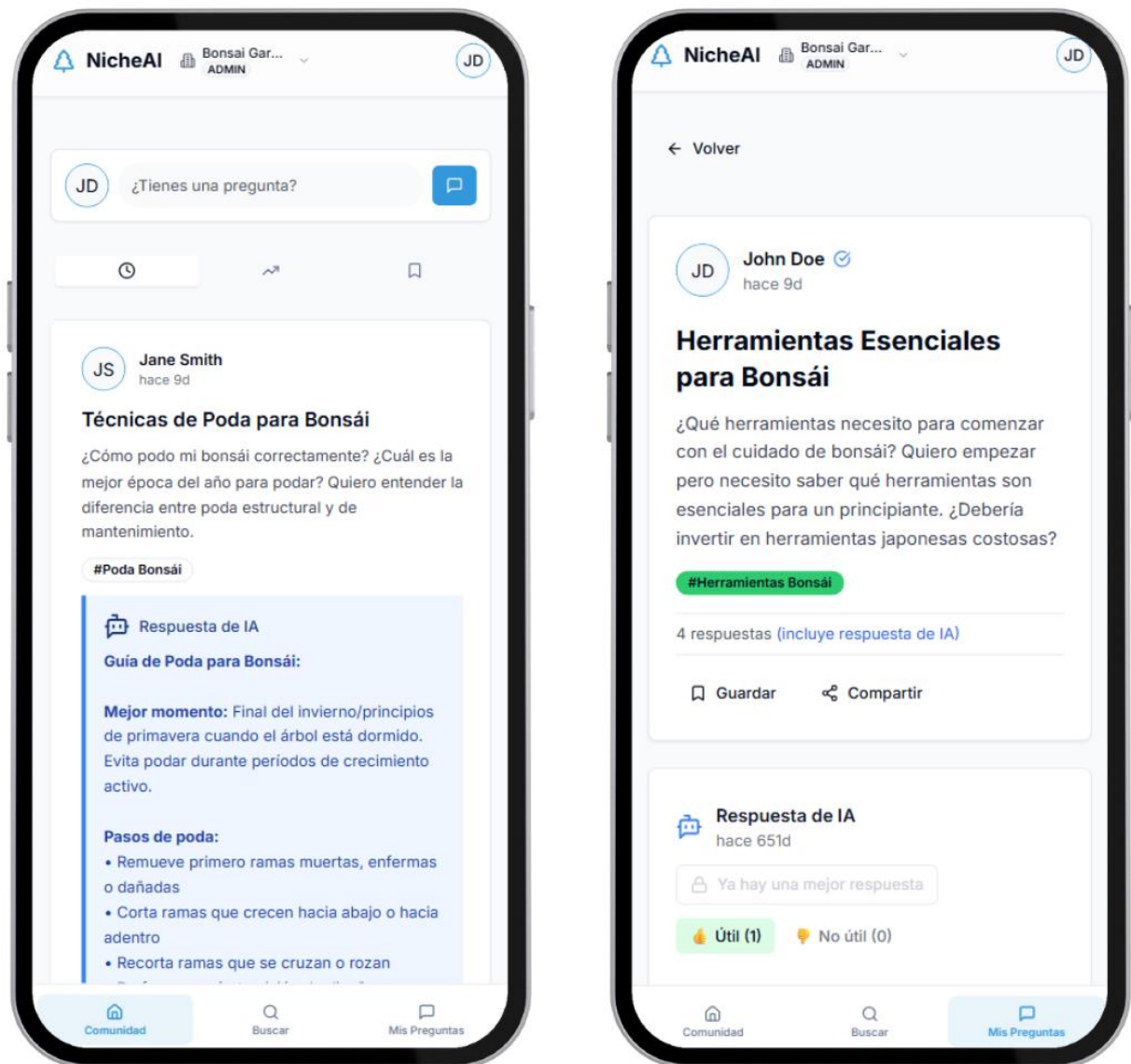


Ilustración 25: Visualización desde dispositivo iPhone 15.

RNF2. Integración de modelos de lenguaje de gran escala (LLMs)

Este requerimiento surge de un requerimiento explícito del cliente, quien especificó que la plataforma debe estar basada en LLMs. El sistema debe utilizar modelos de lenguaje de gran escala como componente central para la generación de respuestas especializadas. La plataforma debe estar diseñada específicamente para aprovechar las capacidades de procesamiento de lenguaje natural de los LLMs, implementando técnicas como RAG o *Agentic RAG* para proporcionar respuestas contextualizadas y precisas en dominios de conocimiento especializados.

RNF3. Despliegue del frontend en Vercel

Este requerimiento surge de un requerimiento explícito del cliente, quien especificó el uso de *Vercel* como plataforma de despliegue. El sistema debe utilizar *Vercel* como plataforma de despliegue para la aplicación *frontend*, cumpliendo con los requerimientos específicos del cliente. La aplicación debe estar configurada para aprovechar las capacidades de *Vercel*, incluyendo su integración con *Next.js*, despliegues automáticos desde el repositorio de código fuente. El proceso de despliegue debe ser automatizado y no requerir intervención manual para actualizaciones del *frontend*.

RNF4. Precisión de respuestas

El sistema debe garantizar un nivel de precisión y exactitud suficiente en las respuestas generadas por el modelo LLM para cumplir con las expectativas de los usuarios finales. Este requerimiento surge del atributo de calidad Adecuación Funcional, ya que es crítico para generar confianza en las respuestas de la IA en dominios especializados. Se considera aceptable una satisfacción con las respuestas obtenidas mayor a 4.5/5, evaluada mediante votaciones de usuarios y testing automatizado que evalúe aspectos como precisión, recall y ausencia de alucinaciones en las respuestas brindadas.

RNF5. Facilidad de uso para usuarios con limitaciones tecnológicas

El sistema debe permitir que el 90% de los usuarios nuevos puedan completar las tareas principales (ingreso, consulta de preguntas, consulta de perfil, realizar una pregunta) sin necesidad de tutoriales, dentro de los primeros 5 minutos de uso, sin requerir formación previa especializada. Este requerimiento surge del atributo de calidad Usabilidad/

Capacidad de Aprendizaje, considerando que el público objetivo incluye personas de edad avanzada con habilidades tecnológicas limitadas.

RNF6. Accesibilidad universal y soporte multi idioma

El producto debe ser utilizable por personas con distintas características demográficas y capacidades, incluyendo diferencias de edad, habilidades físicas y cognitivas, cultura, idioma, género y situación socioeconómica. Este requerimiento surge del atributo de calidad Usabilidad - Accesibilidad, ya que es fundamental para garantizar que la plataforma sea utilizable por personas con distintas características demográficas y capacidades. La interfaz debe cumplir con estándares internacionales de accesibilidad (WCAG 2.1 nivel AA) y soportar al menos 2 idiomas (Inglés y Español) con la integración necesaria para poder añadir más idiomas en el futuro.

RNF7. Protección de datos y comunicaciones seguras

Las comunicaciones entre cliente y servidor deben estar protegidas con HTTPS (TLS 1.2 o superior) y el sistema debe garantizar que los datos sensibles sean accesibles únicamente a usuarios y sistemas autorizados, mediante mecanismos de control de acceso y cifrado adecuados. Este requerimiento surge del atributo de calidad Seguridad/Confidencialidad, ya que es esencial para proteger la información sensible de los usuarios y garantizar comunicaciones seguras. Todos los accesos a datos sensibles deben pasar por un sistema de autenticación y autorización, y las contraseñas deben almacenarse utilizando algoritmos de hashing seguros.

RNF8. Intercambiabilidad de proveedores de servicios externos

El sistema debe estar diseñado para permitir la sustitución de proveedores de servicios externos con modificaciones mínimas en el código fuente. Este requerimiento surge del atributo de calidad Mantenibilidad/Modularidad, ya que permite la flexibilidad en la elección de proveedores sin afectar la arquitectura del sistema. Específicamente, debe ser posible cambiar de proveedor de base de datos vectorial, de proveedor de *embeddings* y de modelo de IA sin afectar el resto del código o flujo existente. El sistema debe mantener su funcionalidad completa independientemente del proveedor seleccionado, y la implementación de un nuevo proveedor debe requerir únicamente la creación de

adaptadores específicos que sigan interfaces predefinidas, sin modificar la lógica de negocio principal.

RNF9. Arquitectura modular con bajo acoplamiento

Cada módulo que compone el sistema debe estar diseñado de forma que su implementación y cambios internos no afecten el funcionamiento de otros módulos no relacionados. Este requerimiento surge de los atributos de calidad Mantenibilidad y Modularidad, ya que es fundamental para facilitar el mantenimiento y la evolución del sistema. Modificar o desplegar un módulo no debe requerir cambios en más del 5% de los demás módulos, y todos los módulos deben usar únicamente interfaces públicas documentadas para interactuar entre sí.

RNF10. Escalabilidad

El sistema debe ser escalable horizontal y verticalmente sobre infraestructura en la nube y debe ser capaz de manejar picos de tráfico mediante auto-scaling. Este requerimiento surge de los atributos de calidad Flexibilidad y Escalabilidad, ya que es esencial para permitir la expansión del sistema a nuevos dominios de conocimiento. La solución debe poder desplegarse en distintos entornos de ejecución con mínima configuración adicional, permitiendo la expansión a nuevos dominios de conocimiento sin requerir desarrollos adicionales complejos.

RNF11. Integridad y trazabilidad de modificaciones

Toda modificación de datos críticos debe requerir autenticación válida y validación de permisos. Este requerimiento surge del atributo de calidad Seguridad/Integridad y Responsabilidad, ya que es fundamental para garantizar la auditoría y trazabilidad de las acciones en el sistema. Debe existir un sistema de auditoría que registre todas las modificaciones a los datos, incluyendo usuario y fecha. El 100% de las modificaciones de datos críticos deben estar registradas y ser auditables.

RNF12. Segregación de datos entre comunidades

La información de cada comunidad debe mantenerse completamente aislada de manera que no exista posibilidad de acceso cruzado a datos de otra comunidad. Este requerimiento surge del atributo de calidad Seguridad/Confidencialidad, ya que es crítico

para garantizar el aislamiento de datos en una arquitectura *multi-tenant*. Cada comunidad debe funcionar como un silo de datos independiente, donde los documentos, preguntas, respuestas y configuraciones de una comunidad no deben ser accesibles ni visibles para usuarios de otras comunidades. El sistema debe garantizar que la base de datos vectorial, los *embeddings* y toda la información asociada a una comunidad específica permanezca segregada y no se comparta entre diferentes dominios de conocimiento.

5. Investigación del sistema de IA

Durante esta etapa, se llevó a cabo una investigación orientada a definir las decisiones tecnológicas fundamentales del proyecto. El objetivo principal fue establecer la arquitectura óptima para la solución de Inteligencia Artificial (IA) que permitiera construir un asistente especializado capaz de responder preguntas sobre el cuidado de bonsáis con precisión y confiabilidad.

Si bien el acceso a la información necesaria estaba garantizado, el desafío radicaba en determinar la mejor forma de integrarla y estructurarla dentro del sistema para optimizar la calidad de las respuestas. Asimismo, se debía asegurar que dichas respuestas fueran precisas y verificables, minimizando las alucinaciones comunes en los modelos de lenguaje cuando operan fuera de su conocimiento base. Finalmente, la solución debía diseñarse con una arquitectura escalable y extensible, de modo que pudiera adaptarse fácilmente a otros dominios de conocimiento más allá del caso de los bonsáis.

Para abordar estos desafíos, se implementaron Pruebas de Concepto (POCs) que evaluaron empíricamente las alternativas tecnológicas disponibles. Las decisiones abarcaron desde la infraestructura de IA (proveedor de LLM, lenguaje de programación, estrategia de incorporación de conocimiento) hasta aspectos arquitectónicos específicos (base de datos vectorial, enfoque de recuperación y generación, y mecanismo de generalización taxonómica). Cada POC documentó los resultados obtenidos, fundamentando las decisiones técnicas finales con evidencia empírica. Los detalles de estas investigaciones se presentan en las siguientes subsecciones.

5.1. Investigaciones iniciales

Durante los primeros meses del proyecto, a partir de noviembre de 2024, se desarrolló una serie de investigaciones exploratorias orientadas a definir el rumbo tecnológico de la solución. Estas actividades constituyeron la base conceptual y técnica sobre la cual se construyó el sistema de inteligencia artificial.

5.1.1. Proveedores de modelo de IA

Para la selección de un proveedor de LLM, se consideraron factores clave como rendimiento, costo, facilidad de uso y opciones de personalización. La evaluación se

realizó a fines de 2024, considerando los proveedores líderes disponibles: *OpenAI* y *Google AI*.

Google Gemini (de *Google AI*) se destaca por ser un modelo multimodal capaz de procesar texto, imágenes y código, con escalabilidad robusta respaldada por la infraestructura de Google. Sin embargo, al ser relativamente nuevo, su ecosistema de herramientas y documentación es menos maduro comparado con otras alternativas.

OpenAI ofrece modelos de vanguardia como GPT-3.5 y GPT-4, reconocidos por su capacidad excepcional para generar texto coherente. Su API es accesible y cuenta con una gran comunidad activa de desarrolladores, aunque sus costos pueden ser variables con el uso intensivo.

Atributo	OpenAI (ChatGPT)	Google Gemini
Modelos Disponibles	GPT-3.5, GPT-4, GPT-4 Turbo	<i>Gemini 1, Gemini 1.5, Gemini 2</i>
Capacidad Multimodal	Limitada a texto e imágenes en GPT-4	Soporte nativo para texto, imágenes y audio
Ventana de Contexto	Hasta 32,000 tokens con GPT-4 <i>Turbo</i>	Hasta 1 millón de <i>tokens</i> con <i>Gemini 2</i>
Integración	API flexible con SDKs para <i>Python, Node.js</i> , etc.	Integración profunda con el ecosistema de <i>Google</i>
Implementación RAG	Requiere herramientas externas como bases vectoriales	Capacidades nativas de RAG integradas
Documentación	Exhaustiva y con fuerte apoyo de la comunidad	Menos madura pero con soporte empresarial de <i>Google</i>
Personalización	<i>Fine-tuning</i> disponible en GPT-3.5, limitado en GPT-4	Probablemente soportado en versiones avanzadas

Tabla 2: Comparación entre OpenAI y Google Gemini

Se seleccionó a *OpenAI* como proveedor del modelo de IA debido a su madurez tecnológica, documentación completa y comunidad activa de desarrolladores. Su API simple y SDKs bien documentados para *Python* simplificaron la integración, mientras que su estructura de precios transparente basada en tokens permitió una estimación precisa de costos. Adicionalmente, los modelos GPT han demostrado rendimiento excepcional en generación de texto coherente y preciso.

Aunque *Google Gemini* presenta características prometedoras, la madurez del ecosistema de *OpenAI* y la facilidad de implementación fueron factores decisivos. Una ventaja crucial fue la mayor transparencia de *OpenAI* en el manejo de datos y privacidad, alineándose mejor con los requerimientos de protección de datos y el diseño *multi-tenant* del sistema.

El análisis detallado de esta investigación, incluyendo las pruebas realizadas y consideraciones adicionales, se encuentra documentado en el Anexo 13.6.1.

5.1.2. Javascript vs Python

La elección del lenguaje de programación es una decisión fundamental que impacta directamente en la eficiencia del desarrollo, la escalabilidad de la solución y en las capacidades técnicas del proyecto. Esta decisión adquirió particular relevancia dado que el sistema requería capacidades específicas de procesamiento de lenguaje natural, manejo de *embeddings* vectoriales, integración con APIs de LLMs y herramientas de evaluación y monitoreo del comportamiento del modelo.

Se realizó un análisis comparativo entre *Python* y *JavaScript* considerando su madurez, ecosistema y nivel de soporte en el ámbito de la inteligencia artificial. *Python* se destaca como el lenguaje dominante en ciencia de datos y aprendizaje automático, gracias a su sintaxis simple, amplia comunidad y ecosistema especializado que facilita la experimentación y el prototipado rápido de modelos.

Por su parte, *JavaScript* ha extendido su alcance hacia el aprendizaje automático mediante frameworks como *TensorFlow.js*; sin embargo, su infraestructura sigue siendo más limitada para tareas de procesamiento intensivo o integraciones con modelos avanzados.

Un factor decisivo fue que *Opik*, plataforma de evaluación y monitoreo de LLMs utilizada en el proyecto, ofrece SDK únicamente para *Python*. En consecuencia, se determinó que

Python era la opción más adecuada para este proyecto, gracias a su ecosistema robusto en ciencia de datos, herramientas avanzadas como *Opik*, y una comunidad activa que asegura soporte y recursos actualizados.

El análisis detallado de ambos lenguajes, incluyendo comparativas de rendimiento, ecosistema de librerías y capacidades para tareas de inteligencia artificial, se presenta en el Anexo 13.6.2.

5.1.3. RAG vs. *Fine Tuning*

Una vez definido el proveedor de LLM y el lenguaje de programación, surgía la pregunta fundamental sobre cómo dotar al modelo de conocimiento especializado sobre el cuidado de bonsáis. Los LLMs generales tienden a proporcionar respuestas genéricas y ocasionalmente incorrectas cuando se les consulta sobre dominios altamente especializados. El sistema requería que el asistente respondiera con información específica sobre especies particulares, técnicas de cuidado adaptadas a contextos geográficos y recomendaciones precisas basadas en conocimiento experto.

Para dotar al LLM de conocimiento específico y actualizado, se evaluaron dos técnicas principales: *fine-tuning* y *RAG*.

El *fine-tuning* consiste en reentrenar el modelo de lenguaje con un conjunto de datos específico del dominio, ajustando sus pesos para que internalice el conocimiento especializado. Sin embargo, este enfoque presentaba limitaciones significativas para los requerimientos del proyecto.

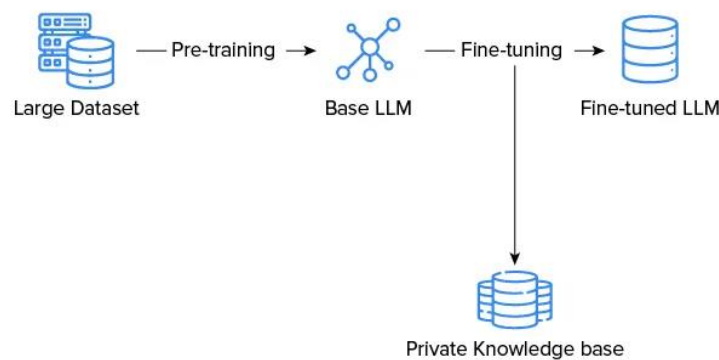


Ilustración 26: Diagrama de flujo de proceso de *Fine-Tuning*. Tomado de [30]

En primer lugar, la falta de flexibilidad representaba un problema crítico. El conocimiento sobre bonsáis evoluciona constantemente: se incorporan nuevas especies, se descubren técnicas mejoradas y se actualizan recomendaciones basadas en experiencias de la comunidad. Con *Fine-tuning*, cada actualización del conocimiento requeriría un nuevo proceso de reentrenamiento del modelo, lo que resulta costoso en términos de tiempo y recursos computacionales. Esta rigidez no se alineaba con la visión de un sistema dinámico que pudiera crecer y mejorar continuamente.

El costo elevado constituía otra limitación importante. El *Fine-tuning* demanda infraestructura de alto rendimiento (GPUs especializadas) y tiempo considerable de procesamiento. Para un proyecto con recursos limitados y con la necesidad de iterar rápidamente, estos costos representaban una barrera significativa.

Adicionalmente, el *fine-tuning* adolece de falta de transparencia. Cuando el modelo internaliza el conocimiento, resulta difícil rastrear el origen de la información en cada respuesta. En este caso, donde la confiabilidad era fundamental, la capacidad de citar fuentes y verificar la procedencia de cada recomendación era un requerimiento esencial.

Por otra parte, la técnica de *Retrieval-Augmented Generation* (RAG) ofrece un enfoque diferente. En lugar de modificar el modelo, RAG mantiene el conocimiento especializado en una base de datos externa y lo recupera dinámicamente según la consulta del usuario. El modelo de lenguaje recibe tanto la pregunta como los documentos recuperados, utilizándolos como contexto para generar una respuesta fundamentada.

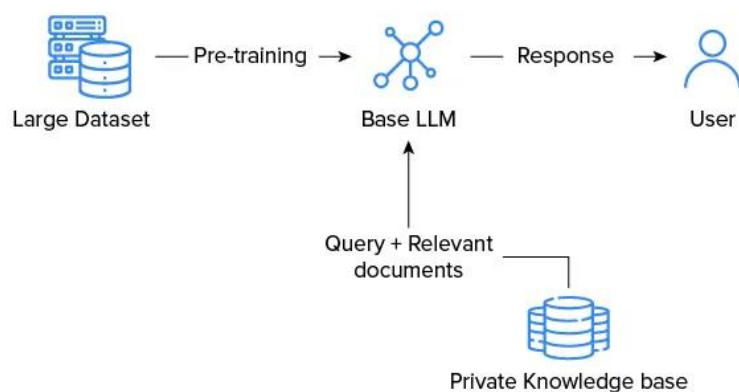


Ilustración 27: Diagrama de flujo de proceso RAG. Tomado de [30]

Este enfoque presentaba ventajas significativas para los requerimientos del proyecto. La flexibilidad para manejar datos dinámicos en tiempo real permitía actualizar el conocimiento simplemente agregando nuevos documentos a la base de datos, sin necesidad de reentrenar ningún modelo. La incorporación de nueva información sobre especies, técnicas o recomendaciones podía realizarse de forma inmediata y con costos mínimos.

La transparencia que ofrece RAG resultaba particularmente valiosa. Cada respuesta generada puede acompañarse de referencias a los documentos específicos utilizados, permitiendo a los usuarios verificar la información y confiar en las recomendaciones proporcionadas. Esta trazabilidad era fundamental para construir credibilidad en un dominio donde información incorrecta podía resultar en daños a los árboles.

La escalabilidad de RAG se alineaba con la visión de extender el sistema a múltiples nichos de conocimiento. El diseño *multi-tenant* permite que diferentes comunidades (bonsái, apicultura, astronomía amateur) mantengan sus propias bases de conocimiento aisladas, compartiendo la misma infraestructura de IA pero accediendo únicamente a su información específica.

Se optó por implementar RAG como estrategia principal del proyecto debido a su flexibilidad, transparencia, escalabilidad y menor costo operacional. Esta decisión técnica permitió construir un sistema que no solo resuelve el problema actual sino que establece una base sólida para evolución y expansión futuras.

El documento de investigación completo generado para esta sección se puede encontrar en el Anexo 13.6.3 y 13.6.4.

5.1.4. Proveedor de base de datos

Una vez confirmada la adopción de RAG como estrategia principal, surgía una necesidad técnica inmediata: seleccionar una solución de almacenamiento y recuperación de información que soportara búsquedas semánticas eficientes. El sistema de RAG funciona convirtiendo documentos y consultas en representaciones vectoriales (*embeddings*) y buscando similitud matemática entre ellos. Esta operación requiere una base de datos especializada capaz de manejar vectores de alta dimensionalidad y realizar búsquedas de similitud a gran velocidad.

Los requerimientos del producto establecían restricciones específicas para esta selección. El sistema debía manejar documentos sobre múltiples especies de árboles, organizados en una estructura *multi-tenant* donde cada comunidad accede únicamente a su información aislada. Las consultas de los usuarios debían responderse en tiempo real con baja latencia, lo que exigía un proveedor con rendimiento optimizado. Adicionalmente, la arquitectura debía soportar filtrado por metadatos taxonómicos (especie, género, familia, orden) para implementar el mecanismo de generalización jerárquica planificado.

Se evaluaron cuatro opciones principales de bases de datos vectoriales disponibles en el mercado durante 2024. *Milvus* se presenta como una solución de código abierto con alta escalabilidad e integración con *frameworks* de aprendizaje automático como *TensorFlow* y *PyTorch*. *Weaviate*, también de código abierto, ofrece una API basada en *GraphQL* y capacidades nativas de búsqueda semántica. *Qdrant* se destaca por su optimización para búsquedas en tiempo real y soporte para búsquedas híbridas que combinan similaridad vectorial con filtros tradicionales. Finalmente, *Pinecone* se posiciona como una solución completamente gestionada que prioriza el alto rendimiento y la facilidad de uso, aunque con un costo mayor y como plataforma de código cerrado.

La decisión de seleccionar *Pinecone* se fundamentó en varios factores clave. Su naturaleza de servicio gestionado eliminaba la necesidad de configurar, mantener y escalar infraestructura de base de datos, permitiendo al equipo concentrarse en el desarrollo de la lógica de negocio del sistema. El rendimiento optimizado de *Pinecone* para búsquedas de gran escala con baja latencia se alineaba directamente con los requerimientos de respuesta en tiempo real del producto. La integración simplificada con ecosistemas de RAG, mediante SDKs bien documentados para *Python*, aceleraba significativamente el proceso de desarrollo.

Las capacidades de filtrado por metadatos de *Pinecone* eran fundamentales para implementar el mecanismo de generalización taxonómica, permitiendo búsquedas progresivas por especie, género, familia y orden.

Aunque *Pinecone* presenta costos operacionales más elevados que alternativas de código abierto y ofrece menor personalización al ser una plataforma cerrada, se consideró que estos compromisos eran aceptables dado el contexto del proyecto. Esta decisión garantizó

una implementación más rápida y robusta del sistema, minimizando riesgos técnicos asociados con la configuración y mantenimiento de infraestructura de base de datos.

El documento de investigación completo generado para esta sección se puede encontrar en el Anexo 13.6.5 y 13.6.6.

5.2. Investigaciones posteriores

Una vez definida la arquitectura central, la investigación continuó de forma iterativa a lo largo del desarrollo del proyecto. En esta etapa, el foco se desplazó hacia la experimentación con componentes específicos y la resolución de nuevos desafíos surgidos durante la implementación.

5.2.1. RAG vs Agentic RAG

Con la infraestructura base definida (proveedor de LLM, lenguaje de programación, estrategia RAG y base de datos vectorial), surgía una pregunta arquitectural fundamental: ¿cómo debía organizarse el proceso de recuperación y generación de respuestas? En el desarrollo de este sistema de consulta de información sobre árboles, se exploraron dos enfoques arquitecturales distintos: RAG tradicional y *Agentic RAG*. La decisión de implementar ambos responde a la necesidad de ofrecer al cliente opciones de experimentación y evaluar empíricamente cuál resulta más efectivo en este dominio específico.

5.2.1.1. RAG Tradicional con Grafo

El enfoque tradicional se construyó como un flujo jerárquico y determinista que refleja la clasificación taxonómica de los árboles (especie → género → familia → orden → general). El sistema sigue una secuencia predefinida de pasos, avanzando progresivamente hacia niveles más generales cuando no encuentra información suficientemente específica.

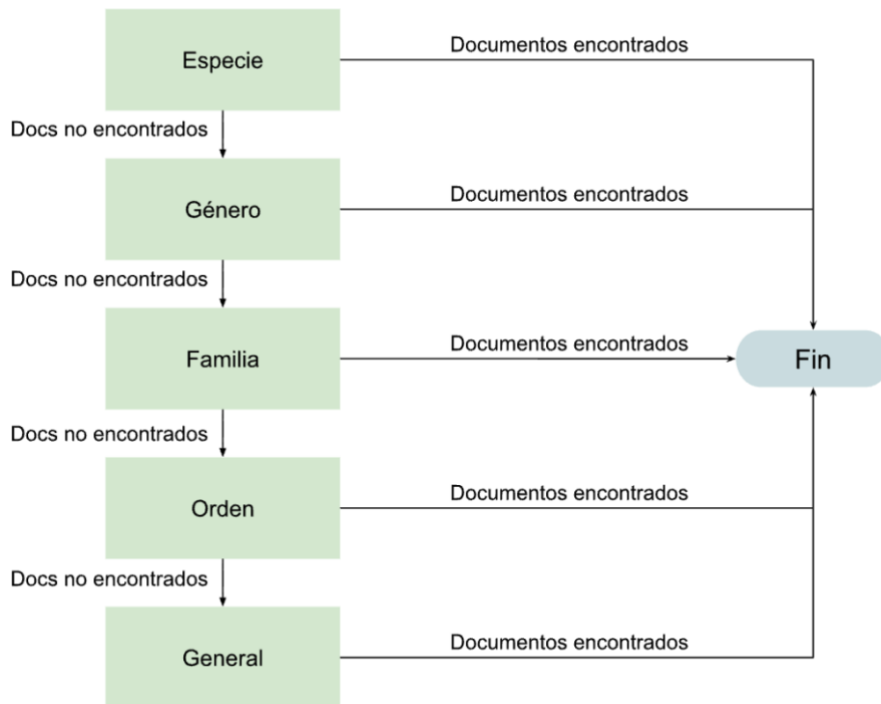


Ilustración 28: Flujo jerárquico del RAG tradicional

Este modelo ofrece múltiples ventajas que justifican su elección como punto de partida. En primer lugar, la jerarquía taxonómica se alinea naturalmente con la lógica del dominio, lo que permite que el proceso de recuperación siga el mismo razonamiento que emplearía un experto humano al identificar un árbol. Además, su naturaleza determinista garantiza que la misma consulta recorra siempre el mismo camino, aportando reproducibilidad y facilidad para la depuración.

Desde el punto de vista operativo, resulta también un enfoque eficiente, ya que reduce el número de llamadas al modelo al priorizar la lógica del flujo antes que la inferencia.

Finalmente, el esquema jerárquico favorece la transparencia del sistema, permitiendo indicar explícitamente el nivel de detalle en el que se obtuvo la respuesta (especie, género, familia u orden).

5.2.1.2. Agentic RAG

En contraste, el enfoque agéntico introduce un componente de razonamiento autónomo. En este caso, el control de la búsqueda recae en un agente que decide dinámicamente qué herramientas utilizar, en qué orden y cuántas veces invocarlas, siguiendo un proceso iterativo de razonamiento (*ReAct*).

Este paradigma aporta una flexibilidad superior, ya que el agente puede adaptarse a consultas menos estructuradas o incluso ambiguas, explorando distintos caminos hasta hallar la información más pertinente. Además, su capacidad de razonamiento iterativo le permite reconsiderar resultados parciales y ajustar la estrategia antes de emitir una respuesta final.

No obstante, estas ventajas conllevan ciertos compromisos. El sistema presenta menor determinismo, dado que una misma consulta podría seguir rutas de razonamiento diferentes en ejecuciones sucesivas. También implica mayor latencia y costo computacional, al requerir múltiples llamadas adicionales al modelo durante el proceso de razonamiento. Por último, su menor transparencia dificulta la trazabilidad interna y la depuración del comportamiento del agente.

5.2.1.3. Comparación

Aspecto	RAG Tradicional (Grafo)	Agentic RAG
Determinismo	Alto y reproducible	Bajo, resultados variables
Transparencia	Alta, cada paso es explícito	Media, razonamiento opaco
Eficiencia	Alta, pocas llamadas al modelo	Baja, más llamadas intermedias
Alineación con el dominio	Perfecta con la jerarquía taxonómica	Depende de la calidad del <i>prompt</i>
Facilidad de mantenimiento	Alta, flujo claro y modificable	Media, ajustes complejos en el <i>prompt</i>

Tabla 3: Comparación RAG tradicional vs Agentic RAG

Para un dominio como la clasificación de árboles, donde la información se organiza de forma jerárquica y bien definida, el RAG tradicional con grafo es la opción más adecuada. Ofrece reproducibilidad, menor costo y mayor transparencia en las respuestas. Esto lo hace especialmente apropiado cuando la confiabilidad y la consistencia son fundamentales.

El enfoque agéntico, aunque menos eficiente, se mantiene dentro del sistema como una alternativa experimental. Puede resultar útil en otros nichos donde no se tiene una taxonomía tan estricta y en casos de borde más complejos, cuando la información es ambigua o incompleta, y como base para futuras investigaciones sobre agentes más sofisticados.

5.2.2. Generalización

Uno de los principales desafíos del proyecto fue cómo ofrecer respuestas pertinentes en ausencia de información específica sobre una determinada especie. La solución se apoyó en un principio botánico esencial, según el cual las especies taxonómicamente relacionadas comparten características relevantes, lo que permite inferir datos válidos a partir de niveles taxonómicos superiores.

Fundamento Botánico

El conocimiento botánico se organiza en una jerarquía taxonómica donde especies del mismo género comparten características morfológicas, requerimientos de cultivo y patrones de crecimiento similares. Esta similitud disminuye gradualmente en niveles superiores de la jerarquía.

Una familia agrupa varios géneros relacionados que comparten características estructurales y fisiológicas generales, mientras que el orden constituye un nivel aún más amplio, agrupando familias que comparten rasgos evolutivos y morfológicos fundamentales. Este principio de similitud decreciente permite inferir información útil sobre especies con datos limitados a partir del conocimiento disponible en sus grupos taxonómicos superiores.

Estrategia de Búsqueda Jerárquica

El sistema implementa una estrategia de búsqueda jerárquica estructurada en cinco niveles de especificidad, organizados de mayor a menor precisión.

- En el nivel de **especie**, la búsqueda se realiza utilizando el nombre científico completo del ejemplar.
- En el nivel de **género**, se restringe al primer componente del nombre, correspondiente al género botánico.

- A continuación, el **nivel de familia** agrupa los géneros relacionados, mientras que el **nivel de orden** amplía la búsqueda a familias afines dentro del mismo orden taxonómico.
- Finalmente, el **nivel general** elimina las restricciones taxonómicas y aplica principios botánicos comunes a cualquier árbol, permitiendo recuperar información relevante incluso cuando no existen registros específicos sobre la especie consultada.

Esta jerarquía refleja fielmente la clasificación taxonómica utilizada universalmente en botánica y permite realizar un “préstamo de conocimiento” entre especies emparentadas, aprovechando la estructura natural del dominio para maximizar el valor informativo disponible en el sistema.

Principio de Transparencia

Un aspecto crítico del diseño es la transparencia en la comunicación del nivel de generalización alcanzado. Cuando se responde con información de un nivel superior al de especie, el sistema debe indicar explícitamente esta situación al usuario. Por ejemplo: "No tengo información específica sobre [especie consultada], pero puedo ayudarte con información sobre el género [género correspondiente]...". Esta honestidad permite al usuario calibrar apropiadamente su confianza en la información y tomar decisiones informadas sobre cómo aplicarla a su situación específica.

Manejo de Información Incompleta

El mecanismo debe ser robusto ante información taxonómica incompleta o inconsistente. Si el sistema no puede determinar el género de una especie, o la familia de un género, no debe fallar catastróficamente sino simplemente saltar ese nivel en la jerarquía, continuando con el siguiente nivel disponible.

Aplicabilidad a otros nichos

Si bien el mecanismo de generalización jerárquica fue diseñado específicamente para el dominio botánico, el principio subyacente es extensible a cualquier dominio de conocimiento que presente una estructura jerárquica natural con niveles de especificidad decreciente.

El sistema se fundamenta en dos premisas generalizables: primero, que entidades dentro de un mismo nivel jerárquico comparten características comunes; segundo, que cuando no existe información específica sobre una entidad particular, es posible inferir información útil desde niveles superiores de abstracción, comunicando explícitamente al usuario el nivel de generalización aplicado.

Este enfoque podría aplicarse a diversos dominios:

- **Organizacional:** Empresa → Departamento → Equipo → Profesional individual
- **Geográfico:** País → Región → Ciudad → Barrio
- **Tecnológico:** Categoría de producto → Familia → Modelo específico → Versión
- **Normativo:** Marco regulatorio general → Legislación sectorial → Norma específica → Artículo particular

En cada caso, la estrategia consistiría en buscar primero información específica al nivel más bajo de la jerarquía y, ante su ausencia, ascender progresivamente hacia niveles más generales, manteniendo siempre la transparencia sobre el nivel de especificidad de la información proporcionada. La robustez del sistema ante información incompleta y su capacidad de "préstamo" de conocimiento entre entidades relacionadas son características transferibles a cualquier dominio estructurado jerárquicamente.

6. Arquitectura e implementación del proyecto

En este capítulo se presenta una descripción general de la arquitectura y la implementación de la solución propuesta, abarcando tanto los componentes de software como la infraestructura que la soporta. Se detalla la integración de modelos de lenguaje a gran escala (LLMs) para la generación de respuestas inteligentes, explicando los criterios de diseño adoptados, los desafíos técnicos enfrentados y los factores de calidad considerados durante el desarrollo.

Asimismo, se analizan las decisiones tecnológicas que guiaron la construcción del sistema y se describe el sistema de IA generado, incluyendo el diseño e implementación de los mecanismos RAG y *Agentic RAG*, los procesos de observabilidad y evaluación de los modelos, y las métricas empleadas para garantizar su desempeño.

6.1. Desafíos para lograr la solución

¿Cómo lograr que un modelo de lenguaje a gran escala funcione de manera eficiente, segura y escalable dentro de un ecosistema web? Esta fue la pregunta que marcó los principales desafíos técnicos y conceptuales enfrentados durante el proyecto.

Integración entre componentes

La interoperabilidad entre componentes desarrollados en distintos lenguajes y entornos de ejecución representó un aspecto clave durante la implementación de la arquitectura. Esta separación tecnológica se adoptó por motivos de especialización, pero exigió diseñar interfaces de comunicación claras mediante *API REST* que garantizara la consistencia y el rendimiento del sistema.

Seguridad y manejo de información sensible

La interacción con servicios externos, como *OpenAI*, *Pinecone* y el que la infraestructura fuese desplegada en *AWS* y *Vercel*, requirió implementar mecanismos de autenticación robustos y un manejo cuidadoso de credenciales y claves de las distintas APIs. Para abordar estos desafíos, se implementaron buenas prácticas de seguridad orientadas a proteger la integridad y confidencialidad del sistema. En particular, se fortaleció la configuración del entorno de ejecución y se aseguró la transmisión de datos mediante el

uso de HTTPS y un control de acceso basado en roles, lo que garantizó una comunicación segura y una correcta gestión de los permisos de los distintos usuarios y servicios.

Observabilidad y evaluación de modelos

Otro desafío relevante fue la incorporación de mecanismos de monitoreo y evaluación para los LLMs, con el fin de contar con visibilidad sobre su comportamiento y desempeño. Para dar respuesta a esta necesidad, se integró la plataforma *Opik*, que permitió registrar métricas de rendimiento, analizar los prompts y las respuestas generadas, y detectar posibles desviaciones o errores en los procesos de inferencia. Esta implementación resultó fundamental para garantizar la confiabilidad del sistema y favorecer la mejora continua de los flujos de generación de respuestas.

Escalabilidad y mantenimiento

Finalmente, se identificó como desafío la necesidad de que la arquitectura pudiera escalar de forma independiente cada uno de sus componentes, garantizando un desempeño estable frente a variaciones en la demanda y evitando cuellos de botella. Para abordar este requerimiento, se adoptó un diseño modular con despliegues en AWS y *Vercel*, que permite escalar horizontalmente los distintos servicios, incluyendo el *frontend*, el *backend*, el servicio RAG y las bases de datos de acuerdo con la carga del sistema. En este esquema, AWS proporciona la infraestructura base y la gestión dinámica de recursos, mientras que *Vercel* facilita la distribución y escalabilidad automática del *frontend*, optimizando el rendimiento global y la experiencia del usuario. Esta combinación de tecnologías permitió mejorar la resiliencia, la eficiencia operativa y la mantenibilidad del sistema a largo plazo.

6.2. Descripción de la arquitectura

Una vez definidos los requerimientos y los principales atributos de calidad, se procedió al diseño de la arquitectura del sistema. En la siguiente figura se presenta un diagrama que sintetiza su estructura general.

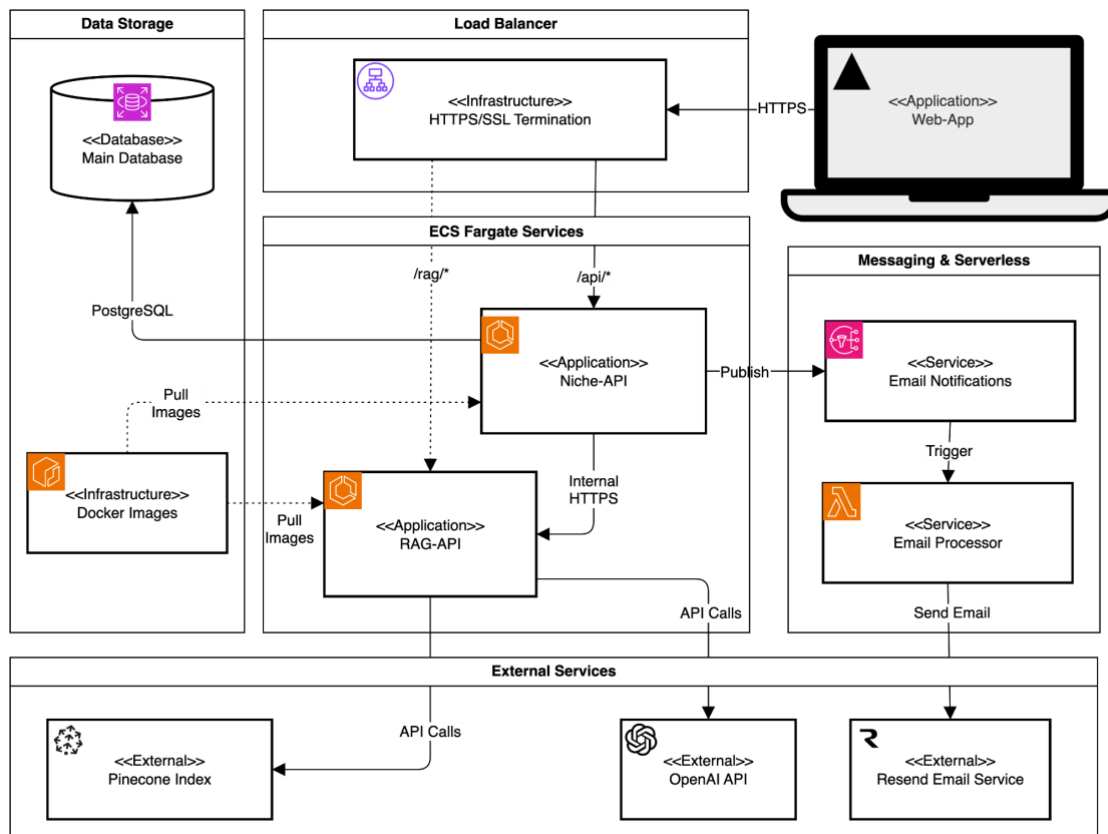


Ilustración 29: Diagrama de Arquitectura de Alto Nivel. Elaborado a partir de [31].

Niche-AI Web App

NicheAI Web-App constituye la interfaz principal de interacción del usuario con el sistema. Su objetivo es brindar una experiencia fluida e intuitiva para explorar contenidos, formular consultas y acceder a respuestas especializadas generadas por IA en las distintas comunidades.

La aplicación está desarrollada con *Next.js*, sobre *React* y *TypeScript*, e implementa un diseño moderno y adaptable mediante *Tailwind CSS*. Su despliegue en *Vercel* permite

escalado automático, distribución global a través de *content delivery network* (CDN) e integración continua mediante *GitHub Actions* (CI/CD).

App Load Balancer

El componente *App Load Balancer*, implementado mediante un *Application Load Balancer* (ALB) de AWS, constituye el punto de entrada principal al sistema del *backend*. Su rol es recibir todo el tráfico HTTP/HTTPS proveniente de las aplicaciones web y distribuirlo de forma eficiente hacia los servicios *backend* desplegados en ECS (*Amazon Elastic Container Service*) *Fargate*. Siendo ECS un servicio de orquestación de contenedores que ejecuta tareas/servicios en *Fargate*.

El ALB también se encarga de manejar la terminación SSL/TLS, gestionando los certificados digitales y descargando la carga de cifrado de los servicios internos. De esta forma, actúa como *gateway* de tráfico seguro, garantizando el enrutamiento correcto y la disponibilidad del sistema.

Niche-API

El componente Niche-API constituye el núcleo lógico y funcional del sistema. Se trata de una *API REST multi-tenant*, desarrollada en *Node.js* y *TypeScript*, que concentra la lógica de negocio principal, incluyendo la gestión de autenticación, administración de usuarios y comunidad, moderación de contenido y la integración con servicios externos de inteligencia artificial y mensajería.

La API se ejecuta en *AWS ECS Fargate*, dentro de un entorno contenedorizado, escalable y administrado automáticamente, donde expone sus funcionalidades a través del endpoint */api/*. Su arquitectura modular adopta los principios de *Clean Architecture*, lo que favorece la mantenibilidad, la facilidad de prueba y la separación de responsabilidades, asegurando una base sólida para la evolución futura del sistema.

RAG-API

El RAG-API constituye un componente central dentro del sistema de consultas inteligentes sobre conocimiento de nicho. Desplegado como servicio independiente en *AWS ECS Fargate*, este servicio implementa el patrón *Retrieval-Augmented Generation* (RAG) para proporcionar respuestas contextualizadas y precisas. Desarrollado en *Python*

utilizando *FastAPI*, el RAG-API se encarga de recuperar información relevante desde un índice vectorial y procesar las consultas de manera que puedan ser enriquecidas y respondidas con alto nivel de detalle.

Su arquitectura modular facilita la integración y comunicación eficiente con otros componentes del sistema, al tiempo que garantiza escalabilidad, alta disponibilidad y observabilidad de las operaciones. Además, el uso de modelos de lenguaje avanzados permite generar respuestas coherentes y contextualizadas, asegurando que el sistema mantenga un desempeño confiable y consistente frente a distintos tipos de consultas.

Main Database

La base de datos principal actúa como el núcleo relacional del sistema, sirviendo como punto central para el almacenamiento persistente y consistente de la información. El mismo se implementó sobre una instancia *PostgreSQL* 17.4 desplegada en el Servicio gestionado de bases de datos relacionales AWS RDS (*Amazon Relational Database Service*), con una configuración *db.t3.micro* que ofrece un equilibrio adecuado entre rendimiento y costo. En ella se resguarda la información crítica de la aplicación, como usuarios, configuraciones, metadatos y demás datos estructurados, bajo las garantías transaccionales propias del modelo ACID.

La elección de *PostgreSQL* respondió tanto a consideraciones técnicas como de experiencia del equipo. Su soporte nativo para datos estructurados y semiestructurados (JSON) resultó especialmente valioso para gestionar los elementos dinámicos asociados a los distintos tenants, permitiendo una mayor flexibilidad sin renunciar a la solidez del modelo relacional. Este enfoque híbrido facilitó la representación de datos variables dentro de un marco estructurado y coherente, aprovechando además la familiaridad del equipo con entornos relacionales para simplificar la operación y el mantenimiento del sistema.

Asimismo, la capacidad de aislar y segmentar la información entre tenants se alinea directamente con los requisitos no funcionales definidos en el diseño, asegurando la segregación lógica de datos y evitando posibles interferencias o cruces de información entre dominios independientes.

Finalmente, el uso de AWS RDS agrega una capa adicional de confiabilidad y seguridad al proporcionar escalabilidad elástica, copias de seguridad automáticas y aislamiento mediante VPC, características que consolidan la robustez operativa de la solución.

Docker Images

El *Elastic Container Registry* (ECR) de AWS funciona como repositorio centralizado de las imágenes *Docker* de la aplicación. Contiene los principales servicios *nicheai-node-api* y *nicheai-python-api*, gestionando versiones mediante etiquetas semánticas y políticas de ciclo de vida que optimizan la administración automática de imágenes. Este componente es esencial para mantener un flujo de despliegue consistente y seguro.

Los servicios desplegados en ECS Fargate realizan *pulls* de las imágenes desde ECR durante el despliegue, mientras que los desarrolladores incorporan sus cambios a nivel de código mediante pipelines de CI/CD configurado en *GitHub Actions*, garantizando un ciclo de actualización ágil y controlado.

La decisión de utilizar ECR se basa en su integración nativa con el ecosistema AWS, su seguridad incorporada, incluyendo escaneo automático de vulnerabilidades y encriptación de datos en reposo, y su eficiencia en la gestión de versiones mediante etiquetado semántico. Además, ofrece una distribución optimizada a nivel global y una administración de recursos altamente eficiente, ya que implementa deduplicación de capas de contenedores, lo que evita el almacenamiento redundante de datos comunes entre imágenes *Docker* y contribuye a reducir significativamente los costos operativos.

Email Notifications Topic

El uso de *SNS Topic* se debe a la necesidad del sistema de distribuir notificaciones a sus usuarios. Este sistema de mensajería utiliza el patrón *publish and subscribe* para el envío de notificaciones mediante correo electrónico, de manera asíncrona.

La integración permite que el componente Niche-API publique eventos de notificación en el *SNS Topic*, mientras que la función *Lambda* suscrita procesa dichos eventos y envía los correos electrónicos de forma asíncrona, sin bloquear la lógica de negocio. Los permisos de publicación se gestionan mediante documentos JSON que definen permisos

asociados a identidades o recursos que determinan qué acciones están permitidas o denegadas (políticas IAM), garantizando así un control de acceso granular.

Se tomó la decisión de utilizar SNS debido a su capacidad de desacoplamiento, su escalabilidad ante posibles picos de tráfico y sus mecanismos de fiabilidad como su confiabilidad con reintentos automáticos y colas de mensajes fallidos. Además, su modelo de pago por uso optimiza los costos operativos al evitar la necesidad de mantener infraestructura dedicada y facilita la extensión del sistema a nuevos canales de notificación sin la necesidad de modificar el núcleo de la API.

Email Processor Lambda

La función *Lambda Email Processor* procesa los mensajes publicados en el *SNS Email Notifications Topic* y se encarga del envío efectivo de correos electrónicos mediante la API de *Resend*. Esta herramienta permite despachar mensajes mediante su API, sin tener que gestionar servidores SMTP propios. En la siguiente sección se profundizará sobre este componente. Cabe destacar, que debido a la forma en la que fue integrado a la solución, el intercambio de esta herramienta por otra con una funcionalidad similar, no presentaría mayores dificultades.

Esta función incluye un *Lambda Layer* con las dependencias necesarias y centraliza la lógica de envío, la gestión de errores y el registro detallado de eventos para fines de monitoreo y trazabilidad.

Para su operación, la función *Lambda* se activa por *triggers* provenientes de SNS, obtiene las credenciales de envío desde *AWS Secrets Manager* y realiza las llamadas correspondientes a la API de *Resend* para el despacho de los mensajes. Paralelamente, publica logs y métricas en *CloudWatch*, lo que permite supervisar tareas de entrega de notificaciones, latencia y errores, facilitando la observabilidad del componente y de su desempeño.

La decisión de utilizar *Lambda* responde a sus características *serverless*, escalado automático ante variaciones de carga, integración nativa con sistemas basados en eventos y modelo de facturación por ejecución, lo que evita costos por infraestructura inactiva y facilita la operación en picos de demanda.

Resend Email Service

Resend Email Service es un proveedor externo de correo electrónico transaccional responsable del envío de todas las notificaciones de la aplicación. Este servicio garantiza una alta tasa de entregabilidad y proporciona métricas detalladas sobre la interacción de los usuarios, como aperturas y clics.

La función *Lambda* se comunica con *Resend* para gestionar el envío de correos electrónicos, obteniendo las credenciales de autenticación desde *AWS Secrets Manager*, al igual que la función *Lambda*.

La elección para el uso de *Resend* se basó en su facilidad de integración, elevada *deliverability*, su capacidad analítica avanzada (en especial el análisis sobre apertura y clics por parte de los usuarios) y la confiabilidad de infraestructura, respaldada por un *Service Level Agreement (SLA)* garantizado.

Pinecone Index

Pinecone Index es un servicio externo de base de datos vectorial utilizado para almacenar los *embeddings* generados por el sistema, con el fin de habilitar funcionalidades de RAG. Este servicio permite realizar búsquedas semánticas de manera eficiente y escalable sobre grandes volúmenes de información, facilitando la recuperación de contenido relevante en función de la similitud en su significado.

El componente se integra con la RAG API mediante llamadas a la API de *Pinecone* para la inserción y consulta de vectores, lo que permite mantener actualizada la base de conocimiento mediante inserciones y así optimizar el rendimiento de las búsquedas contextuales dentro de los vectores.

OpenAI API

OpenAI API se integra en el sistema para proporcionar capacidades avanzadas de procesamiento de lenguaje natural (NLP), generación de texto y creación de *embeddings*. Constituye el núcleo del motor de inteligencia artificial que posibilita tanto la generación de respuestas contextuales, como la representación semántica de documentos utilizada en la funcionalidad de RAG.

La RAG API se comunica con la API de *OpenAI* para realizar solicitudes de generación de texto y *embeddings*, utilizando credenciales seguras almacenadas en *AWS Secrets Manager* para su autenticación.

Sprig API

Sprig se integra en la arquitectura de NicheAI como un componente especializado en la recolección y análisis de métricas de interacción y experiencia de usuario. Su incorporación responde a la necesidad de obtener retroalimentación cualitativa y contextual directamente desde la interfaz, permitiendo evaluar de forma continua la usabilidad, satisfacción y efectividad de las respuestas generadas por la inteligencia artificial. A diferencia de los sistemas tradicionales de valoración binaria, *Sprig* ofrece una instrumentación más granular mediante *eventos* y encuestas dinámicas, lo que posibilita una observación detallada del comportamiento del usuario y de los puntos críticos en su flujo de interacción.

6.3. Atributos de calidad

El diseño de la arquitectura de software estuvo fuertemente determinado por los atributos de calidad definidos en las etapas iniciales del proyecto, que establecieron los lineamientos técnicos y las decisiones estructurales adoptadas. En particular, los factores más influyentes fueron la mantenibilidad, la seguridad, la usabilidad y la flexibilidad.

Mantenibilidad

En términos de mantenibilidad, se priorizó una arquitectura modular y desacoplada, basada en componentes independientes con responsabilidades claramente definidas. Este enfoque permitió minimizar el impacto de los cambios, facilitar la extensión del sistema y habilitar el desarrollo paralelo entre los distintos equipos.

Por otro lado, en la infraestructura se adoptó el uso de *Infrastructure as Code* (IaC) mediante *Terraform*, lo que contribuye directamente a la mantenibilidad al versionar toda la configuración del entorno en código, eliminando configuraciones manuales propensas a errores y facilitando el seguimiento de cambios, *rollbacks* y la replicación exacta de ambientes. La estrategia de contenerización con *Docker* complementa esta mantenibilidad al encapsular cada servicio con todas sus dependencias en imágenes

inmutables, eliminando inconsistencias entre ambientes de desarrollo y producción. Esta combinación reduce drásticamente el tiempo dedicado a tareas operativas, permitiendo que el equipo se enfoque en la evolución funcional del sistema.

Disponibilidad

Desde las primeras etapas se buscó asegurar la continuidad operativa del sistema ante posibles fallos de componentes individuales, adoptando un enfoque tolerante a errores y escalable horizontalmente. Para ello, los servicios principales se diseñaron como componentes *stateless*, lo que permite replicarlos y distribuir la carga de manera equilibrada mediante un balanceador de tráfico. Esta decisión, junto con la incorporación de mecanismos de monitoreo y reintento automático ante fallas transitorias, permitió definir una arquitectura capaz de mantener su funcionamiento incluso frente a interrupciones parciales.

En cuanto a la infraestructura, se implementó un *Application Load Balancer* (ALB) que distribuye el tráfico entre múltiples instancias y realiza *health checks* automáticos, redirigiendo las peticiones únicamente hacia contenedores saludables. Adicionalmente, se configuraron políticas de *auto-scaling* que permiten ajustar dinámicamente la cantidad de instancias según la demanda, asegurando disponibilidad incluso durante picos de tráfico inesperados.

Seguridad

Se implementó un sistema de gestión centralizada de identidades mediante el patrón *Identity Provider*, donde el servicio de autenticación actúa como autoridad central encargada de validar credenciales, generar *tokens JWT* y gestionar sesiones de usuario. Este enfoque permite identificar de manera única cada acción realizada en el sistema y habilita un modelo de control de acceso basado en roles y permisos, asegurando que cada usuario tenga acceso únicamente a los recursos autorizados según su nivel de privilegios dentro de su comunidad correspondiente.

Para proteger las comunicaciones, se estableció como requisito obligatorio el uso de HTTPS con TLS 1.2 o superior en todas las interacciones entre cliente y servidor, garantizando el cifrado de datos en tránsito. Las contraseñas se almacenan utilizando algoritmos de hashing seguros como *bcrypt*, evitando su exposición incluso ante

eventuales brechas de seguridad. Además, se incorporó un sistema de auditoría que registra todas las modificaciones a datos críticos, incluyendo el usuario responsable y la marca temporal, lo que permite trazabilidad completa de las operaciones sensibles.

En el componente RAG, se implementó aislamiento estricto entre comunidad mediante el uso de *namespaces* diferenciados en la base de datos vectorial, garantizando que las consultas de una comunidad no puedan acceder a documentos de otras comunidades. Cada solicitud al servicio RAG incluye validación de la comunidad solicitante, evitando fugas de información entre contextos aislados.

Por otro lado, la infraestructura implementa un modelo de defensa en profundidad mediante la segregación de componentes críticos en subnets privadas sin acceso directo a Internet. Los roles IAM se configuraron siguiendo el principio de mínimo privilegio, otorgando únicamente los permisos estrictamente necesarios para cada servicio. Los *Security Groups* actúan como firewalls virtuales que restringen el tráfico únicamente a los puertos y protocolos requeridos, mientras que el *NAT Gateway* posibilita que los servicios internos puedan realizar actualizaciones y comunicarse con servicios externos sin exponerse directamente. Para permitir que los desarrolladores accedan de forma segura a recursos internos durante tareas de diagnóstico y mantenimiento, se implementó *AWS Systems Manager Session Manager* mediante una instancia bastión en subnet privada, eliminando la necesidad de exponer puertos SSH públicos y proporcionando acceso auditable y controlado a la base de datos y otros componentes críticos sin comprometer la seguridad perimetral del sistema.

Usabilidad

El atributo de usabilidad influyó directamente en decisiones de arquitectura orientadas a ofrecer una experiencia fluida, intuitiva y accesible para usuarios con diversos perfiles. Se optó por una arquitectura de *frontend* basada en *Single Page Application* (SPA) con renderizado del lado del servidor (SSR) mediante *Next.js*, lo que permite tiempos de carga optimizados y una navegación sin interrupciones, mejorando significativamente la percepción de rapidez y responsividad del sistema.

La arquitectura *multi-tenant* implementada permite que cada comunidad personalice la apariencia visual de su espacio mediante temas configurables, fortaleciendo el sentido de

pertenencia y facilitando la identificación del entorno de trabajo. Se diseñaron flujos de navegación claros y predecibles, con validaciones en tiempo real que previenen errores de usuario mediante retroalimentación inmediata y mensajes descriptivos. Adicionalmente, se priorizó la accesibilidad siguiendo estándares WCAG 2.1 nivel AA, garantizando compatibilidad con tecnologías asistivas y soporte multi idioma, lo que amplía significativamente el alcance demográfico del sistema.

En cuanto al componente RAG, se priorizó la selección de un modelo de lenguaje capaz de generar respuestas coherentes, contextualizadas y comprensibles, minimizando alucinaciones y garantizando que las respuestas se ajusten estrictamente al conocimiento almacenado en la base vectorial. Se implementaron mecanismos de evaluación continua que permiten medir la calidad y relevancia de las respuestas generadas, asegurando que el sistema cumpla con las expectativas de satisfacción de los usuarios finales.

Flexibilidad

La flexibilidad del sistema fue un factor determinante para asegurar su capacidad de evolución y adaptación a futuros cambios en requisitos, tecnologías o contextos de uso. Para ello, se adoptó ampliamente el uso de patrones de diseño que promueven el bajo acoplamiento y facilitan la extensibilidad sin comprometer la estabilidad del sistema existente. Entre los patrones implementados se destacan el patrón *Repository* para abstraer el acceso a datos y desacoplar la lógica de negocio de la capa de persistencia, el patrón *Singleton* gestionado mediante un contenedor de inyección de dependencias que garantiza instancias únicas de servicios críticos, el patrón *Facade* aplicado en el cliente RAG para simplificar la comunicación con el servicio de inteligencia artificial externo.

La arquitectura modular basada en contenedores *Docker* permite desplegar el sistema en distintos entornos (local, nube) con mínima configuración adicional, facilitando tanto el desarrollo como la operación en producción. Se diseñó el *frontend* con enfoque responsive, adaptándose automáticamente a diferentes dispositivos y tamaños de pantalla sin requerir versiones especializadas. Además, la separación clara entre componentes (*frontend*, *backend*, base de datos, servicio del sistema de *IA*) y el uso de interfaces bien definidas permiten reemplazar o actualizar tecnologías específicas sin afectar al resto del sistema, garantizando la capacidad de incorporar mejoras tecnológicas conforme evoluciona el ecosistema de LLMs y herramientas asociadas.

En el componente RAG, se implementó la combinación de los patrones *Factory* y *Adapter* mediante una interfaz abstracta (*DatabaseInterface*) que define las operaciones estándar sobre bases de datos vectoriales, y un *DatabaseFactory* que instancia el adaptador correspondiente según la configuración del sistema. El *PineconeAdapter* implementa esta interfaz, encapsulando toda la lógica específica de *Pinecone*, lo que permite intercambiar proveedores de bases de datos vectoriales sin modificar la lógica de negocio del RAG. Esta misma arquitectura se replica para los proveedores de *embeddings* (*OpenAI*, *HuggingFace*), permitiendo evaluar diferentes alternativas según criterios de costo, rendimiento y calidad.

6.4. Desarrollo de la arquitectura

Esta sección describe la implementación concreta de los componentes que conforman la plataforma NicheAI, detallando cómo los principios arquitectónicos y atributos de calidad definidos se materializan en una solución técnica operativa.

El desarrollo del sistema se organiza en tres subsistemas principales. El *backend* lógico (Niche-API) funciona como el núcleo operativo, mediando entre las solicitudes externas, la lógica de negocio y las fuentes de datos mediante una arquitectura en capas inspirada en los principios de *Clean Architecture*. El *backend* de inteligencia artificial (RAG-API) constituye el componente diferenciador del sistema, implementando técnicas de RAG a través de dos estrategias complementarias, un flujo determinístico basado en grafos de estado y un enfoque agéntico con razonamiento autónomo. Finalmente, el *frontend* (Web-App) proporciona una interfaz desarrollada en *React/Next.js* con soporte *multi-tenant*, integrando capacidades de internacionalización, personalización temática y observabilidad de la experiencia de usuario, garantizando una interacción flexible y adaptada a distintos contextos.

Cada subsección detalla la estructura interna, las decisiones tecnológicas y los patrones aplicados, traduciendo los requerimientos en una implementación modular y preparada para extenderse a otros dominios.

6.4.1. Backend Lógico

El componente Niche-API constituye el núcleo operativo de la plataforma y cumple un papel fundamental dentro de la arquitectura del sistema. Su función principal es intermediar entre las solicitudes externas, la lógica de negocio y las fuentes de datos, actuando como un punto de integración entre la base de datos principal, el servicio de inteligencia artificial y el sistema de notificaciones.

Este servicio se ejecuta en un entorno contenedorizado mediante *AWS ECS Fargate*, lo que permite una gestión completamente administrada de los contenedores, con despliegues independientes y escalabilidad automática, sin requerir la administración de la infraestructura subyacente.

La aplicación se construye mediante un build *Docker multi-stage* optimizado, lo que permite generar una imagen basada en *Node.js* y *TypeScript*, que incorpora tanto las dependencias necesarias como el código fuente del sistema. La arquitectura de despliegue está diseñada para garantizar, en la medida de lo posible, una alta disponibilidad a través de una implementación multi-zona de disponibilidad (*Multi-AZ*) y verificaciones de estado (*health checks*) continuas.

El autoescalado dinámico se gestiona automáticamente a partir de métricas de *CPU* y tráfico entrante, mientras que el *Application Load Balancer* (ALB) se encarga de distribuir las solicitudes entrantes hacia los contenedores activos mediante el endpoint interno */api/*.

Todo el monitoreo, el registro de métricas y la trazabilidad de eventos se administra mediante *AWS CloudWatch*, lo que proporciona una visibilidad completa sobre el comportamiento del servicio y facilita la detección temprana de anomalías o degradaciones en el rendimiento. En la Ilustración 30 se muestra cómo se visualizan dichos logs en la consola de AWS.

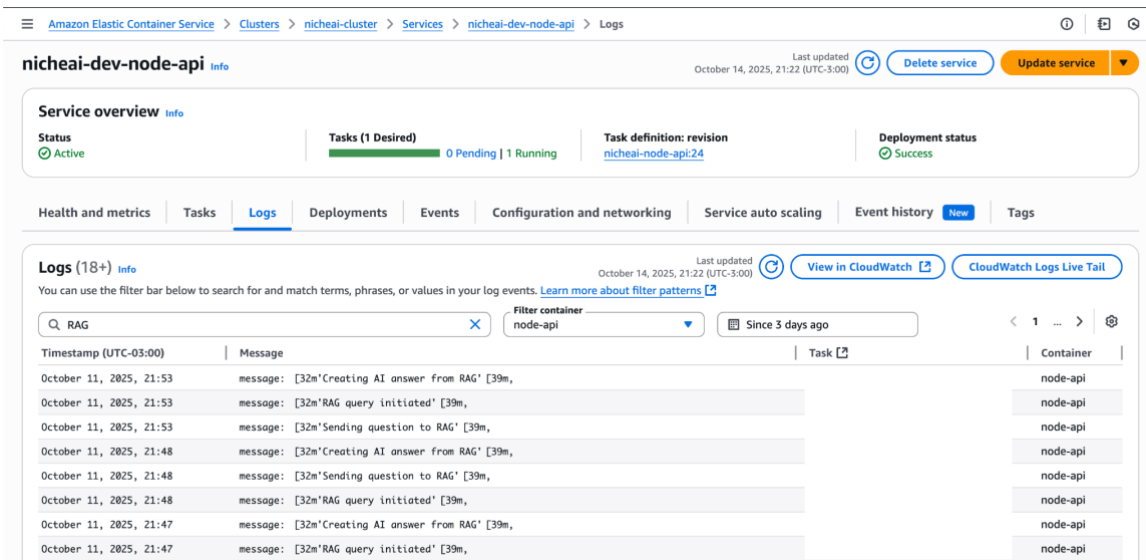


Ilustración 30: Logs del Niche-API filtrados por la palabra “RAG”.

6.4.1.1. Arquitectura interna

La estructura interna de Niche-API se fundamenta en una arquitectura en capas inspirada en los principios de *Clean Architecture* [32] y *Domain-Driven Design* (DDD). Este enfoque busca separar las responsabilidades y promover un bajo acoplamiento entre los distintos módulos, favoreciendo la mantenibilidad, extensibilidad y testabilidad del sistema.

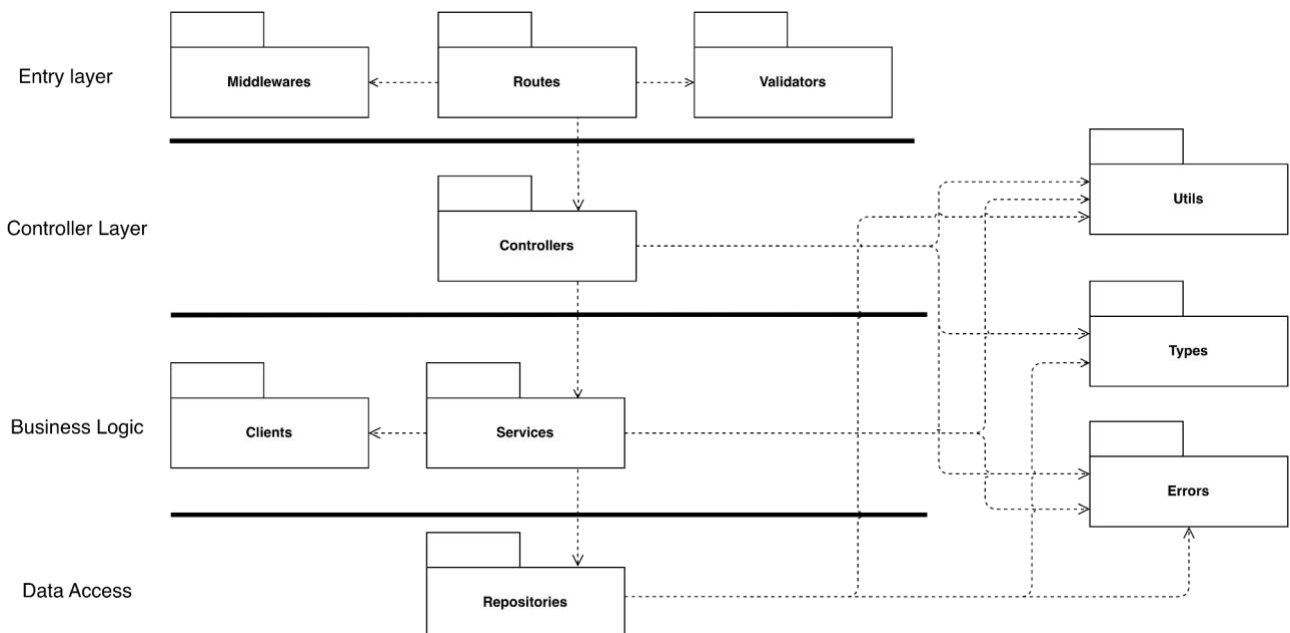


Ilustración 31: Diagramas de capas del backend Niche-API

Cómo se observa en la Ilustración 31, en la capa superior se ubican las *routes* y *validators*, responsables de definir los endpoints públicos y las reglas de validación que garantizan la integridad de los datos de entrada. Los *middlewares* actúan como una capa transversal que intercepta las solicitudes HTTPS antes de que lleguen a los controladores, aplicando mecanismos de autenticación, autorización, control de acceso y registro de trazas. Su funcionamiento sigue el patrón *Chain of Responsibility*, lo que permite encadenar distintas funciones de procesamiento de manera ordenada y flexible.

Las *controllers* actúan como punto de coordinación entre las solicitudes entrantes y la lógica de negocio, derivando la ejecución hacia los *services*, que constituyen el núcleo funcional de la aplicación. Los *services* concentran la lógica de negocio y orquestan las operaciones de los módulos inferiores, asegurando coherencia y consistencia del dominio.

En un nivel inferior, se encuentran los *repositories*, implementados mediante *Prisma ORM*, los cuales abstraen el acceso a la base de datos y mantienen la independencia del motor de persistencia. Esta capa de abstracción se implementa siguiendo el *Repository Pattern*, facilitando las pruebas unitarias y reduciendo el impacto de posibles cambios tecnológicos en la capa de almacenamiento.

La arquitectura se completa con los módulos de *clients*, *utils*, *types* y *errors*. Los *clients* encapsulan la comunicación con servicios externos, como lo es RAG-API, aplicando el *Adapter Pattern* para aislar la integración. Los módulos de *utils*, *types* y manejo de errores (*errors*) proveen componentes reutilizables, definiciones de tipo y una estructura homogénea para la captura y propagación de excepciones.

6.4.1.2. Integración con otros componentes del sistema

El componente Niche-API mantiene una comunicación bidireccional con diversas partes del sistema. Su interacción más directa se da con la base de datos principal, implementada en *PostgreSQL*, a la que accede mediante *Prisma ORM*. Este mecanismo permite ejecutar migraciones automáticas, garantizar el cumplimiento de transacciones ACID, (Atomicidad, Consistencia, Aislamiento y Durabilidad) que garantizan fiabilidad en transacciones de bases de datos, y ofrecen un entorno de persistencia seguro y consistente.

Por otra parte, Niche-API establece comunicación con la RAG-API a través de peticiones *HTTPS* con control de *timeouts* y reintentos automáticos, encapsuladas bajo un cliente

especializado. Con el objetivo de aumentar la resiliencia ante fallos externos, se implementa el *Circuit Breaker Pattern*, que permite gestionar caídas temporales o latencias excesivas sin comprometer la disponibilidad global del sistema. En términos operativos, cada vez que un usuario genera una pregunta, esta se envía como consulta a la RAG-API. Este proceso se abordará con mayor detalle en capítulos posteriores.

De manera complementaria, el sistema publica eventos relevantes, como lo son invitaciones en un *Topic SNS*, el cual activa una función *AWS Lambda* encargada de enviar notificaciones por correo electrónico. Este mecanismo permite mantener una arquitectura asíncrona basada en eventos, desacoplando la generación de información del proceso de notificación. Finalmente, el *Application Load Balancer* supervisa la disponibilidad de los contenedores ECS y expone un endpoint de salud */health/ping* para permitir la verificación continua del estado del servicio.

6.4.1.3. Justificación tecnológica

La elección de las tecnologías que conforman Niche-API responde a una combinación de factores técnicos y estratégicos. *Node.js* proporciona un modelo de programación asíncrono y orientado a eventos, eficiente para manejar operaciones de entrada y salida intensivas, como consultas a bases de datos o llamadas a servicios externos. Al combinarlo con *TypeScript*, se incorpora un tipado estático y verificaciones en tiempo de compilación, lo que permite detectar errores tempranamente, mejorar la mantenibilidad y documentar implícitamente los tipos de datos en toda la aplicación.

El uso de *Express.js*, un *framework* minimalista para *Node.js*, aporta flexibilidad y control granular sobre rutas, *middleware* y flujo de ejecución de solicitudes *HTTPS*. Esta elección facilita la implementación explícita de mecanismos de autenticación, autorización, validación y registro de eventos (*logging*), sin la rigidez estructural que podrían imponer *frameworks* más prescriptivos.

Para la capa de persistencia, *Prisma ORM* [33] fue seleccionado por su integración nativa con *TypeScript*, su capacidad de generar migraciones automáticas y su enfoque *type-safe*, garantizando la integridad de los datos en tiempo de compilación. Esto permite abstraer la complejidad de las consultas SQL, mantener independencia respecto al motor de base de datos y mejorar la testeabilidad de la lógica de acceso a datos.

Asimismo, la utilización del contenedor de inyección de dependencias *tsyringe* [34] permite implementar un diseño desacoplado y modular, favoreciendo la separación de responsabilidades entre componentes.

En cuanto a la infraestructura, *AWS ECS Fargate* se eligió por ofrecer un entorno de ejecución administrado, autoescalable y sin necesidad de gestionar servidores, simplificando el despliegue de contenedores *Docker*. Su integración con *Application Load Balancer* permite distribuir el tráfico de manera eficiente entre instancias, mientras que *CloudWatch* y *Parameter Store* facilitan el monitoreo centralizado, la trazabilidad de eventos y la gestión segura de configuraciones sensibles.

Durante la fase de evaluación tecnológica, se analizaron alternativas como *NestJS*, un *framework* altamente estructurado que ofrece integración completa con *TypeScript* y soporte para patrones arquitectónicos avanzados, y *Next.js*, que permite desarrollar aplicaciones *fullstack* combinando *frontend* y *backend* en un mismo proyecto. Sin embargo, *NestJS* fue descartado debido a su sobrecarga estructural y a la menor flexibilidad para personalizar la capa de *middleware*. Por su parte, *Next.js* no resultaba óptimo para un servicio puramente *backend*, ya que se buscaba mantener el aislamiento con el *frontend* y controlar de manera más directa la comunicación entre los servicios, incluyendo la interacción con el servicio de IA. Por estas razones, se optó por la combinación de *Node.js* + *Express.js*, que permite un control más preciso sobre el flujo de ejecución y una curva de aprendizaje más plana para el equipo de desarrollo.

6.4.1.4. Características operativas y patrones aplicados

El diseño de Niche-API incorpora una serie de mecanismos destinados a mejorar la seguridad, la escalabilidad y la observabilidad.

Sistema de autenticación dual

El sistema de autenticación dual constituye el núcleo de la estrategia de seguridad de Niche-API. Este modelo define dos niveles jerárquicos de sesión que operan de manera complementaria: una sesión global (*UserSession*) y una sesión específica por comunidad (*UserTenantSession*).

Ambas capas trabajan en conjunto para proporcionar una autenticación fluida, segura y contextualizada, manteniendo la integridad del proceso de acceso en entornos donde los usuarios pueden pertenecer a múltiples comunidades.

El primer nivel de autenticación se establece a través del modelo *UserSession*, que representa la sesión global del usuario en el sistema. Tras un inicio de sesión exitoso, el usuario obtiene un *token JWT* que no contiene información asociada a ninguna comunidad. Este *token* permite la interacción con rutas generales como */auth/tenants* y habilita la posibilidad de cambiar de comunidad sin necesidad de volver a autenticarse. En este nivel, el sistema valida la identidad del usuario de forma independiente del contexto de la comunidad, actuando como un punto de entrada común para todas las comunidades a las que el usuario tiene acceso.

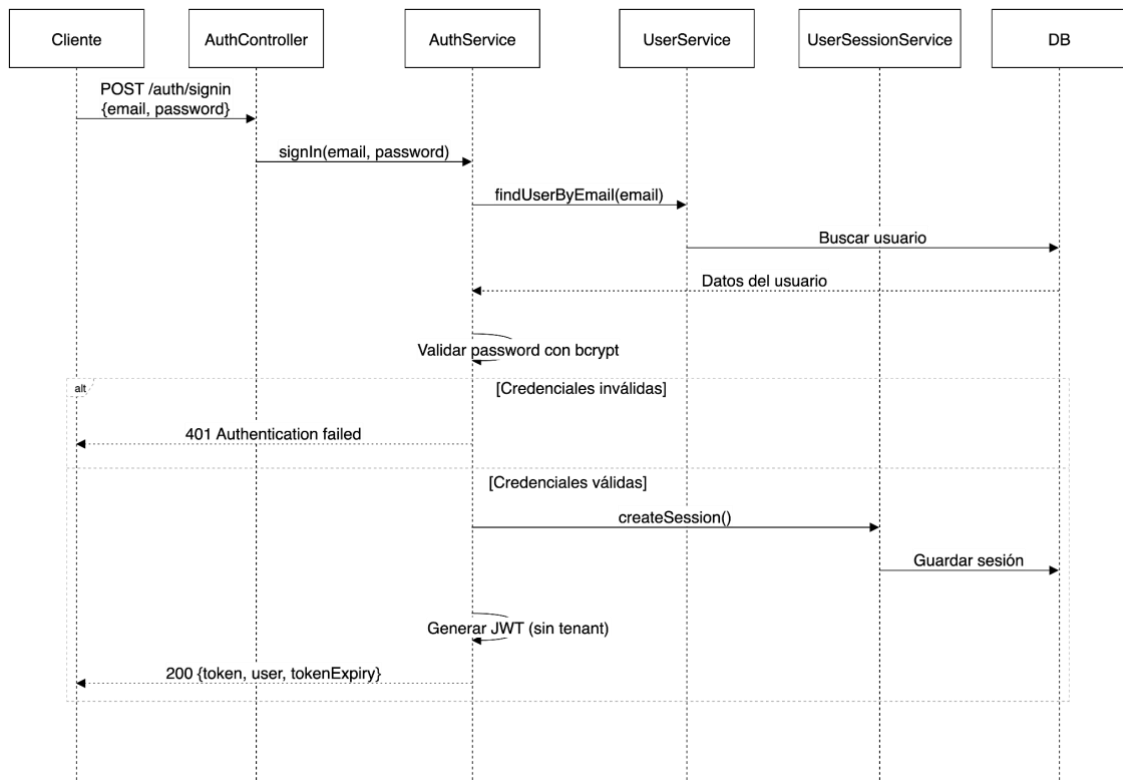


Ilustración 32: Diagrama de secuencia de la autenticación del primer nivel

Una vez autenticado globalmente, el usuario selecciona la comunidad con la que desea operar. En ese momento se genera una sesión específica de la comunidad, representada por el modelo *UserTenantSession*, que incorpora el *tenantId*, el rol correspondiente del usuario dentro de ese contexto e información extra como el estado del usuario dentro de esa comunidad o la fecha de cuando se unió a la misma.

Este *token* restringe el acceso a los recursos pertenecientes exclusivamente al tenant seleccionado, garantizando la correcta aplicación de los permisos definidos. De esta manera, cada operación se ejecuta dentro de un entorno completamente aislado, con un control granular de los privilegios en función del contexto organizacional activo.

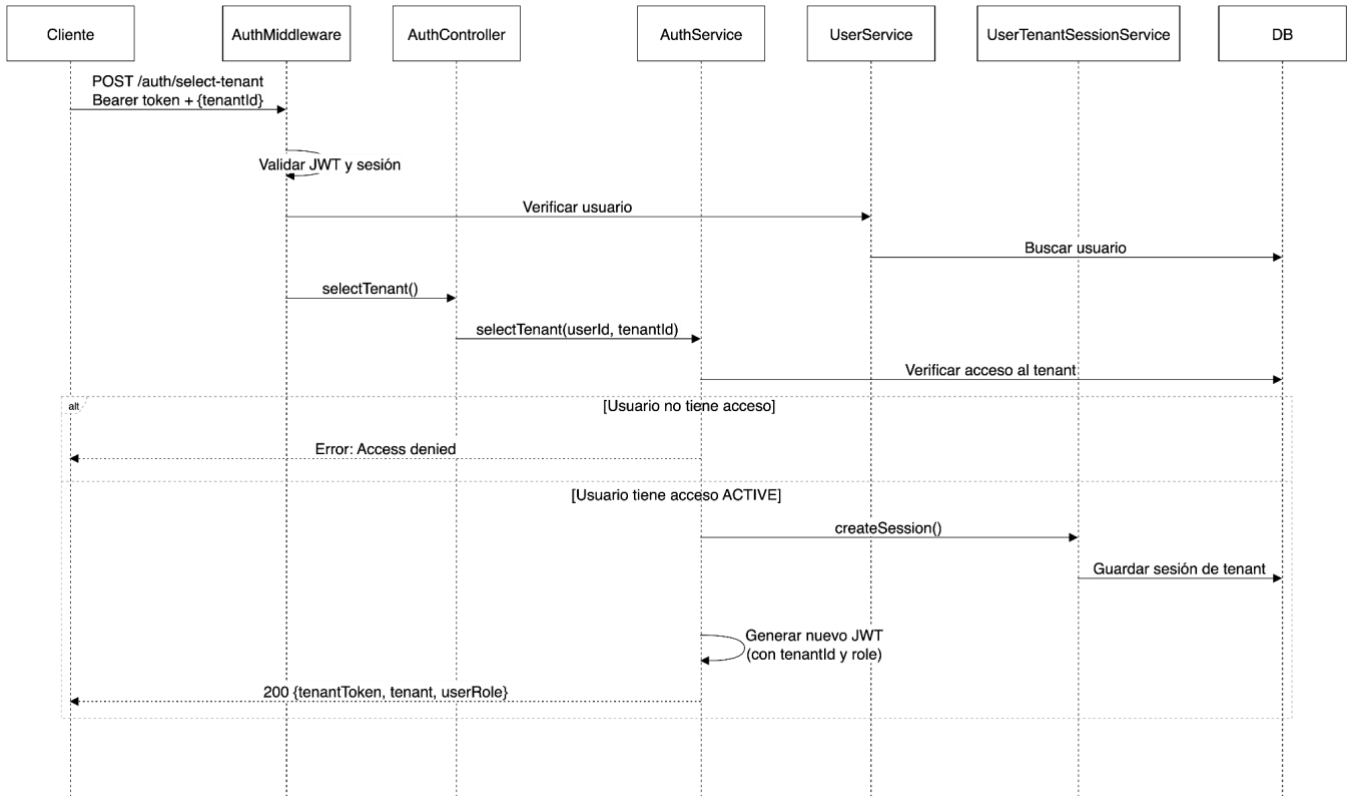


Ilustración 33: Diagrama de secuencia de la autenticación del segundo nivel

Realizar pregunta

El proceso de creación de preguntas constituye uno de los flujos centrales del sistema, ya que integra múltiples capas arquitectónicas e involucra la interacción con servicios externos de inteligencia artificial.

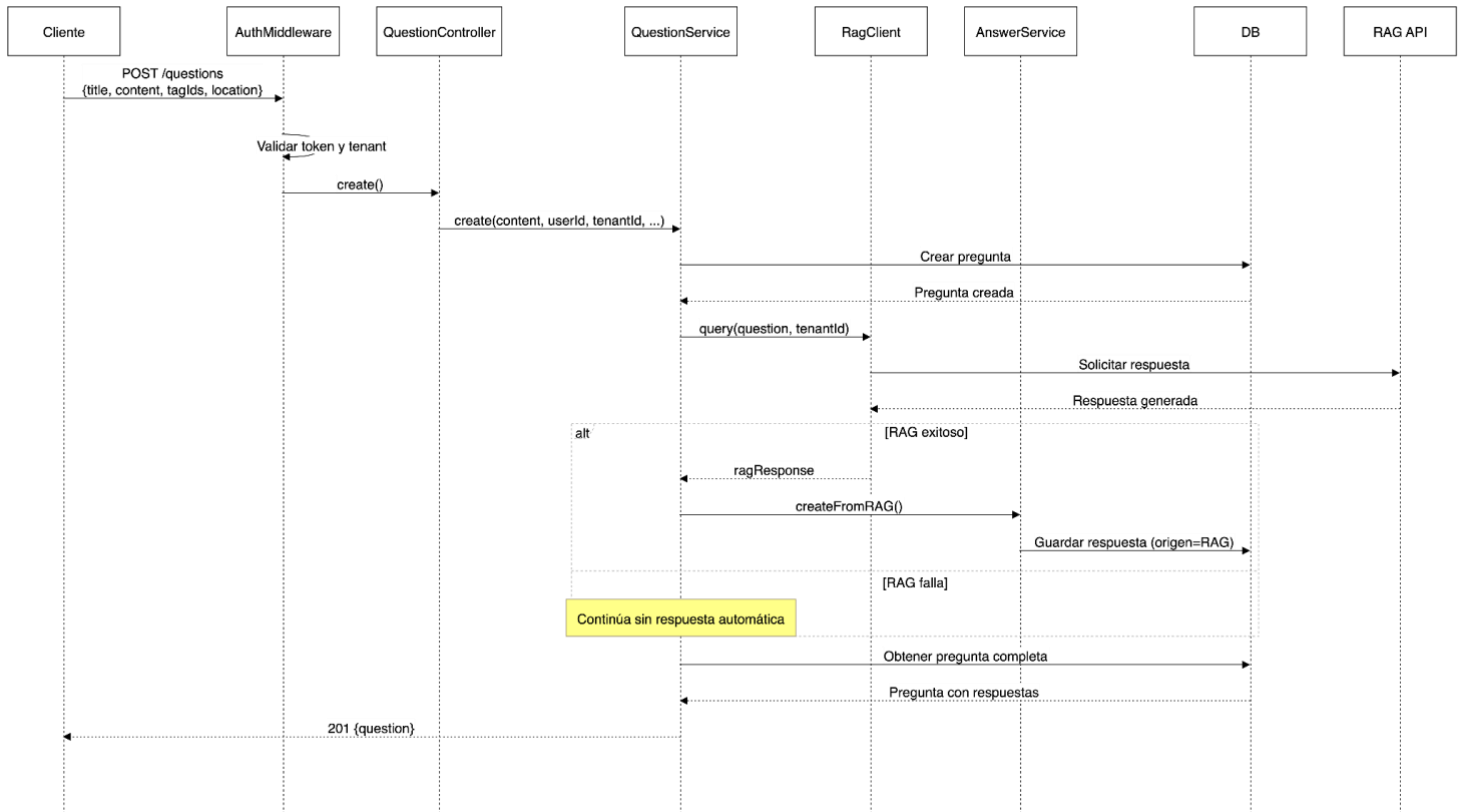


Ilustración 34: Diagrama de secuencia de realizar una pregunta

El flujo de creación de una pregunta inicia cuando un cliente *HTTPS* envía una solicitud *POST* al endpoint */questions*, incluyendo datos como título, contenido, etiquetas y ubicación. Esta solicitud pasa primero por el *ValidationMiddleware*, que aplica el principio *fail-fast* para asegurar que los datos cumplan con el formato y las reglas esperadas, deteniendo el proceso ante cualquier error. Si las validaciones son exitosas, el *QuestionController* toma el control, actuando como intermediario que extrae los parámetros y delega la lógica principal al *QuestionService*, siguiendo el patrón *skinny controller, fat service* [35].

En el *QuestionService* se gestionan las reglas de negocio vinculadas a la creación de preguntas. Se verifica la coherencia de los datos y la validez de los identificadores antes de delegar la persistencia al *QuestionRepository*, implementado con *Prisma ORM*. Este

repositorio se encarga de almacenar la pregunta y sus relaciones dentro de una transacción SQL, asegurando la consistencia de los datos. Tras la persistencia, se envía la pregunta al RAG-API, que retorna una respuesta. El sistema incluye tolerancia a fallos, permitiendo continuar con la creación de la pregunta incluso si el servicio RAG no está disponible.

Si la RAG-API produce una respuesta válida, esta se procesa a través del *AnswerService*, que define su estado inicial según la configuración del tenant: puede quedar como *PENDING* para revisión manual o publicarse directamente como *APPROVED*. Las respuestas se almacenan con el origen identificado como RAG, diferenciándolas de las humanas. Finalmente, el *QuestionService* devuelve la pregunta recién creada con sus respuestas, votos y metadatos asociados. Todo el flujo está protegido por un middleware centralizado de manejo de errores que asegura respuestas coherentes y registra eventos para monitoreo y auditoría.

6.4.2. Backend del Sistema de IA

El *backend* del sistema de inteligencia artificial constituye el núcleo lógico encargado de procesar las consultas de los usuarios y generar respuesta fundamentadas en la información especializada sobre biosais. Su propósito principal es resolver los cuatro tipos de preguntas definidos en los requerimientos funcionales: aquellas referidas a nombres comunes distintos para una misma especie, nombres iguales que identifican especies diferentes, generalizaciones por género y generalizaciones amplias cuando no se dispone de información específica.

Además del procesamiento de consultas, el *backend* gestiona los procesos de carga, administración e indexación de las fuentes de datos del sistema, que incluyen el *nomenclátor* botánico y las *data pills* informativas. Estos procesos aseguran la disponibilidad de la información en la base de datos vectorial. Para ello, el sistema expone una API que centraliza las operaciones principales mediante *endpoints* dedicados a la realización de consultas, la carga de datos y la recuperación de los documentos más relevantes asociados a cada solicitud.

Entre los atributos de calidad definidos para este componente, la adecuación funcional se establece como el más relevante. Este atributo, entendido como la capacidad del sistema para proveer resultados exactos y relevantes cuando es utilizado por los usuarios

específicos, representa el criterio fundamental de éxito del producto. Tanto el cliente como los usuarios finales identificaron la precisión de las respuestas como la principal necesidad durante la validación del problema, dado que la información incorrecta o contradictoria constituye uno de los mayores desafíos dentro del ámbito del bonsái.

En consecuencia, el *backend* fue diseñado con el objetivo de priorizar la fiabilidad de las respuestas y la detección de escenarios en los que el modelo no dispone de información suficiente, evitando así las denominadas alucinaciones propias de los modelos de lenguaje. Este enfoque se traduce en una arquitectura que no solo busca responder correctamente, sino también reconocer sus límites, preservando la confianza del usuario en el sistema.

Un segundo objetivo de calidad clave es la mantenibilidad, asociada a la capacidad del sistema para evolucionar sin introducir cambios estructurales significativos. A solicitud del cliente, el *backend* fue desarrollado siguiendo principios de modularidad e interfaces desacopladas, lo que permite sustituir con facilidad los proveedores de modelos de lenguaje o bases vectoriales. Este diseño asegura su adaptabilidad futura ante avances tecnológicos o nuevos requerimientos, garantizando así la continuidad y escalabilidad de la solución.

6.4.2.1. Arquitectura general del backend del sistema de IA

El *backend* implementa dos modos principales de operación, diseñados para responder eficientemente a diferentes tipos de consultas: un flujo con agente, orientado a preguntas complejas que requieren razonamiento o múltiples pasos, y un flujo sin agente, optimizado para consultas directas y de menor complejidad.

En el primer caso, la solicitud del usuario atraviesa un proceso de clasificación inicial que determina el tipo de pregunta formulada. A partir de ello, el sistema identifica el nombre del árbol mencionado, construye la información contextual relevante y delega la resolución a un agente, responsable de orquestar las herramientas necesarias para obtener la respuesta. Este agente actúa como un coordinador inteligente dentro del flujo, seleccionando y combinando dinámicamente las operaciones requeridas, como búsquedas, inferencias o verificaciones, en función de las necesidades específicas de la consulta.

Por su parte, el modo RAG tradicional adopta un enfoque más lineal. En este flujo, la pregunta es analizada para identificar la especie involucrada y construir un contexto informativo con los documentos más relevantes. Luego, se ejecutan recuperaciones progresivas a distintos niveles, especie, género y general, hasta obtener una respuesta satisfactoria o determinar con certeza que no existe información suficiente en la base de conocimiento.

El *backend* fue implementado utilizando *LangGraph*, un *framework* de código abierto orientado al desarrollo de aplicaciones de inteligencia artificial basadas en grafos de estado. Este enfoque permite modelar flujos de trabajo complejos donde cada nodo representa una operación específica (por ejemplo, recuperar información o generar una respuesta) y las aristas definen las transiciones entre estados según condiciones lógicas o resultados intermedios.

El sistema mantiene un estado compartido (*State*) que se propaga a lo largo de todo el grafo, garantizando la coherencia de la información durante la ejecución del flujo. Este estado comprende datos esenciales, incluyendo la pregunta original del usuario, el nombre del árbol identificado, la información taxonómica completa (especie, género, familia y orden), los documentos recuperados que conforman el contexto, la respuesta generada y los metadatos asociados al nivel de especificidad alcanzado.

Gracias a esta arquitectura, el *backend* logra combinar razonamiento estructurado, recuperación contextual y generación de respuestas en un único flujo coherente, garantizando precisión, trazabilidad y flexibilidad en la interacción con los usuarios.

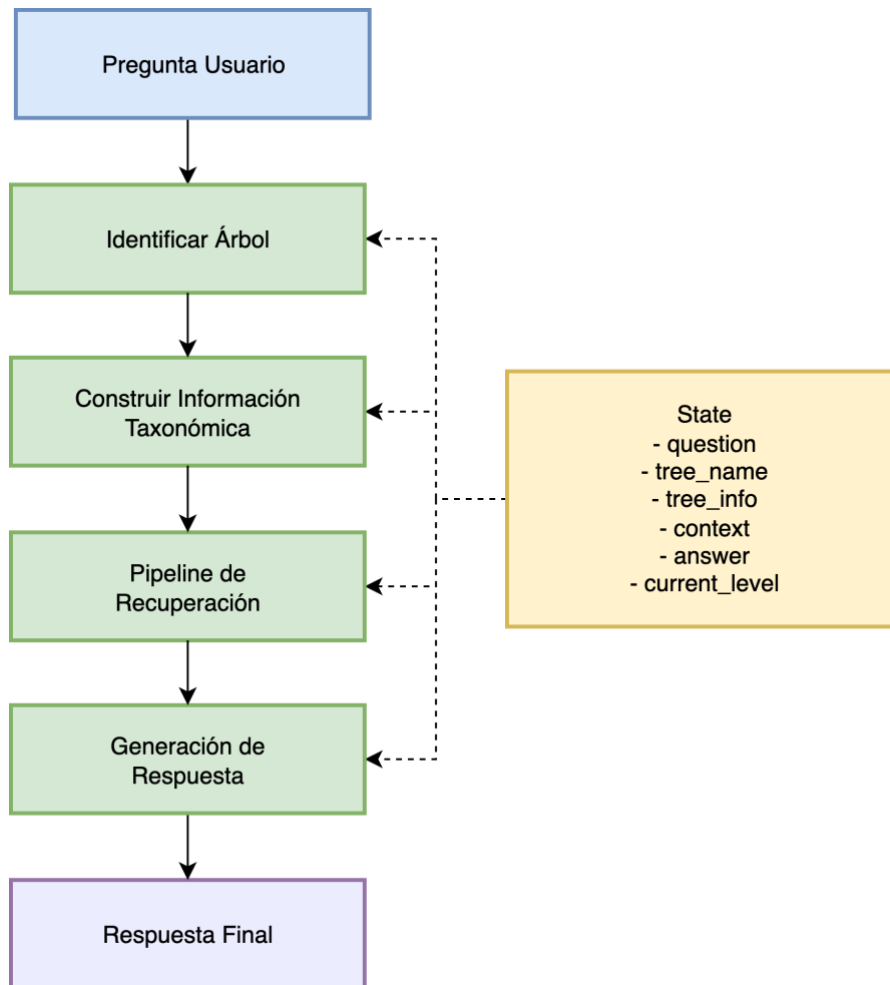


Ilustración 35: Flujo general del proceso y estructura del estado compartido del backend.

La arquitectura del *backend* se organiza, por tanto, como una secuencia de etapas interconectadas que combinan identificación, recuperación y generación de conocimiento. La Ilustración 35 sintetiza este flujo general, mostrando las principales fases del proceso y la estructura del estado compartido que se mantiene a lo largo de la ejecución. Este estado garantiza la trazabilidad de la información y la coherencia entre las operaciones del grafo.

6.4.2.2. Componentes del backend

El *backend* del sistema de inteligencia artificial se compone de un conjunto de módulos interconectados que colaboran para procesar las consultas de los usuarios, recuperar la información relevante y generar respuestas fundamentadas. La Ilustración 36 presenta la vista lógica de estos componentes y las relaciones entre ellos.

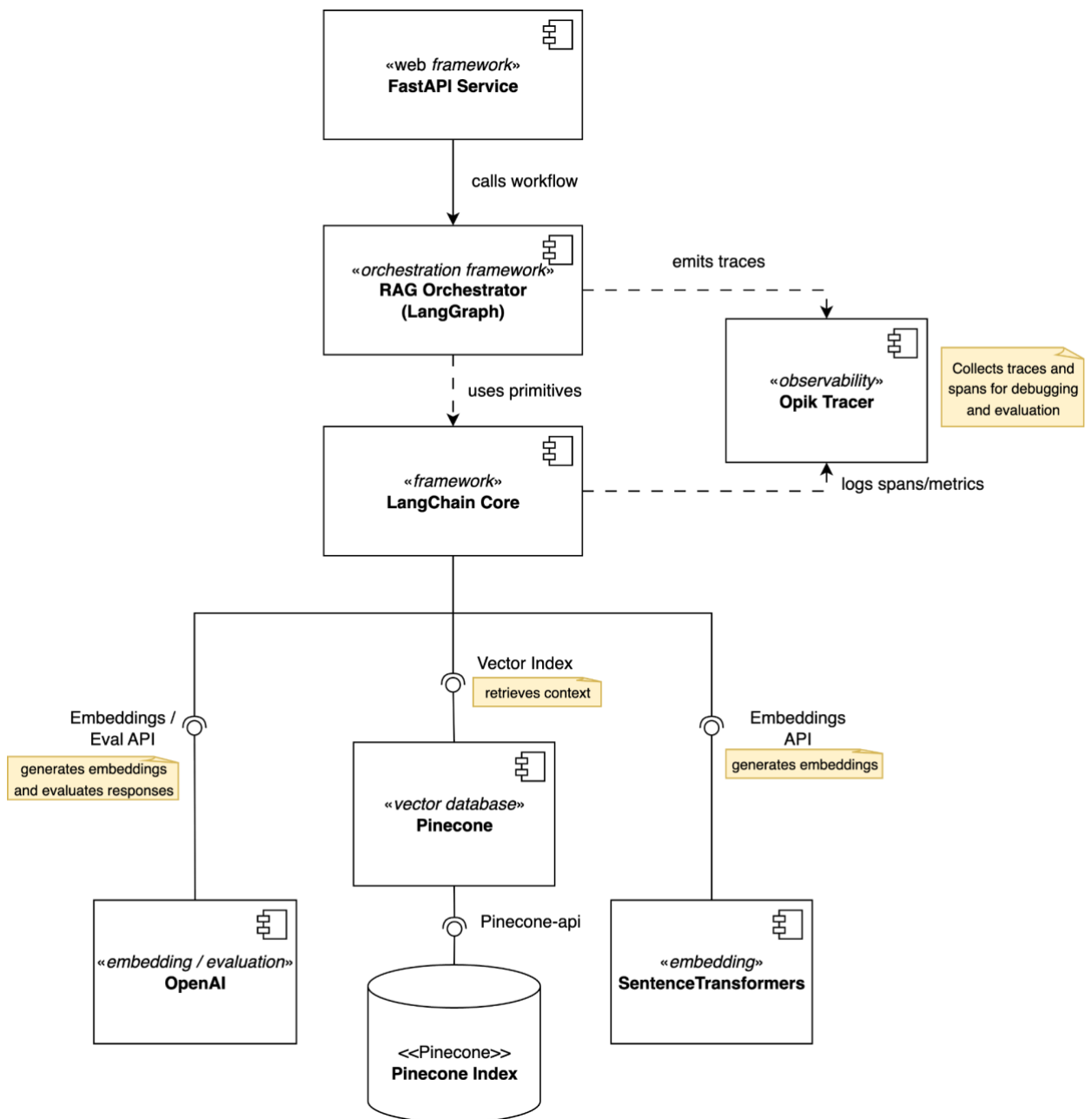


Ilustración 36: Diagrama de componentes y conectores del backend del sistema de inteligencia artificial.

En la capa superior se encuentra el servicio web, implementado con el *framework FastAPI*, que expone los *endpoints* del sistema y gestiona las solicitudes entrantes. Este servicio constituye el punto de acceso principal a las funcionalidades del *backend*, incluyendo el envío de preguntas, la carga de datos y la obtención de los documentos más relevantes.

El siguiente nivel corresponde al *RAG Orchestrator*, desarrollado sobre *LangGraph*, que actúa como motor de orquestación de los flujos de inferencia. Este componente coordina las distintas etapas del proceso, clasificación de la pregunta, recuperación de contexto, razonamiento y generación de respuesta, aplicando estrategias diferenciadas según el tipo de consulta (con o sin agente). Su diseño modular permite incorporar nuevas herramientas o modificar la secuencia de pasos sin alterar la estructura general del sistema.

El *LangChain Core* constituye la base funcional sobre la cual se construyen los *chains* y *retrievers* que implementan la lógica de RAG. Este *framework* facilita la integración con modelos de lenguaje y bases vectoriales, encapsulando la complejidad de la comunicación entre componentes y favoreciendo la mantenibilidad del código. La investigación realizada sobre *LangChain* se puede encontrar en el Anexo 13.6.7

Para la persistencia y búsqueda semántica, el sistema utiliza la base vectorial *Pinecone*, donde se almacenan los *embeddings* de los documentos y su metadata. Cada comunidad o dominio de conocimiento opera dentro de su propio *namespace*, garantizando el aislamiento de la información y permitiendo la segmentación por comunidad. Desde este componente se ejecutan las búsquedas de similitud que nutren el contexto de las respuestas.

Los modelos de *embeddings* pueden ser provistos tanto por *OpenAI* como por *SentenceTransformers*, seleccionados dinámicamente mediante un *Strategy Pattern*. Este patrón de diseño permite encapsular diferentes algoritmos de generación de *embeddings* bajo una interfaz común, facilitando el intercambio entre proveedores sin modificar el código que los consume. La selección de la estrategia se realiza en función de parámetros de configuración que consideran el balance entre precisión, costo y velocidad según el caso de uso. *OpenAI* se emplea principalmente para generar *embeddings* durante las consultas de usuario y evaluar la calidad de las respuestas, aprovechando su mayor capacidad semántica. Por su parte, *SentenceTransformers* se utiliza para la indexación

inicial de documentos y recuperaciones donde el costo y la velocidad son prioritarios. Esta arquitectura desacoplada garantiza que, ante la aparición de nuevos proveedores o modelos de *embeddings*, el sistema pueda incorporarlos sin afectar la lógica central del RAG.

El *backend* incorpora además un módulo de observabilidad basado en *Opik Tracer*, encargado de recolectar *traces* y *spans* de las operaciones del sistema. Esta integración permite analizar el comportamiento interno de cada consulta, detectar cuellos de botella y evaluar el desempeño de las distintas etapas del flujo RAG.

En conjunto, estos componentes conforman un *backend* modular, extensible y trazable, capaz de combinar múltiples fuentes y proveedores de inteligencia artificial bajo una misma estructura de ejecución, garantizando así tanto la exactitud de las respuestas como la mantenibilidad futura del sistema.

6.4.2.3. Identificación del Árbol y Construcción de Información Taxonómica

Antes de iniciar el proceso de recuperación jerárquica, el sistema debe identificar el árbol mencionado en la consulta del usuario y construir su información taxonómica completa. Este proceso se realiza en dos pasos secuenciales implementados como nodos en el grafo.

Nodo de Identificación del Árbol

El primer nodo utiliza un modelo de lenguaje (*GPT-4o-mini*) para extraer el nombre del árbol mencionado en la consulta del usuario. Este componente analiza la pregunta en lenguaje natural y determina a qué árbol se refiere, pudiendo identificar tanto nombres científicos como nombres comunes.

Por ejemplo, ante la pregunta "¿Cómo debo podar mi coronilla?", el nodo identifica "coronilla" como el nombre del árbol consultado. Este nombre se almacena en el estado compartido para su uso posterior.

Nodo de Construcción de Información Taxonómica

Una vez identificado el nombre del árbol, el segundo nodo realiza una consulta al nomenclátor botánico para obtener su información taxonómica completa. Este proceso se lleva a cabo mediante dos búsquedas secuenciales. En primer lugar, se efectúa una búsqueda en el nomenclátor principal, consultando una colección especializada (*main_nomenclator*) que contiene los mapeos entre nombres comunes y nombres científicos, considerando la ubicación geográfica del usuario para resolver posibles ambigüedades (por ejemplo, "coronilla" en Uruguay frente a Argentina en dónde se conoce como "coronillo"). A continuación, con el nombre científico obtenido, se realiza una búsqueda en el nomenclátor específico, consultando el nomenclátor taxonómico completo para recuperar la clasificación jerárquica, incluyendo género, familia y orden. El resultado de este proceso es un objeto estructurado como el de la Ilustración 37.

```
{
  "specie": "Scutia buxifolia",
  "genus": "Scutia",
  "family": "Rhamnaceae",
  "order": "Rosales",
  "common_names": [
    {"name": "coronilla", "location": "Uruguay"}
  ]
}
```

Ilustración 37: Objeto taxonómico de un árbol

Este objeto taxonómico es fundamental para el funcionamiento de la generalización jerárquica, ya que proporciona los filtros necesarios para cada nivel de búsqueda.

6.4.2.4. Búsquedas Jerárquicas por Nivel Taxonómico

Tanto la implementación RAG tradicional como la agéntica comparten un conjunto común de estrategias de búsqueda jerárquica, implementadas como nodos en el primer caso y como herramientas en el segundo. Estas estrategias permiten organizar la

recuperación de información de manera progresiva según la disponibilidad y especificidad de los datos, asegurando que el sistema genere respuestas pertinentes en distintos contextos.

El sistema define cinco niveles de búsqueda, de mayor a menor especificidad. La búsqueda por especie recupera documentos filtrando por el nombre científico completo, constituyendo el nivel más preciso. La búsqueda por género abarca todas las especies dentro del mismo género, agrupando aquellas con características morfológicas y requerimientos de cuidado similares. La búsqueda por familia permite recuperar documentos de especies pertenecientes a la misma familia botánica, ofreciendo información más general pero todavía relevante gracias a las similitudes evolutivas y ecológicas.

Los niveles restantes incluyen la búsqueda por orden, que agrupa familias relacionadas dentro del mismo orden taxonómico y proporciona información más amplia, útil cuando no se dispone de datos más específicos; y finalmente, la búsqueda general se realiza sin filtros taxonómicos, recuperando información aplicable a árboles en general y principios universales de cuidado que pueden extenderse a diversas especies.

Este enfoque jerárquico asegura que el sistema pueda ofrecer respuestas útiles y adaptadas a la disponibilidad de información, desde los datos más específicos hasta recomendaciones generales.

6.4.2.5. Implementación de RAG Tradicional con Grafo

La implementación principal del sistema utiliza un grafo de estado (*StateGraph*) que define explícitamente cada nivel de búsqueda como un nodo independiente. Esta arquitectura proporciona un flujo de ejecución determinístico y transparente, donde cada transición entre niveles taxonómicos está claramente definida mediante rutas condicionales.

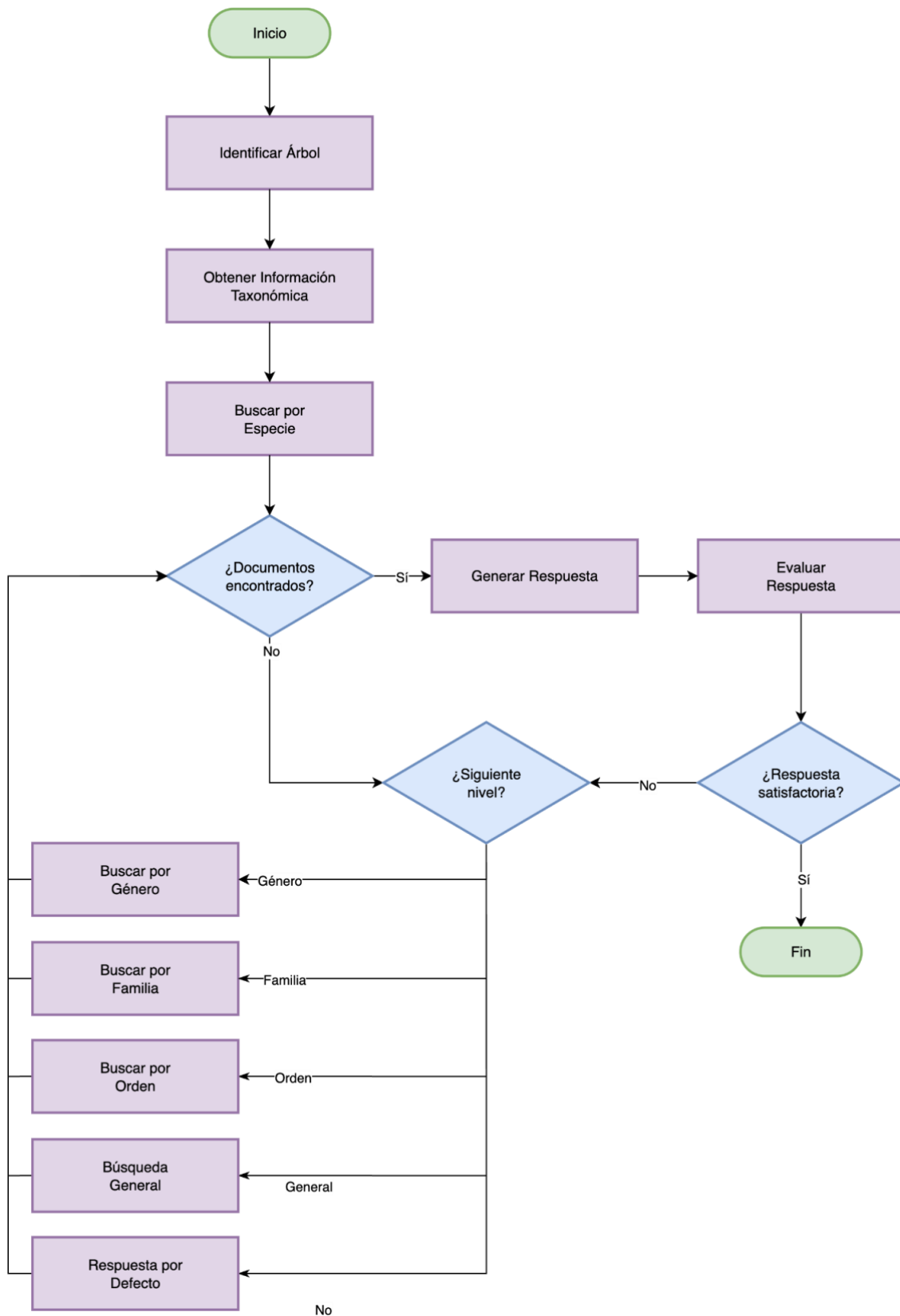


Ilustración 38: Grafo de estado completo para la implementación RAG tradicional.

Nodos de Búsqueda Jerárquica

El sistema implementa cinco nodos de búsqueda, correspondientes a los niveles taxonómicos previamente descritos: especie, género, familia, orden y general. Cada nodo, al encontrar documentos relevantes, genera de manera inmediata una respuesta preliminar mediante la invocación de un LLM, utilizando un *prompt* especializado que incluye: el contexto recuperado de los documentos, la pregunta original del usuario, el nivel de especificidad actual e instrucciones sobre cómo enmarcar la respuesta según dicho nivel.

Nodo Evaluador

Dentro de la arquitectura del sistema, un componente fundamental es el nodo evaluador, encargado de determinar si la respuesta generada satisface de manera adecuada la consulta realizada por el usuario. Este nodo se apoya en un LLM configurado para producir una salida estructurada que permite un análisis detallado de la respuesta, considerando criterios clave que aseguran su calidad y aplicabilidad.

La evaluación se centra en cuatro dimensiones principales: la utilidad práctica de la información, garantizando que el contenido pueda ser aplicado efectivamente; el respaldo documental, que asegura que las afirmaciones estén fundamentadas en los documentos recuperados; la especificidad, que determina si la información es lo suficientemente concreta para resultar relevante; y la relevancia directa, que verifica si la respuesta aborda la pregunta del usuario sin desviarse hacia información tangencial. De este modo, el nodo evaluador no solo mide la calidad de la respuesta, sino que también contribuye a la mejora continua del sistema mediante la retroalimentación obtenida.

El evaluador está configurado para ser deliberadamente permisivo con información de niveles superiores de la jerarquía.

Por ejemplo, si al buscar por especie sólo se recuperan documentos sobre poda pero la pregunta es sobre riego, el evaluador marcará la respuesta como insatisfactoria y el sistema buscará en el siguiente nivel.

Mecanismos de Enrutamiento

El grafo utiliza dos mecanismos de enrutamiento condicional que determinan el flujo de ejecución:

Enrutamiento post-búsqueda

Después de cada nodo de búsqueda jerárquica, el sistema evalúa si se encontraron documentos relevantes en la base de datos. Si la búsqueda fue exitosa y retornó documentos, el flujo se dirige al nodo evaluador para analizar la calidad de la respuesta generada. Si no se encontraron documentos en ese nivel taxonómico, el sistema avanza automáticamente al siguiente nivel de la jerarquía (de especie a género, de género a familia, etc.).

Enrutamiento post-evaluación

Después de que el evaluador analiza la calidad de la respuesta, el sistema determina el siguiente paso basándose en dos factores: si la respuesta fue considerada satisfactoria y en qué nivel taxonómico se encuentra actualmente.

Si la evaluación determina que la respuesta es satisfactoria, el flujo concluye y la información generada se retorna al usuario. En caso contrario, cuando la respuesta se considera insatisfactoria, el sistema avanza al siguiente nivel de la jerarquía taxonómica para intentar generar una respuesta más adecuada. De esta manera, si la evaluación falla en el nivel de especie, la búsqueda se amplía al nivel de género; si falla en el nivel de género, se procede al nivel de familia; y si persiste la insuficiencia de información, se continúa al nivel de orden y, posteriormente, al nivel general.

Finalmente, cuando no existen niveles superiores disponibles, el flujo dirige la consulta al nodo de respuesta por defecto, garantizando que el usuario reciba siempre algún tipo de respuesta, aunque sea general o de último recurso.

Este mecanismo de doble enrutamiento garantiza que el sistema explore exhaustivamente la jerarquía taxonómica antes de declarar que no dispone de información relevante.

Nodo de Respuesta por Defecto

En el caso extremo donde ningún nivel de la jerarquía taxonómica produce una respuesta satisfactoria según el evaluador, el sistema recurre a un nodo de respuesta por defecto que proporciona un mensaje honesto y claro al usuario: "Lo siento, pero no tengo información para responder a tu pregunta."

Este nodo cumple una función crítica en la robustez del sistema, garantizando que siempre se proporcione una respuesta al usuario, incluso cuando se han agotado todas las opciones de búsqueda disponibles sin encontrar información relevante o útil.

6.4.2.6. Implementación de *Agentic* RAG

La implementación agéntica adopta un enfoque fundamentalmente diferente basado en un agente autónomo con acceso a herramientas. En esta arquitectura, la lógica de generalización taxonómica no está codificada en un grafo explícito, sino que se define mediante instrucciones detalladas en el *prompt* del sistema que guían el comportamiento del agente.

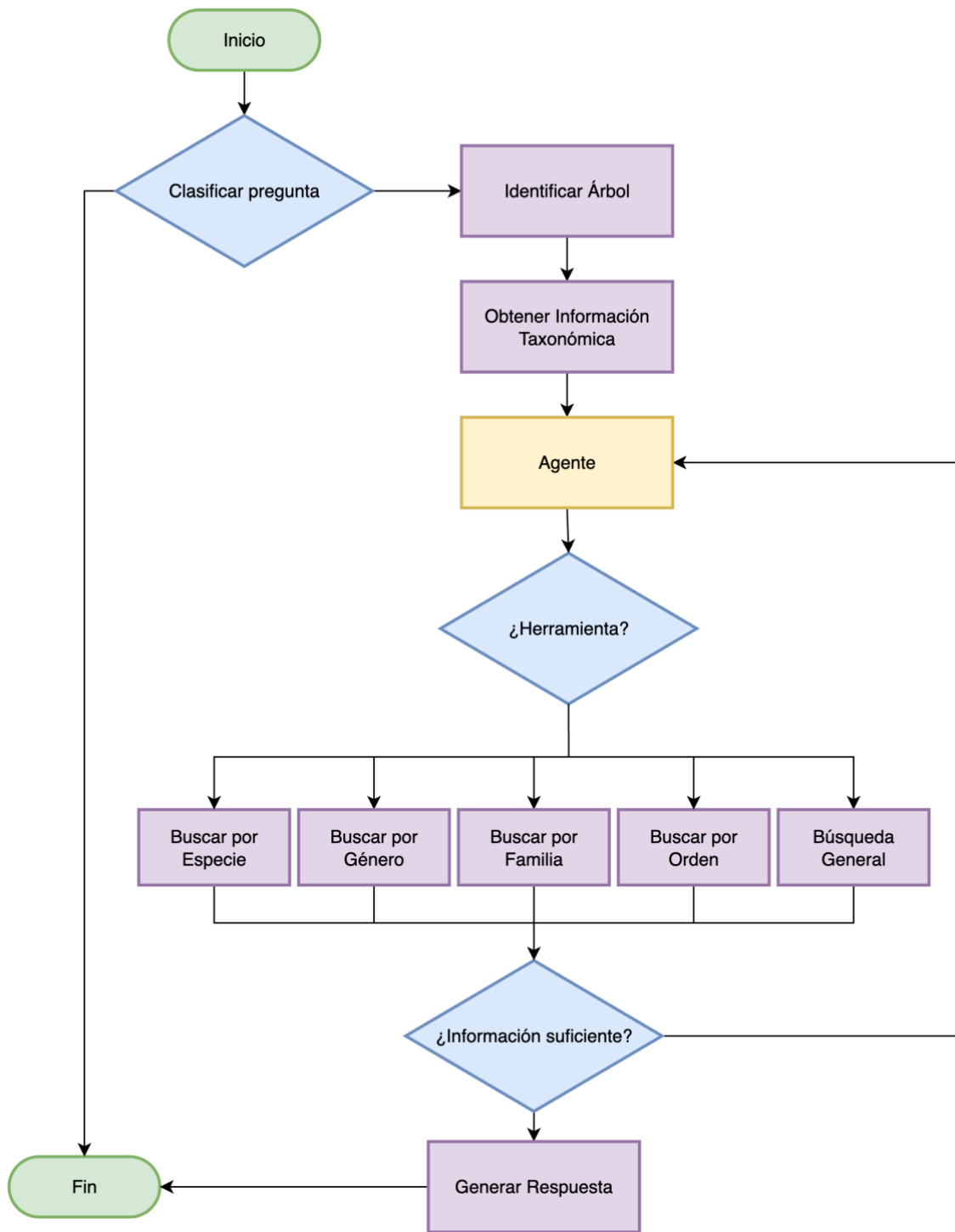


Ilustración 39: Arquitectura del sistema agéntico

Nodo Clasificador

Antes de iniciar el proceso agéntico, el sistema incorpora un nodo clasificador encargado de determinar si la consulta requiere la intervención del agente o puede ser respondida de manera directa. Entre las consultas que reciben respuesta inmediata se incluyen saludos o presentaciones ("Hola", "Buenos días"), preguntas sobre las capacidades del sistema ("¿Qué puedes hacer?"), mensajes de agradecimiento y solicitudes de aclaración no relacionadas con árboles. Estas interacciones permiten ofrecer una respuesta rápida sin necesidad de recurrir al análisis del dominio botánico.

En contraste, las consultas que requieren intervención del agente comprenden preguntas sobre especies específicas o su cuidado, consultas relacionadas con enfermedades, poda o patrones de crecimiento, y cuestiones sobre plantación, suelo o necesidades de riego. En estos casos, el agente aplica el conocimiento del dominio y los niveles taxonómicos pertinentes para generar respuestas precisas y útiles. Este paso de clasificación optimiza el sistema al evitar la ejecución innecesaria del agente y los costos asociados para consultas simples.

Herramientas del Agente

El agente cuenta con cinco herramientas especializadas que implementan las estrategias de búsqueda jerárquica descritas anteriormente. A diferencia de la implementación RAG tradicional donde el flujo está codificado explícitamente en el grafo, en esta arquitectura las herramientas funcionan como capacidades que el agente puede invocar de manera dinámica durante su ciclo de razonamiento.

Cada herramienta cuenta con documentación detallada integrada que especifica su nivel taxonómico, prioridad de uso y casos apropiados de aplicación. Esta documentación es interpretada por el agente durante su proceso de razonamiento, permitiéndole seleccionar la herramienta más apropiada en cada momento según el contexto de la consulta y los resultados obtenidos en pasos anteriores.

Sistema de *Prompts* del Agente

El sistema de *prompts* del agente se encarga de guiar su comportamiento y definir la estrategia de búsqueda jerárquica para responder a consultas sobre árboles y plantas. Esta estrategia se estructura en varias secciones que indican al agente cómo procesar la información disponible y cómo priorizar sus búsquedas.

El *prompt* establece que el agente actúa como un asistente experto en árboles y plantas, con el objetivo de encontrar información precisa y relevante sobre árboles específicos. Para lograr esto, el agente utiliza un enfoque jerárquico de búsqueda que permite acceder primero a información más específica y, si es necesario, recurrir a fuentes más generales.

El agente analiza inicialmente la información proporcionada por el usuario para identificar los niveles taxonómicos disponibles. A partir de esta información, sigue un orden de prioridad para realizar la búsqueda, avanzando a niveles más generales únicamente cuando no se encuentra información relevante en el nivel anterior.

El *prompt* también guía al agente sobre cómo formular las respuestas según el nivel de información disponible. Cuando se dispone de información específica de la especie, el agente responde directamente utilizando los datos encontrados. Si únicamente se cuenta con información a nivel de género, familia u orden, el agente debe reconocer esta limitación e indicar que los datos se aplican a ese nivel taxonómico general. Finalmente, cuando se recurre a información general, el agente debe señalar que se trata de un recurso de último nivel y basar la respuesta en principios generales sobre el cuidado de árboles.

Ciclo ReAct del Agente

El agente opera mediante un ciclo de razonamiento-acción (*ReAct*) implementado por *LangGraph*. En cada iteración, analiza el estado actual, la información disponible y los resultados obtenidos hasta el momento, basando en ello la decisión sobre qué herramienta utilizar o si debe generar la respuesta final. Luego, invoca la herramienta seleccionada, obtiene los resultados y los interpreta para determinar si son suficientes o si debe continuar la búsqueda. Este ciclo se repite hasta que el agente considera que cuenta con información suficiente para responder o hasta que ha agotado todas las opciones disponibles.

A diferencia de la implementación basada en grafo, el agente posee autonomía completa para interpretar los resultados de cada búsqueda y evaluar la relevancia de la información recuperada. Esta capacidad de razonamiento y toma de decisiones elimina la necesidad de un nodo evaluador explícito, ya que el propio agente verifica la calidad y pertinencia de la información en cada paso.

6.4.2.7. Generación de respuestas

Ambas implementaciones utilizan el mismo mecanismo fundamental para generar respuestas a partir de los documentos recuperados. El proceso de generación recibe como entrada la pregunta original del usuario, los documentos recuperados como contexto, la información taxonómica del árbol consultado, y el nivel de especificidad taxonómica alcanzado.

Con esta información, el sistema construye un *prompt* especializado que instruye al modelo de lenguaje sobre cómo formular la respuesta de manera apropiada.

El *prompt* de generación contiene un conjunto de instrucciones esenciales que aseguran que el agente proporcione respuestas precisas, relevantes y consistentes con la información disponible.

Entre estas instrucciones destacan:

- El agente debe basar sus respuestas únicamente en la información proporcionada por los documentos disponibles, evitando recurrir a conocimientos generales o suposiciones sobre el cuidado de los árboles.
- La respuesta debe construirse estrictamente a partir de los datos presentes en la información disponible, sin mencionar documentos, contexto interno o detalles de implementación.
- El agente debe enmarcar su respuesta según el nivel de especificidad de la información obtenida, indicando claramente si se trata de datos a nivel de especie, género, familia, orden o información general.
- La formulación de la respuesta debe ser útil, informativa y profesional, evitando un tono apoloético o de incertidumbre.

- En todo momento, el agente debe dejar explícito el nivel de información en el que se está basando, de manera que el usuario comprenda la precisión y el alcance de la respuesta proporcionada.

Estas instrucciones garantizan que el agente mantenga un comportamiento consistente y confiable, alineado con los objetivos de precisión y relevancia del sistema.

6.4.2.8. Comparación de Implementaciones

Aspecto	RAG Tradicional (Grafo)	Agentic RAG
Arquitectura	Grafo de estado explícito con nodos y aristas	Agente con herramientas y ciclo <i>ReAct</i>
Control de flujo	Funciones de enrutamiento condicional	Razonamiento autónomo del agente
Evaluación	Nodo evaluador explícito con <i>prompt</i> estructurado	Evaluación integrada en el razonamiento del agente
Determinismo	Completamente determinístico para mismas entradas	Variable, depende del razonamiento
Número de llamadas LLM	~7-11 (identificación + taxonomía + recuperaciones + evaluaciones)	~15-25 (incluye iteraciones <i>ReAct</i>)
Transparencia	Alta - cada paso es inspeccionable	Media - razonamiento menos explícito
Mantenimiento	Modificación de nodos y rutas en el grafo	Ajuste de <i>prompts</i> y herramientas
Manejo de casos ambiguos	Limitado a la lógica programada	Mejor capacidad de interpretación contextual

Tabla 4: Comparación detallada de las implementaciones RAG tradicional y Agentic RAG.

Para el dominio específico de este proyecto, la implementación basada en grafo se consolidó como la opción principal debido a varias ventajas clave. En primer lugar, ofrece una alineación perfecta con el dominio, ya que la jerarquía taxonómica se mapea directamente a la estructura del grafo. Además, permite una reproducibilidad que resulta

fundamental para validar el comportamiento del sistema y facilitar la depuración de problemas. Asimismo, proporciona eficiencia, con un costo computacional significativamente menor, y garantiza transparencia, al permitir que cada decisión pueda ser trazada y auditada de manera clara.

La implementación agéntica se mantiene como una alternativa experimental, ofreciendo varias utilidades. En primer lugar, permite una evaluación comparativa, brindando al cliente la posibilidad de experimentar con ambos enfoques. Por otra parte, resulta útil en casos de borde complejos, donde la rigidez de la estructura basada en grafos podría resultar limitante. Finalmente, constituye una base para investigación futura, sirviendo como punto de partida para desarrollos más sofisticados que incorporen razonamiento avanzado.

6.4.2.9. Observabilidad y evaluación del modelo

La implementación de un sistema RAG requiere mecanismos robustos de observabilidad para garantizar la calidad de las respuestas y permitir la mejora continua del sistema. En este proyecto, se integró la plataforma *Opik* para proporcionar trazabilidad completa de todas las operaciones del sistema, desde la recepción de una consulta hasta la generación de la respuesta final.

Sistema de trazabilidad

El sistema de observabilidad desarrollado permite rastrear cada invocación del grafo de *LangGraph*, capturando el flujo completo de ejecución a través de todos los nodos. Esta trazabilidad ofrece una visión integral del comportamiento del sistema y permite analizar tanto su rendimiento como la efectividad de las estrategias implementadas.

En particular, se registra la latencia de cada componente, incluyendo la extracción del nombre del árbol, las consultas al nomenclátor, la recuperación de documentos en la base vectorial y la generación de la respuesta final, lo que posibilita identificar cuellos de botella y optimizar el desempeño general. A su vez, se documenta la cantidad y tipo de documentos recuperados en cada nivel taxonómico, junto con sus puntuaciones de similitud y metadatos asociados, información clave para evaluar la eficiencia de los mecanismos de búsqueda.

El sistema también conserva un registro detallado del flujo de decisiones ejecutado por el modelo. En el grafo jerárquico tradicional, este seguimiento incluye la detección de documentos relevantes y los resultados de la evaluación de calidad de las respuestas; mientras que en la versión agéntica se monitorean las herramientas empleadas por el agente y el orden en que fueron invocadas.

Finalmente, se almacenan todos los prompts y respuestas generadas por el modelo, lo que permite realizar un análisis cualitativo del comportamiento del sistema, identificar patrones problemáticos y ajustar los parámetros de inferencia. Esta trazabilidad integral, visualizada mediante la plataforma *Opik* (Ilustración 40), constituye un componente esencial para garantizar la confiabilidad, transparencia y mejora continua del sistema de inteligencia artificial.

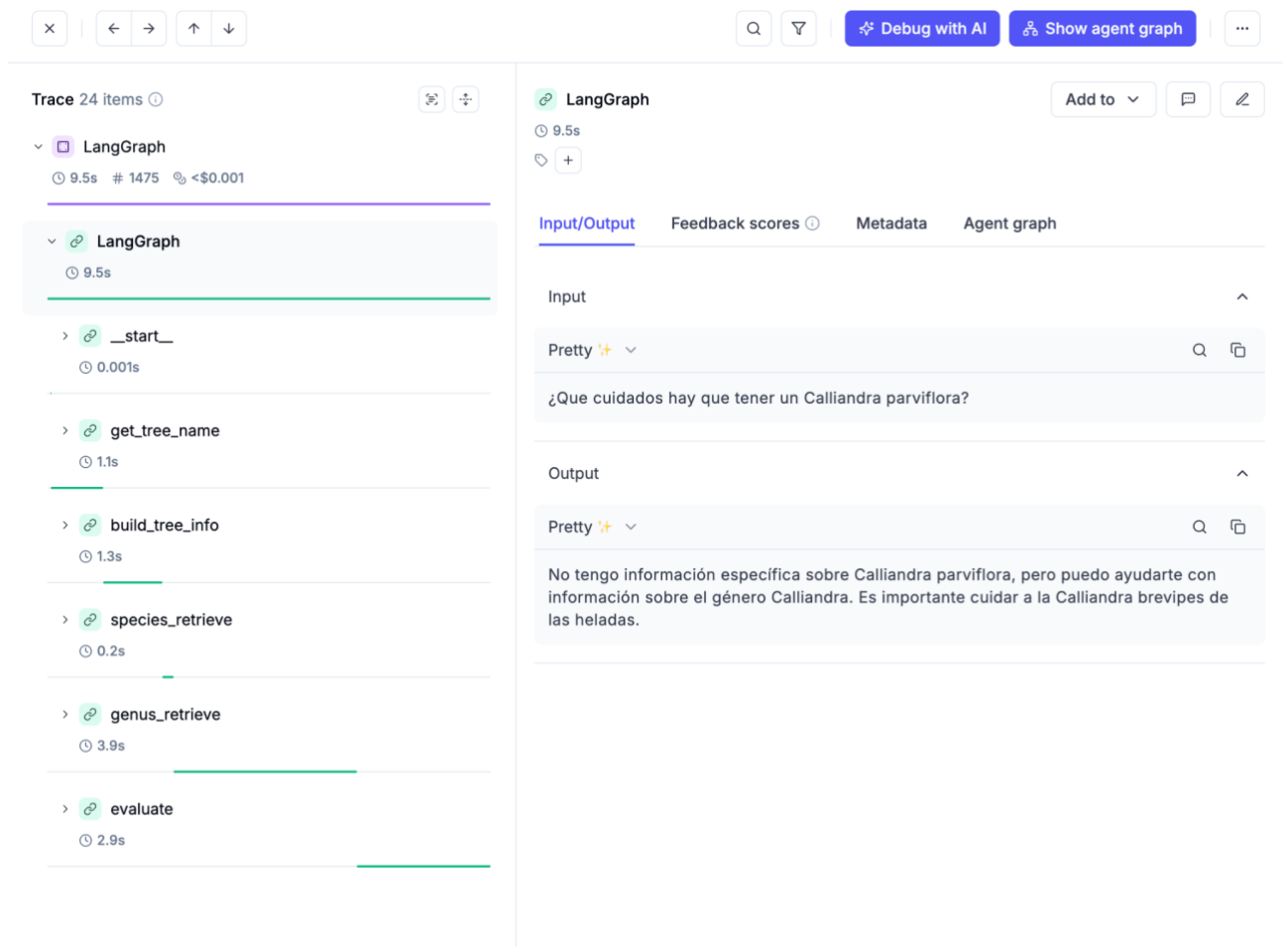


Ilustración 40: Captura de Opik para una ejecución del grafo RAG.

Métricas de evaluación

Con el propósito de evaluar de manera sistemática la calidad del sistema desarrollado, se implementó un *framework* de evaluación basado en casos de prueba con entradas y salidas esperadas. El conjunto de evaluación está compuesto por veintiocho casos de prueba que abarcan diversos escenarios, incluyendo consultas sobre especies específicas con información disponible, consultas que requieren recurrir a niveles taxonómicos superiores, variaciones regionales en los nombres comunes de las especies y diferentes aspectos del cuidado de árboles, como riego, sustrato, ubicación, poda y alambrado.

El sistema fue evaluado utilizando las métricas proporcionadas por *Opik*, las cuales permiten cuantificar distintos aspectos de la recuperación y generación de información. Entre las métricas seleccionadas se incluyen *Context Precision*, que evalúa si los documentos recuperados son relevantes para la consulta del usuario, midiendo la precisión de la recuperación, donde valores cercanos a uno indican alta precisión; *Context Recall*, que determina si el sistema recuperó toda la información necesaria para responder adecuadamente la pregunta, siendo valores próximos a uno indicativos de buena cobertura contextual; y *Hallucination*, que detecta si el modelo genera información no respaldada por el contexto recuperado. Esta última métrica resulta crítica para garantizar que las respuestas se fundamenten estrictamente en la base de conocimiento, y valores cercanos a cero son los más deseables, constituyendo la principal prioridad del equipo.

La Ilustración 41 presenta los resultados de la última evaluación ejecutada sobre el conjunto de pruebas, donde se observa una mejora significativa en la precisión contextual y una marcada reducción en el nivel de alucinaciones.






 Feedback score	 surviving_gerbil_3018
 context_precision_metric	0.745454545
 context_recall_metric	0.714285714
 hallucination_metric	0.04

Ilustración 41: Métricas de la última evaluación

Por último, los resultados detallados de las evaluaciones, incluyendo los valores obtenidos para cada métrica y los casos de prueba ejecutados, se incluyen en el Anexo 13.7.

Inspección y análisis de performance

La integración con *Opik* permite inspeccionar cada nodo del grafo y evaluar la performance de cada estrategia de recuperación, lo que resulta especialmente valioso en el contexto del sistema jerárquico implementado. Entre los aspectos analizados se incluyen la tasa de éxito por nivel taxonómico, que indica con qué frecuencia cada nivel produce una respuesta satisfactoria; los patrones de *fallback*, que muestran cuántas consultas requieren descender en la jerarquía y en qué nivel se encuentra típicamente información relevante; y la calidad de las respuestas por nivel, evaluando si las respuestas generadas a partir de información de niveles superiores mantienen un nivel de utilidad aceptable para el usuario.

Esta capacidad de observabilidad no solo facilita el *debugging* y la identificación de problemas, sino que también proporciona *insights* para optimizar el sistema. Por ejemplo, permite identificar especies o géneros con información insuficiente en la base de conocimientos, detectar preguntas frecuentes que no se responden adecuadamente, y evaluar si el criterio de evaluación de calidad de respuestas es apropiado o requiere ajustes.

6.4.3. Frontend

La Web-App NicheAI se construye sobre una arquitectura *React/Next.js* con un enfoque modular y soporte *multi-tenant*, orientado a garantizar la flexibilidad, mantenibilidad y escalabilidad del sistema, así como una experiencia de usuario optimizada. Este enfoque permite que múltiples organizaciones compartan la misma infraestructura sin comprometer la seguridad ni la personalización visual y funcional de cada entorno.

El diseño de la aplicación incorpora un conjunto de tecnologías contemporáneas que fortalecen su capacidad de adaptación. Entre ellas, destaca el uso de *next-intl*, que habilita la internacionalización con soporte bilingüe (inglés y español), y *next-themes*, que posibilita la personalización dinámica de la interfaz mediante temas configurables. La

autenticación se implementa a través de *JWT tokens* , garantizando un control de acceso seguro y eficiente.

La elección del *framework* Next.js responde a su capacidad para combinar *Server-Side Rendering* (SSR), *Static Site Generation* (SSG) e *Incremental Static Regeneration* (ISR), lo que optimiza tanto el rendimiento como el posicionamiento SEO y la percepción de velocidad por parte del usuario. Estas características, junto con la incorporación de *React* y *Radix UI*, facilitan el desarrollo de componentes reutilizables y accesibles, mientras que la estructura predecible de carpetas y el tipado estricto de *TypeScript* favorecen la mantenibilidad y la escalabilidad del código.

El despliegue de la aplicación se realiza en *Vercel*, una decisión tomada en función de los requerimientos del cliente y las ventajas que ofrece esta plataforma. *Vercel* permite una entrega continua y automatizada mediante CI/CD con *Git* y escalado bajo demanda, además de integrar mecanismos de monitoreo que facilitan la observabilidad y el control operativo.

Finalmente, la integración con *Sprig* permite un seguimiento automático de sesiones, eventos y encuestas de satisfacción, posibilitando la medición continua de la calidad de las respuestas generadas por la IA y de la percepción del usuario respecto a la plataforma. En particular, se implementa un mecanismo de encuestas dinámicas que se activan luego de que el usuario recibe una respuesta por parte del sistema de inteligencia artificial, ver Ilustración 42. Estas encuestas son disparadas por eventos registrados desde la Web-App mediante la API de *Sprig*, lo que permite obtener retroalimentación contextual directamente vinculada con la interacción reciente del usuario. La información recopilada es procesada por el ecosistema de *Sprig* y transformada en métricas accionables que reflejan niveles de satisfacción, puntos de fricción y oportunidades de mejora en la experiencia general. De esta forma, los datos obtenidos alimentan un proceso de mejora continua orientado a la optimización de la usabilidad, la efectividad de las respuestas generadas y la toma de decisiones basada en evidencia empírica.

¿Qué tan precisa fue la información sobre bonsái en la respuesta de IA?

Evalúa si los datos técnicos, consejos de cuidado y detalles sobre especies fueron correctos.

1 2 3 4 5

Muy inexacta Muy precisa

Powered by Sprig

Ilustración 42: Encuesta realizada a través de Sprig luego de obtener una respuesta de IA.

6.5. Infraestructura como Código

La infraestructura de la plataforma fue implementada siguiendo el paradigma de *Infrastructure as Code* (IaC), lo que permitió definir, versionar y desplegar todos los componentes de manera declarativa y automatizada. Este enfoque garantiza la reproducibilidad, trazabilidad y mantenibilidad de los entornos, atributos esenciales en proyectos donde la escalabilidad y la portabilidad son requisitos clave de calidad.

Para la implementación de IaC se seleccionó *Terraform* como herramienta principal, debido a su compatibilidad nativa con múltiples proveedores cloud, su lenguaje declarativo HCL (*HashiCorp Configuration Language*) de fácil lectura y su amplia comunidad de soporte. Además, *Terraform* ofrece una gestión robusta del estado de la infraestructura mediante archivos remotos y mecanismos de bloqueo, lo que facilita el trabajo colaborativo y la aplicación controlada de cambios. La elección de esta herramienta también respondió a la necesidad de que, en etapas futuras, el cliente pudiera desplegar la infraestructura en su propia cuenta de AWS sin requerir una reconfiguración manual, aprovechando la modularidad y reutilización del código *Terraform*.

Inicialmente se evaluó la posibilidad de mantener dos entornos diferenciados, uno de *staging* y otro de producción, con estados de *Terraform* separados, permitiendo pruebas previas antes de aplicar cambios en el entorno final. Sin embargo, tras analizar los costos asociados al mantenimiento de infraestructura duplicada (como *NAT Gateways*, balanceadores y bases de datos RDS *multi-AZ*), se decidió conservar un único entorno configurado con parámetros equivalentes a un ambiente productivo. Esta decisión

permitió optimizar los costos sin comprometer la estabilidad ni la capacidad de escalado del sistema.

Aunque la infraestructura diseñada podría considerarse más compleja que lo necesario para un MVP, la decisión respondió a una visión a largo plazo. El proyecto se concibió desde el inicio con una orientación hacia una red interactiva y escalable, por lo que se previó un eventual aumento en la carga de usuarios y solicitudes. En este sentido, la arquitectura se diseñó cumpliendo con el requisito no funcional RNF10 de Escalabilidad, asegurando que la plataforma pudiera adaptarse al crecimiento sin degradar su rendimiento. Implementar la infraestructura bajo un esquema modular y desplegarla como código no sólo permitió planificar el crecimiento del sistema, sino también ofrecer una base sólida para su futura extensión o migración a nuevos entornos.

El entorno actual se ejecuta sobre AWS, proveedor seleccionado tras un análisis comparativo de costos y características frente a alternativas como *Google Cloud Platform* y *Microsoft Azure*. El estudio, posteriormente presentado al cliente, evidenció que AWS ofrecía una mejor relación costo-beneficio para la carga esperada del sistema, además de contar con servicios nativos integrables (como *ECS Fargate*, *RDS* y *Secrets Manager*) que simplifican la operación y mejoran la seguridad.

El uso de IaC resultó especialmente ventajoso para la agilidad operativa. Por ejemplo, durante la fase de pruebas se detectó que el contenedor de la *API Python* (RAG-API) requería más memoria de la inicialmente asignada. Gracias a la configuración declarativa, el ajuste fue tan simple como modificar el valor correspondiente en las variables del entorno *Terraform* y aplicar el cambio con un solo comando (*terraform apply*), sin necesidad de intervención manual sobre los servicios de AWS. Este tipo de flexibilidad refuerza el atributo de calidad de modificabilidad, facilitando iteraciones rápidas sin comprometer la estabilidad del sistema.

7. Gestión del proyecto

Este capítulo describe la planificación, organización y control del proyecto. Se presenta el cronograma con sus hitos, la metodología de trabajo y la transición de *Scrum* a *Kanban*. También se detalla la gestión de la comunicación y las métricas utilizadas para monitorear el rendimiento del equipo y la predictibilidad de las entregas. En conjunto, el enfoque aplicado fue ágil y orientado a la entrega continua de valor, la reducción de riesgos y la toma de decisiones basadas en evidencia.

7.1. Cronograma e hitos

El cronograma del proyecto se construyó como una herramienta central de planificación y seguimiento, con el objetivo de garantizar la correcta distribución del trabajo y el cumplimiento de los plazos establecidos. A medida que el proyecto avanzó, las fechas y tareas fueron ajustándose progresivamente, para reflejar con mayor fidelidad el estado real del proyecto y adaptarse a los cambios surgidos durante la ejecución.

El cronograma constituyó uno de los instrumentos más importantes dentro de la gestión del proyecto, ya que permitió al equipo coordinar actividades, monitorear avances y realizar ajustes basados en evidencia y retroalimentación constante.

La primera versión fue elaborada al inicio del proyecto y representó una planificación de alto nivel basada en supuestos generales del equipo. En esta etapa se definieron las principales fases de trabajo, concepción del proyecto, definición del problema, desarrollo de la solución, pruebas finales y cierre, estableciendo fechas orientativas para los hitos principales, como el MVP y la entrega final. Si bien esta planificación inicial cumplió su propósito de brindar una visión global del proceso, fue concebida como un punto de partida flexible, sujeto a revisión continua.

A medida que el proyecto avanzó, surgieron ajustes derivados de la retroalimentación obtenida tanto durante la primera revisión como por parte del cliente. Esto dio lugar a la segunda versión del cronograma, en la que se mantuvieron los plazos generales, pero se incorporaron nuevas actividades de gestión y control, incluyendo la definición formal de la metodología ágil de trabajo, la centralización de la documentación, la ampliación del plan de riesgos y la introducción de prácticas de aseguramiento de la calidad (SQA).

Además, se añadieron actividades de *discovery* y validación con usuarios, reforzando el enfoque iterativo y de aprendizaje continuo adoptado por el equipo.

Finalmente, la tercera versión se consolidó como la guía definitiva del proyecto. En ella se ajustaron las fechas en función de los hitos ya alcanzados y se reestructuraron las etapas de desarrollo, redefiniendo los intervalos temporales y desglosando nuevas subfases vinculadas a la evolución del producto. Esta versión incorporó pruebas finales de usabilidad con usuarios, ajustes de calidad y métricas de evaluación, consolidando un cronograma más realista y alineado con el progreso técnico alcanzado.

Cada versión del cronograma se elaboró sobre la base de la anterior, incorporando ajustes y nuevas actividades según las lecciones aprendidas y la evolución del proyecto.

En la siguiente figura se presenta la versión final del cronograma simplificado, mientras que las todas las versiones completas se incluyen en el Anexo 13.8. Dichas versiones reflejan la evolución del plan inicial y los ajustes realizados a partir de la retroalimentación del cliente y del tutor académico.

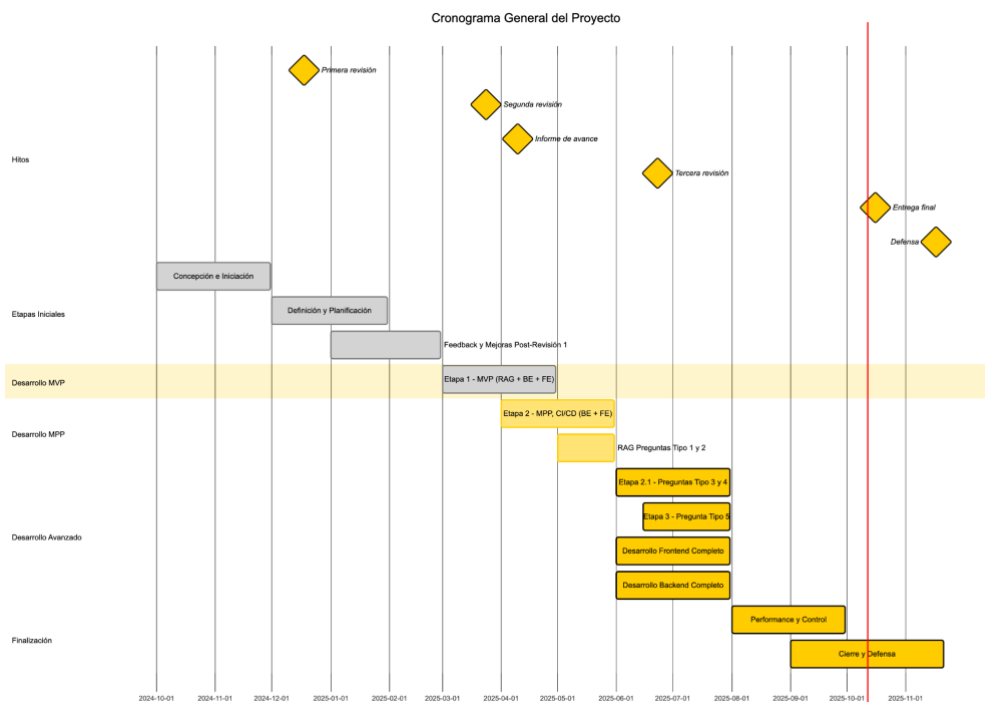


Ilustración 43: Cronograma final del proyecto simplificado

El diagrama muestra la planificación global del proyecto desde octubre de 2024 hasta octubre de 2025, con las principales etapas de desarrollo, validación y entrega.

La evolución del cronograma evidencia la adopción de un enfoque ágil, basado en la retroalimentación constante y la adaptación progresiva, que permitió al equipo mantener el control sobre los plazos y asegurar la coherencia entre la planificación y la ejecución real del proyecto.

7.2. Metodología de trabajo

La gestión del proyecto se apoyó en metodologías ágiles, seleccionadas y adaptadas en función de las características del equipo y la naturaleza del producto. El objetivo fue mantener un flujo de trabajo eficiente, flexible y orientado a la entrega de valor continuo, garantizando al mismo tiempo una comunicación fluida y la trazabilidad de los avances técnicos y estratégicos.

Durante los primeros seis meses del proyecto (octubre de 2024 a marzo de 2025), se aplicó una adaptación del marco *Scrum*, lo que permitió organizar el desarrollo en iteraciones de corta duración, obtener retroalimentación temprana y validar rápidamente las decisiones técnicas con el cliente.

Este enfoque iterativo resultó clave para construir las bases del sistema, consolidar la arquitectura general, implementar el prototipo inicial (MVP) y establecer un ritmo de trabajo predecible y sostenible.

Sin embargo, cuando el proyecto avanzó hacia fases más experimentales, particularmente durante el desarrollo del módulo de inteligencia artificial (a partir de febrero de 2025), la dinámica de *Scrum* comenzó a mostrar ciertas limitaciones. La naturaleza exploratoria de esta etapa requería un esquema más flexible y menos dependiente de entregas cerradas en intervalos fijos.

Por ese motivo, en abril de 2025 el equipo decidió migrar progresivamente hacia *Kanban*, manteniendo los principios ágiles de transparencia y mejora continua, pero reemplazando la estructura de sprints por un flujo de trabajo continuo y visual. Esta transición permitió adaptar el proceso a la variabilidad propia del trabajo con IA, donde las tareas no siempre podían estimarse con precisión, y mejorar la capacidad de respuestas ante descubrimientos o cambios técnicos. De esta forma, el proceso de desarrollo se adaptó mejor a los ciclos de investigación y validación técnica que exigía el proyecto.

7.2.1. Adaptación de *SCRUM*

Durante los primeros seis meses del proyecto, el equipo trabajó bajo una adaptación del marco ágil *Scrum*, que proporcionó estructura y ritmo en las etapas iniciales de desarrollo.

Se establecieron *sprints* de dos semanas, iniciando y finalizando los días jueves, y se planificaban en base a los objetivos definidos junto al cliente. Todas las tareas fueron gestionadas en *Linear*, herramienta que centralizó el *backlog*, permitió el seguimiento de métricas y facilitó la colaboración entre los miembros del equipo.

Cada lunes, de manera asincrónica, el equipo comunicaba el estado actual de sus tareas, y se definían prioridades para la semana. Los jueves se realizaban reuniones de sincronización, destinadas a revisar avances, discutir bloqueos y coordinar los próximos pasos. Esta modalidad reemplazó la dinámica tradicional de *daily meetings*, adaptándose mejor a la disponibilidad del equipo y evitando reuniones innecesarias sin perder comunicación ni trazabilidad.

Al cierre de cada *sprint* se llevaba a cabo una retrospectiva, espacio que permitió detectar bloqueos recurrentes, ajustar estimaciones y mejorar la coordinación del equipo. En la Ilustración 44 se pueden observar las acciones correctivas obtenidas de una de las reuniones de retrospectiva.

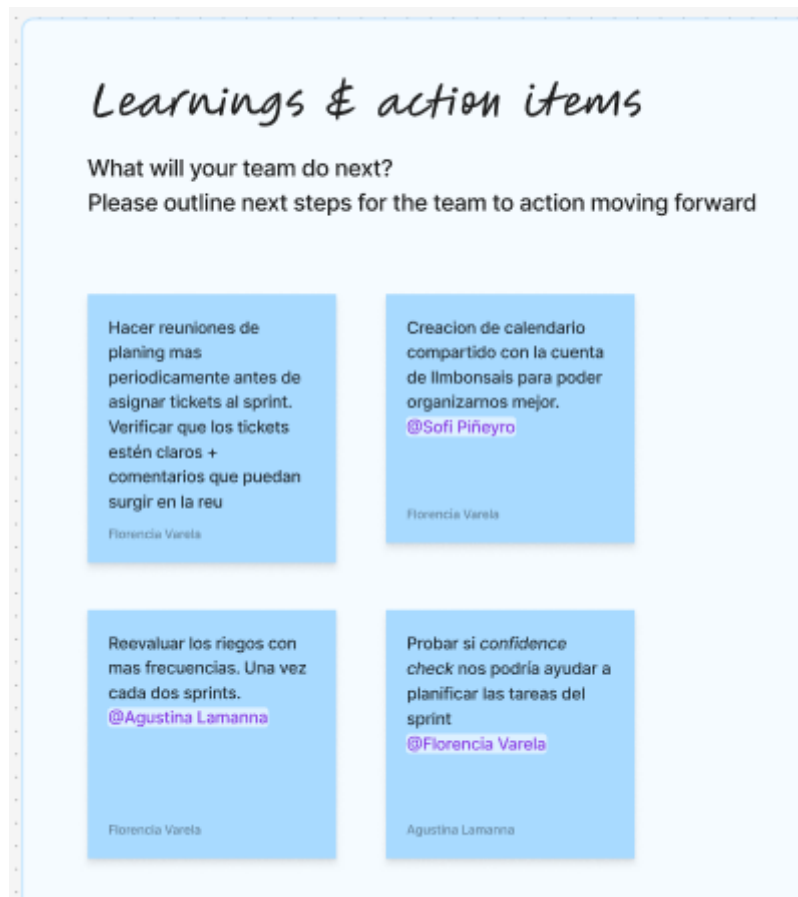


Ilustración 44: Action items resultados de la Retrospectiva

Las tareas de investigación y pruebas de concepto se trabajaron con un *timeboxed* de dos semanas, lo que evitó que la exploración técnica se extendiera indefinidamente y afectara la productividad global.

La relación con el cliente también se integró al proceso ágil. Cada dos semanas se realizaba una reunión de revisión, en la cual se presentaban los resultados alcanzados y se recibía retroalimentación directa. Este intercambio contribuyó a mantener una alineación constante entre las expectativas del cliente y las decisiones técnicas del equipo.

En cuanto a los roles, se adoptaron con flexibilidad para ajustarse al contexto del proyecto. Darío, en su rol de *Product Owner*, definió la dirección general del proyecto y validó las decisiones técnicas de alto nivel; Sofía asumió el rol de *Product Owner interno*, que fue necesario para actuar como nexo directo con el cliente, traduciendo sus requerimientos en tareas concretas y priorizando el *backlog* según el valor de negocio; y Florencia desempeñó el rol de *Scrum Master*, facilitando las dinámicas del equipo y ayudando a remover obstáculos que afectarían el flujo de trabajo.

A diferencia del *Scrum* tradicional, todos los integrantes del equipo asumieron también el rol de desarrolladores. Aunque esta decisión no se alinea estrictamente con lo establecido por *Scrum*, que propone que ni el *Scrum Master* ni el *Product Owner* participen directamente del desarrollo, se consideró adecuada para maximizar la eficiencia y garantizar una distribución equitativa de la carga de trabajo, especialmente teniendo en cuenta el tamaño reducido del equipo y la amplitud técnica del proyecto.

Gracias a esta estructura, el equipo logró una dinámica de colaboración fluida, una comunicación constante y un proceso iterativo de mejora continua, donde cada entrega aportó un incremento tangible al producto y fortaleció la relación con el cliente

Resultados de la etapa *Scrum*

Durante esta etapa, el equipo logró desarrollar la arquitectura base del sistema y los primeros módulos del producto, así como implementar un MVP funcional validado con el cliente. A su vez, esta metodología ayudó al equipo a estabilizar un ritmo de trabajo constante, ya que los sprints fueron completados en tiempo y forma. Por último, facilitó la consolidación de la comunicación interna y la trazabilidad de las decisiones.

7.2.2. Adaptación de *Kanban*

Tras aproximadamente seis meses de trabajo con *Scrum*, y con el módulo de IA siendo desarrollado, el equipo identificó que el marco comenzaba a limitar la flexibilidad necesaria para las tareas de investigación y experimentación con IA.

El problema principal radicaba en que los ciclos de desarrollo para los requerimientos de IA eran difíciles de estimar debido a la falta de conocimiento previo en el dominio, y frecuentemente excedían la duración de un *sprint* individual. Las tareas de investigación y experimentación con diferentes arquitecturas RAG, y optimización de modelos requerían tiempos variables e impredecibles que no se ajustaban a la estructura rígida de *sprints* fijos.

Por este motivo, en abril de 2025 se inició una transición progresiva hacia *Kanban*, que permitió un flujo continuo de trabajo y mayor adaptabilidad a la naturaleza exploratoria del desarrollo de IA, manteniéndose hasta el final del proyecto (octubre de 2025).

En esta nueva etapa, se mantuvo el uso de *Linear* como herramienta principal, configurando un tablero visual con las columnas *Backlog*, *To Do*, *In Progress*, *In Code Review*, *Done*. Este tablero permitió visualizar en tiempo real el estado de cada tarea y priorizar el trabajo de forma dinámica. Además, se establecieron límites de trabajo en progreso (WIP), fijando un máximo de dos tareas simultáneas por persona, con el fin de evitar la sobrecarga y mantener el foco en la finalización de las actividades iniciadas.

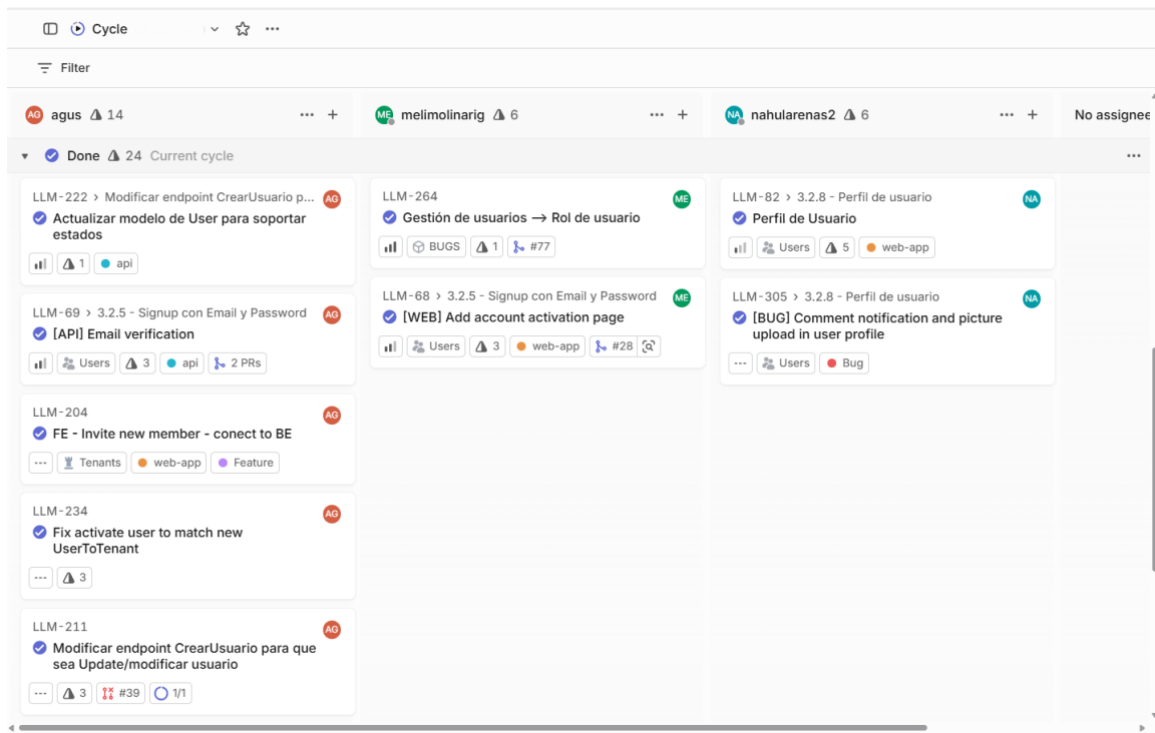


Ilustración 45: Organización del ciclo de desarrollo y asignación de tareas por integrante en Linear

Asimismo, se realizaron revisiones periódicas del tablero y del flujo de trabajo, tanto cuando el equipo lo consideraba necesario como cada dos semanas como máximo. Estas instancias funcionaban como retrospectivas flexibles y permitían redefinir prioridades en base a lo discutido con el cliente y las necesidades del equipo para cumplir los hitos definidos para el producto. A su vez, en reiteradas ocasiones nos permitieron detectar que habían tareas que se podían redefinir con criterios más claros de finalización, reduciendo así el tiempo de ciclo en los periodos siguientes. En particular, ocurrió con la definición del flujo de invitación de usuarios en donde se debieron realizar reuniones del equipo en donde se realizaron tareas de refinamiento adicionales sobre las especificaciones de la historia de usuario correspondiente ya que no quedaba claro para el desarrollador cómo debía implementar dicha funcionalidad.

Una vez completada una épica, el equipo evaluaba y priorizaba la siguiente, garantizando la continuidad del flujo de valor. Este enfoque permitió una entrega continua, sin necesidad de reiniciar ciclos formales, y una mayor capacidad de respuesta ante descubrimientos o cambios en los requerimientos.

Resultados de la etapa *Kanban*

La adopción de *Kanban* generó mejoras significativas en la organización y ejecución del trabajo, especialmente en las tareas vinculadas al desarrollo del sistema de IA. Este enfoque aportó mayor flexibilidad para adaptarse al ritmo variable de experimentación y facilitó la reducción de bloqueos gracias al seguimiento visual del WIP. Asimismo, se observó un aumento sostenido del throughput, alcanzando sus valores más altos hacia el final del proyecto, como se detalla más adelante. Además, permitió la entrega continua de valor y validación temprana, sin interrupciones entre iteraciones.

Además, la implementación de *Kanban* permitió mantener una entrega continua de valor y una validación temprana de los avances, evitando interrupciones entre iteraciones. En conjunto, la transición a este marco consolidó un modelo de trabajo visual, colaborativo y adaptativo, centrado en la optimización del flujo y la eliminación de cuellos de botella, lo que resultó especialmente beneficioso durante las fases más experimentales del proyecto.

7.3. Métricas de gestión

Como parte del seguimiento del proyecto, se definieron y aplicaron un conjunto de métricas de gestión destinadas a evaluar la capacidad real del equipo, identificar posibles cuellos de botella y mejorar la predictibilidad en la planificación.

Estas métricas evolucionaron a lo largo del proyecto junto con el cambio metodológico: durante la primera etapa, en la que se utilizó *Scrum*, se hizo foco en la planificación y el cumplimiento de objetivos de *sprint*; posteriormente, al migrar hacia *Kanban*, las métricas se orientaron al flujo continuo y la eficiencia del proceso.

El objetivo principal de su seguimiento fue validar la estabilidad del proceso de trabajo y confirmar que el ritmo de entrega alcanzado representara la máxima capacidad efectiva del equipo, sin comprometer la calidad.

7.3.1. Métricas en la etapa *Scrum*

Las métricas comenzaron a registrarse de forma sistemática a partir de enero de 2025, cuando el equipo ya contaba con tareas estimadas y un flujo de trabajo más estable.. Este seguimiento se mantuvo hasta abril, abarcando un total de ocho sprints completos.

Durante esta etapa, en la que se utilizó *Scrum* como marco principal de gestión, se priorizaron métricas orientadas a la planificación y al control del cumplimiento de los compromisos asumidos en cada sprint.

Además, las actividades de investigación o *spikes* se midieron de manera *timeboxed*, es decir, con una duración máxima predefinida. En los casos en que el tiempo finalizaba sin alcanzar una conclusión completa, se documentaban los hallazgos parciales y se evaluaba junto con el cliente si correspondía continuar la investigación. Cuando era necesario profundizar, se creaba un nuevo *spike* para el siguiente sprint, con un objetivo más acotado y específico.

Velocidad

Representa la cantidad promedio de puntos de historia completados por *sprint*. En este proyecto, su seguimiento permitió observar la evolución del desempeño a lo largo de las distintas iteraciones y analizar la madurez progresiva del proceso de trabajo.

En la Ilustración 46 se muestra la evolución de la velocidad del equipo desde el *sprint* 5 hasta el *sprint* 12, período en el cual el uso de *Scrum* se encontraba completamente implementado y el tablero de tareas reflejaba de manera consistente el trabajo realizado.

Los primeros cuatro sprints no se incluyeron en la medición, ya que estuvieron orientados principalmente a actividades de organización, definición de procesos y configuración inicial del proyecto; en esa etapa, las tareas registradas en el *backlog* cumplían más una función de recordatorio que de planificación formal, por lo que sus datos no resultaban representativos.

La tendencia general es claramente ascendente: los primeros *sprints* (5–9) presentan valores bajos e irregulares, mientras que hacia el final del período se observa un incremento sostenido, alcanzando los mayores niveles de productividad en los sprints 11 y 12.

Velocidad por Sprint

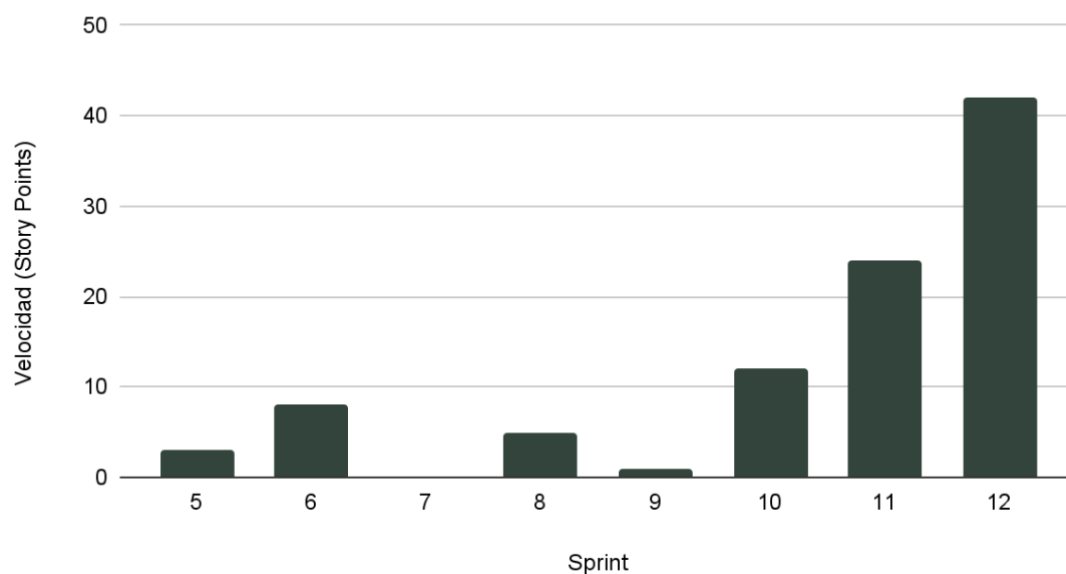


Ilustración 46: Velocidad del equipo por sprint.

La gráfica evidencia una tendencia ascendente en la capacidad de entrega, con valores bajos e irregulares en los primeros sprints debido al trabajo exploratorio y la calibración de estimaciones, y un crecimiento sostenido hacia el final, reflejo de la madurez metodológica y la estabilización del flujo de trabajo.

Esta variabilidad inicial responde a varios factores propios de la etapa de conformación del equipo y del carácter exploratorio del proyecto. En primer lugar, durante los primeros *sprints* aún se estaba ajustando el proceso de estimación y planificación, por lo que los puntos de historia no reflejaban con precisión el esfuerzo real. A esto se suma que gran parte del trabajo inicial consistió en investigación y validación técnica, especialmente vinculada al módulo de inteligencia artificial, actividades que, aunque demandaban un alto esfuerzo, no siempre concluían con entregables tangibles dentro del *sprint*.

En segundo lugar, la mayor parte del esfuerzo del equipo se concentraba en tareas de investigación. Debido a su carácter exploratorio y *timeboxed*, estas actividades no siempre alcanzaban una conclusión definitiva dentro del *sprint*. Como ya se mencionó previamente, en los casos en que el tiempo asignado finalizaba sin resultados concluyentes, se documentaban los avances parciales y se evaluaba, junto con el cliente, la necesidad de continuar en una nueva iteración con un objetivo más acotado.

Como estas tareas no se marcaban formalmente como “*Done*” en el tablero, su esfuerzo no quedaba reflejado en la métrica de velocidad, lo que explica los valores bajos registrados en las primeras iteraciones.

Adicionalmente, en los *sprints* 7 y 9 la velocidad descendió notablemente debido a licencias personales y períodos de exámenes académicos, que redujeron temporalmente la disponibilidad del equipo. A pesar de ello, se mantuvo el trabajo en tareas de documentación, asegurando la continuidad del proyecto.

A partir del *sprint* 10, con el equipo completamente operativo y una mayor claridad en la definición de tareas, la velocidad comenzó a aumentar de forma constante. En esta etapa, el flujo de trabajo se estabilizó, las estimaciones fueron más precisas y el esfuerzo se orientó al desarrollo de funcionalidades completas y entregables verificables.

Sin embargo, al finalizar el ciclo 12, el análisis de la métrica de velocidad permitió identificar un patrón relevante: las tareas que habían reducido el rendimiento del equipo en los *sprints* 7 y 9 correspondían principalmente al desarrollo del módulo RAG. Este hallazgo evidenció que el marco *Scrum* no lograba reflejar de manera fiel el progreso real del equipo, ya que muchas de estas tareas implicaban avances incrementales y exploratorios que no se traducían en entregables cerrados por *sprint*.

A partir de esta observación, se decidió migrar hacia *Kanban*, una metodología que permitió representar con mayor precisión el flujo continuo de trabajo, visualizar los avances parciales de las tareas de investigación y mantener un seguimiento más alineado con la naturaleza iterativa del desarrollo.

Cycle Time

Corresponde al tiempo transcurrido desde que una tarea se inicia hasta que se completa dentro del *sprint*. Su seguimiento permitió evaluar la eficiencia interna del flujo de trabajo y detectar posibles bloqueos o demoras en las tareas de desarrollo.

En los primeros *sprints* se observaron variaciones notables, especialmente las tareas de investigación. Estas tareas, al gestionarse bajo un esquema *timeboxed*, se cerraban al cumplirse el plazo asignado, pero en muchos casos requerían continuarse en *sprints* posteriores para profundizar la investigación o validar nuevos enfoques. Como

consecuencia, algunas tareas presentaban tiempos de ciclo más prolongados, reflejando la naturaleza iterativa y acumulativa del trabajo de experimentación, más que un problema de estimación o eficiencia.

Este análisis permitió identificar las causas de retraso y ajustar la planificación de las iteraciones, logrando con el tiempo una reducción sostenida en el tiempo promedio de ciclo.

Commitment Reliability

Mide la proporción de tareas comprometidas al inicio del sprint que efectivamente se completan al finalizarlo. Esta métrica permitió evaluar la precisión de las estimaciones y la fiabilidad del cumplimiento del plan de trabajo.

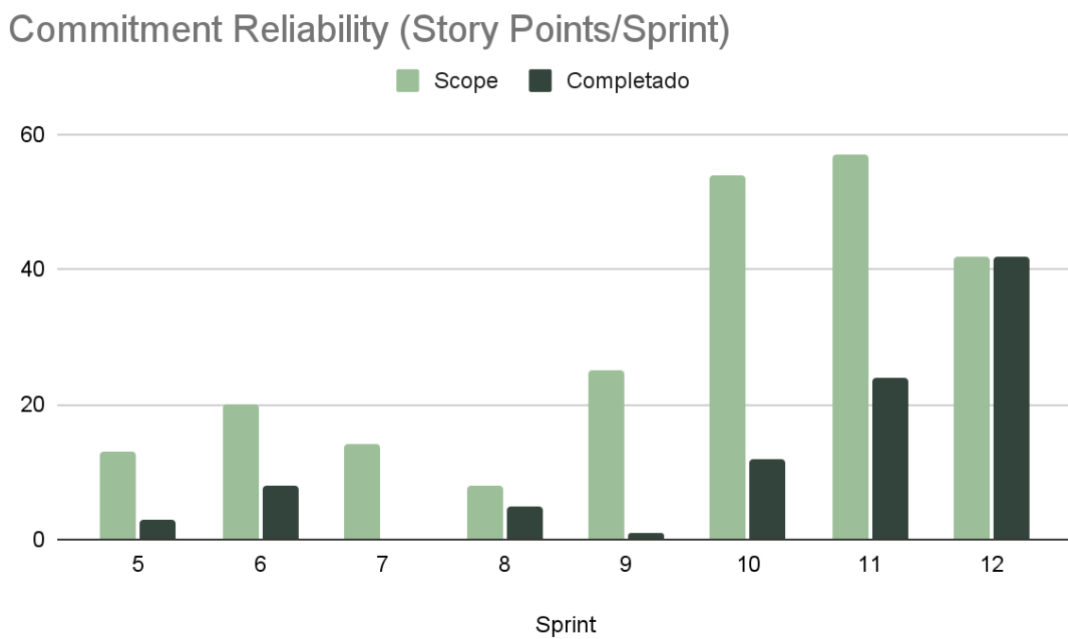


Ilustración 47: Compromiso y cumplimiento por sprint (Commitment Reliability).

La Ilustración 47 muestra la proporción entre las tareas planificadas (*Scope*) y las completadas (*Completed*) durante los sprints 5 al 12. En las primeras iteraciones, la diferencia entre ambas barras es considerable, lo que refleja una baja fiabilidad del compromiso inicial.

Es importante aclarar que durante los primeros *sprints* (5 al 8), la mayor parte del trabajo estuvo compuesta por tareas de investigación (*spikes*), gestionadas bajo un esquema

timeboxed. En estos casos, el *scope* reflejaba las investigaciones planificadas para el *sprint*, mientras que el *completed* correspondía a aquellas que pudieron darse por finalizadas dentro del plazo de dos semanas.

Dado que muchas de estas tareas requerían continuar en iteraciones posteriores, las diferencias entre el trabajo comprometido y el completado no implicaban incumplimiento, sino la naturaleza progresiva y exploratoria del proceso.

A medida que el foco del equipo se desplazó hacia el desarrollo de funcionalidades y entregables concretos, la métrica comenzó a representar con mayor precisión la capacidad real de cumplimiento, mostrando una tendencia clara hacia la estabilización.

A partir del análisis conjunto de la *Commitment Reliability* y el *Cycle Time* durante las retrospectivas, el equipo pudo detectar bloqueos, revisar el tamaño y la definición de las tareas, y mejorar las planificaciones de los *sprints* y la coordinación del trabajo en curso. Esto derivó en la incorporación de las reuniones asincrónicas semanales, que facilitaron la comunicación y la resolución temprana de obstáculos.

A partir de estos ajustes, la fiabilidad de los compromisos aumentó de forma sostenida: las brechas entre tareas planificadas y completadas se redujeron progresivamente, hasta alcanzar valores cercanos al 100 % de cumplimiento hacia los *sprints* finales.

Esto refleja una planificación más realista y una ejecución más predecible, factores que contribuyeron directamente a la estabilidad del flujo y a la mejora continua del proceso.

7.3.2. Métricas en la etapa Kanban

La transición hacia *Kanban* se produjo en abril de 2025, momento en el que el equipo comenzó a medir el desempeño con foco en la eficiencia del flujo de trabajo, la estabilidad del proceso y la capacidad de entrega sostenida.

El seguimiento continuo permitió observar una mejora progresiva en la velocidad y la predictibilidad del trabajo, particularmente desde los primeros ciclos hasta el cierre del proyecto.

Throughput (tareas completadas por ciclo)

El *throughput*, es decir, la cantidad de tareas completadas por ciclo, mostró una evolución claramente ascendente. El promedio general fue de 11,4 tareas por ciclo, con una tendencia creciente hacia el final del proyecto, donde se alcanzaron picos de hasta 19 tareas completadas en un mismo ciclo.

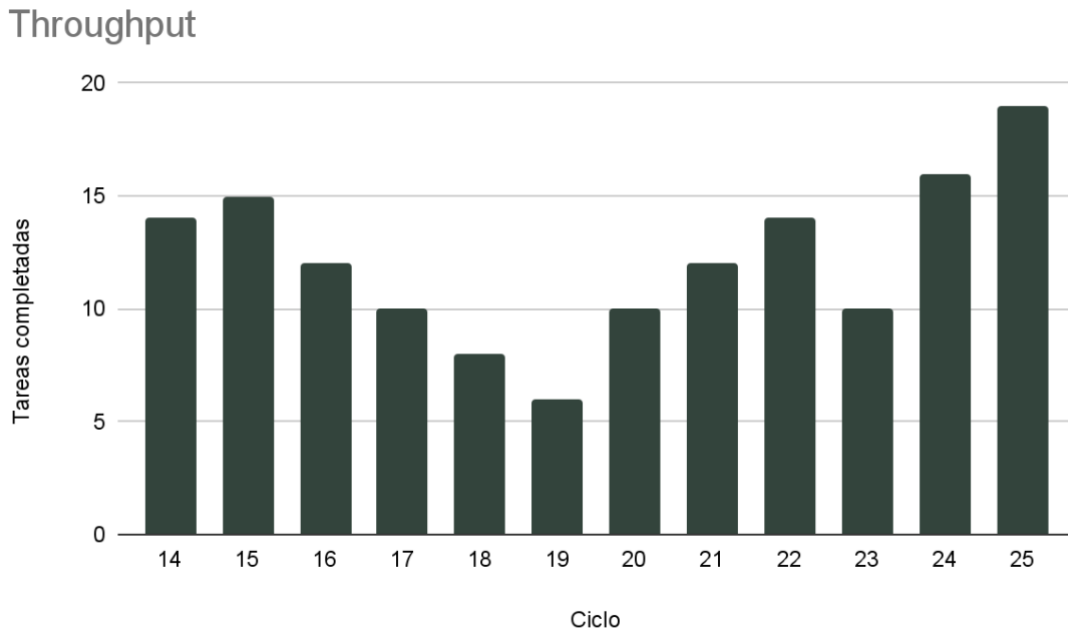


Ilustración 48: Throughput del equipo.

En la Ilustración 48 se aprecia que tras un inicio estable, el equipo experimentó una leve disminución entre los ciclos 16 y 20, seguida de una recuperación sostenida hasta el cierre del proyecto.

Las variaciones observadas se explican principalmente por factores contextuales y técnicos.

Durante los ciclos 16 y 17, el equipo abordó tareas de infraestructura y configuración de CI/CD, actividades de alta complejidad técnica y baja fragmentación, que requerían una dedicación prolongada pero se contabilizaban como pocas tareas en el tablero.

Posteriormente, entre los ciclos 18 y 20, coincidieron períodos de parciales académicos, evaluaciones y licencias personales, lo que redujo temporalmente la capacidad de trabajo del equipo.

A partir del ciclo 21, una vez superado ese período y con la infraestructura ya consolidada, el ritmo de entrega aumentó progresivamente, reflejando una mejora sostenida en la eficiencia y en la fluidez del flujo de trabajo.

Esta métrica se utilizó de forma análoga a la velocidad en la etapa *Scrum*, manteniendo intervalos de dos semanas únicamente como unidad de medición, con el fin de comparar el ritmo de entrega a lo largo del tiempo y asegurar que la productividad no decayera respecto de los valores observados al inicio del proyecto.

El seguimiento de esta métrica funcionó como una herramienta de monitoreo y toma de decisiones. Esto permitió al equipo evaluar de forma continua cuántas tareas se completaban por ciclo, y en función de ello, estimar la carga de trabajo posible para las siguientes iteraciones, considerando la complejidad de las tareas pendientes y los plazos establecidos para cada hito.

Cuando se registraron caídas en la métrica, el equipo analizaba las causas para decidir si era necesario realizar ajustes. En los ciclos 16 y 17, como ya se mencionó, la disminución se atribuyó a tareas de mayor complejidad, por lo que no se tomaron medidas adicionales.

En cambio, durante los ciclos 18 a 20, al observar una reducción pronunciada asociada a evaluaciones académicas y licencias, el equipo replanificó su trabajo. En el ciclo 19, tras identificar que el *throughput* había alcanzado su punto más bajo, se implementaron ajustes organizativos para optimizar el tiempo disponible.

Entre estas medidas se destacó el reemplazo de la reunión sincrónica diaria por actualizaciones asincrónicas, lo que permitió destinar ese espacio a la ejecución de tareas y mejorar el aprovechamiento de las horas efectivas de trabajo.

Gracias a esta reorganización, el equipo logró recuperar el ritmo de entrega en los ciclos siguientes y mantener la continuidad de valor para el cliente, incluso en un período de menor disponibilidad.

Cycle Time (tiempo de ciclo)

El tiempo promedio para completar una tarea fue de aproximadamente 5,8 días de calendario, con una mediana de 5,5 días, considerando todos los ciclos analizados (14 al 25). Estas mediciones no reflejan dedicación continua, sino el tiempo total transcurrido

entre el inicio y la finalización de cada tarea dentro del flujo de trabajo, considerando que el equipo no contaba con una dedicación *full-time*. De esta forma, el *Cycle Time* se interpretó como una medida relativa de flujo y no como una cantidad exacta de horas efectivas invertidas.

En la Ilustración 49 se observa la evolución del *Cycle Time* a lo largo de los ciclos 14 a 25.

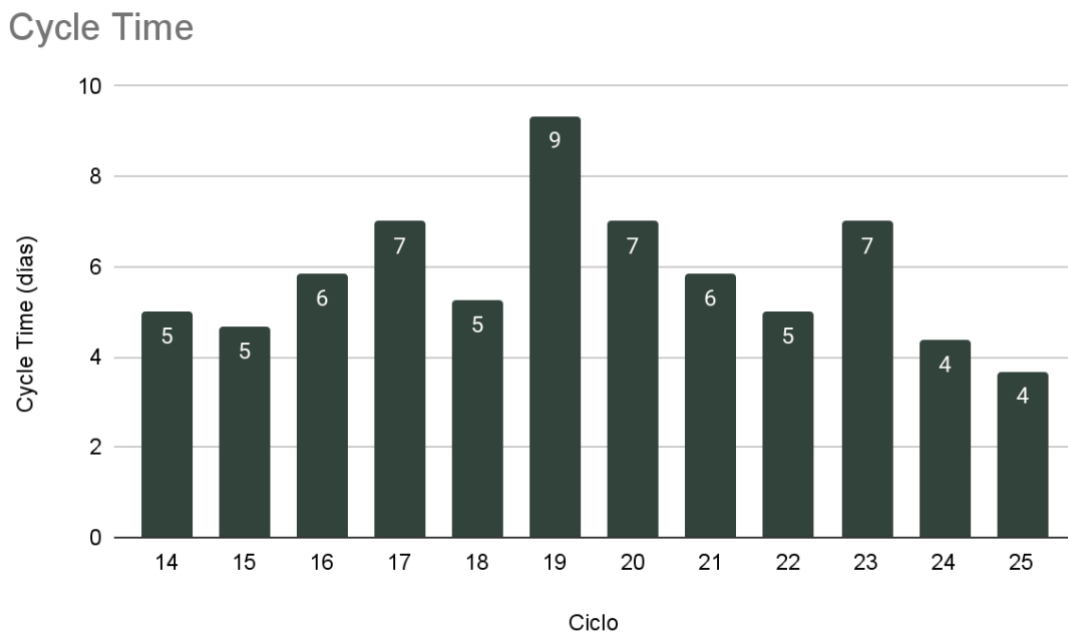


Ilustración 49: Cycle Time a lo largo del proyecto.

Se observa un pico en el ciclo 19, asociado a un período de menor disponibilidad del equipo, seguido por una reducción que evidencia la estabilización y eficiencia del proceso en las últimas iteraciones.

El valor se mantuvo relativamente estable durante gran parte del período, con un pico en el ciclo 19, coincidente con las semanas de exámenes y menor disponibilidad del equipo. Tras este punto, los tiempos de ciclo comenzaron a disminuir progresivamente, alcanzando los valores más bajos en los ciclos finales (24 y 25).

Esta reducción refleja una mayor fluidez en el flujo de trabajo, resultado de la división más granular de tareas, la optimización del tablero *Kanban* y la familiaridad creciente del equipo con el proceso.

Además de reflejar un flujo más ágil y predecible, el seguimiento del *cycle time* resultó fundamental para mejorar la planificación y la comunicación con el cliente. Al conocer los tiempos promedio y máximos de finalización, el equipo pudo estimar con mayor precisión las fechas de entrega y ofrecer información más confiable sobre el estado de avance, fortaleciendo la coordinación entre desarrollo y gestión.

Estos resultados validan un proceso eficiente, sin acumulaciones significativas ni demoras prolongadas, y evidencian la madurez del equipo en la estimación y ejecución de tareas.

Work in Progress (WIP)

El WIP se mantuvo controlado durante todo el período, con un máximo de ocho tareas simultáneas, coincidiendo con el límite establecido por el equipo.

Este valor se definió en función de la capacidad efectiva y la dinámica de trabajo del equipo, buscando equilibrar la carga y evitar la dispersión de esfuerzos.

La decisión surgió a partir de las observaciones realizadas durante las últimas iteraciones de *Scrum*, particularmente en el ciclo 13, momento de transición hacia *Kanban*.

En esa etapa se identificó que, cuando una persona tenía más de dos tareas en progreso, solían aparecer bloqueos o tareas desatendidas: algunas quedaban parcialmente avanzadas y sin atención activa, mientras que otras no eran retomadas por otros integrantes porque ya estaban asignadas y marcadas como iniciadas.

Esta situación generaba una falsa sensación de avance, reducía la visibilidad real del progreso y afectaba la capacidad de respuesta del equipo.

A partir de esa experiencia, se estableció el límite de ocho tareas totales como punto de equilibrio que permitía mantener el foco, detectar bloqueos reales y asegurar que cada tarea iniciada recibiera atención continua hasta su finalización.

Este número también respondió a la estructura operativa del equipo: un subgrupo de dos personas dedicado al módulo de inteligencia artificial, con una tarea por persona, dado que su naturaleza implicaba mayor complejidad y tiempo de ejecución, y otro de tres

integrantes centrado en el desarrollo del producto, con un máximo de dos tareas por persona, lo que llevó naturalmente a un máximo de ocho tareas activas en simultáneo.

Este control permitió evitar la sobrecarga, mantener el enfoque en las tareas activas y garantizar un ritmo sostenible de trabajo.

El seguimiento del WIP también sirvió para definir reglas internas de desarrollo. Por ejemplo, cuando una persona alcanzaba su límite de trabajo en progreso y tenía tareas bloqueadas, no podía iniciar nuevas actividades hasta resolverlas. Esta dinámica fomentó la colaboración entre los integrantes para desbloquear procesos críticos y mantener la fluidez del trabajo.

Asimismo, esta métrica llevó al equipo a decidir que cada funcionalidad fuera desarrollada de punta a punta (*frontend* y *backend*) por la misma persona, asegurando que cada entrega representara una unidad de valor completa y visible para el cliente, en lugar de partes inconexas del sistema.

7.3.3. Distribución de throughput

Durante la etapa de desarrollo, el equipo decidió organizarse en dos subgrupos de trabajo con responsabilidades diferenciadas, con el objetivo de optimizar la eficiencia y aprovechar mejor la especialización adquirida por cada integrante.

Por un lado, dos integrantes se enfocaron en el desarrollo del módulo de inteligencia artificial (IA), mientras que los tres restantes trabajaron en la implementación del resto de la aplicación, incluyendo las capas de *frontend*, *backend* y la integración general del sistema.

Esta división surgió de la necesidad de favorecer la familiarización profunda con cada módulo y reducir la fricción que implicaba el trabajo simultáneo de todos sobre componentes distintos. Al asignar a cada persona la autoría técnica principal de un módulo, el equipo logró aumentar la velocidad de desarrollo y reducir el tiempo necesario para comprender el contexto de cada tarea, lo que derivó en una mejora general del rendimiento.

El módulo de inteligencia artificial presentó características particulares que influyeron en la naturaleza y las métricas de sus tareas. A diferencia de las del producto base, las tareas

de IA solían ser más exploratorias y abiertas, ya que implicaban probar, comparar y validar múltiples enfoques antes de alcanzar la mejor solución posible.

Como consecuencia, el número total de tareas del módulo de IA fue menor, pero la complejidad y duración de cada una fueron mayores en comparación con las tareas del producto. Las métricas asociadas de estas tareas reflejaron tiempos de ejecución más prolongados y menor *throughput*, lo que no implicaba un menor rendimiento, sino la naturaleza distinta del trabajo técnico involucrado.

El equipo tuvo en cuenta estas diferencias en todo momento y las consideró dentro de su análisis de capacidad. Dado que el proyecto combinaba tanto desarrollo de software e investigación aplicada, se entendió que la productividad debía evaluarse no solo en cantidad de tareas, sino también en profundidad de los resultados técnicos obtenidos.

De esta forma, se consideró que el equipo trabajaba a su máxima capacidad efectiva, alcanzando un equilibrio óptimo entre calidad, innovación y velocidad de entrega dentro del contexto de aprendizaje continuo que implicaba el desarrollo del módulo de IA.

Para representar la distribución del *throughput* entre las áreas de desarrollo, se tomaron como referencia los últimos seis ciclos de trabajo, correspondientes al período en el que el equipo había alcanzado su mayor nivel de eficiencia y estabilidad. El *throughput* se midió en función de la cantidad de tareas completadas, al ser la unidad operativa más homogénea entre los dos subgrupos y permitir una comparación consistente de la carga de trabajo relativa dedicada a cada módulo.

Si bien las tareas de inteligencia artificial solían implicar una mayor complejidad técnica y requerir más tiempo individual, el conteo de tareas completadas permitió estimar de manera consistente la proporción de dedicación efectiva entre el desarrollo del módulo de IA y el resto de la aplicación.

Estos ciclos reflejan con mayor precisión la dinámica real de trabajo y la consolidación de la división funcional del equipo, evidenciando un equilibrio sostenible entre investigación y desarrollo de producto.

Distribución del throughput

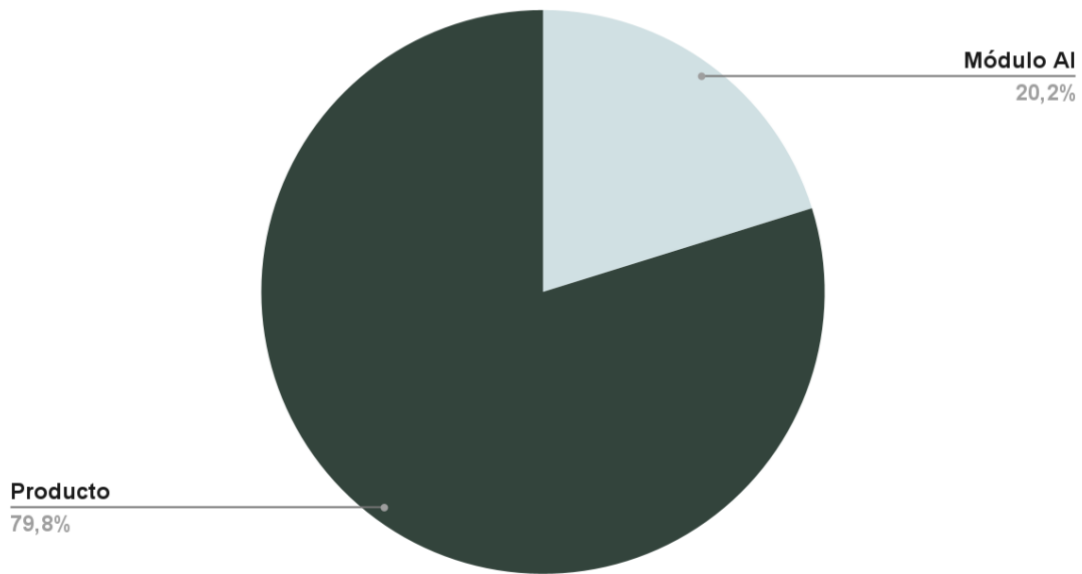


Ilustración 50: Distribución del esfuerzo entre el módulo de inteligencia artificial y el desarrollo del resto de la aplicación.

Estos ciclos reflejan con mayor precisión la dinámica real de trabajo y la consolidación de la división entre el módulo de inteligencia artificial y el desarrollo del resto de la aplicación.

7.3.4. Análisis y aprendizajes

El uso de métricas no fue solo descriptivo sino que activó decisiones concretas de gestión que mejoraron el flujo y la previsibilidad. Los principales aprendizajes, con sus acciones asociadas, fueron las siguientes:

Transición metodológica informada por datos (abril 2025).

La irregularidad en la velocidad de los primeros sprints y la falta de correspondencia entre el progreso real y el medido en el módulo de inteligencia artificial evidenciaron la necesidad de un cambio metodológico. Estos resultados motivaron la transición hacia *Kanban*, que permitió representar con mayor fidelidad el flujo continuo de trabajo y la naturaleza iterativa de las tareas de investigación.

Límite de WIP basado en observación (ciclo 13).

Durante las últimas iteraciones de *Scrum* (particularmente en el ciclo 13), se detectó que cuando un integrante gestionaba más de dos tareas en simultáneo, tendían a generarse bloqueos o tareas desatendidas. A partir de esta observación se estableció un límite de ocho tareas totales, que equilibró la carga de trabajo y garantizó la finalización de las tareas iniciadas antes de incorporar nuevas.

Replanificación ante caída del *throughput* (ciclos 18–20).

La reducción del *throughput* en los ciclos 18 a 20, asociada a períodos de exámenes y licencias, impulsó ajustes organizativos orientados a optimizar el tiempo disponible. Por ejemplo, durante el ciclo 19 se sustituyó la reunión sincrónica diaria por una asincrónica, lo que permitió destinar ese espacio a la ejecución de tareas y mantener la entrega continua de valor.

Decisiones diferenciadas según la causa.

Las caídas observadas en los ciclos 16 y 17 se atribuyeron a tareas de infraestructura y CI/CD, de alta complejidad y baja granularidad. En estos casos no se implementaron cambios, dado que las variaciones respondían a la naturaleza del trabajo y no a una falta de eficiencia, evitando así una fragmentación innecesaria de tareas.

Mejora de la fiabilidad en la planificación.

El análisis conjunto de la *Commitment Reliability* y la velocidad permitió ajustar el tamaño de las tareas y calibrar las estimaciones, reduciendo progresivamente la brecha entre el trabajo planificado y el completado. Hacia los últimos sprints, los valores alcanzaron una fiabilidad cercana al 100 %, reflejando una planificación más precisa y un proceso de ejecución más predecible.

Comunicación más precisa con el cliente.

El conocimiento de los tiempos promedio y máximos de finalización de tareas permitió realizar estimaciones más realistas y establecer compromisos de entrega basados en datos, fortaleciendo la transparencia y la coordinación entre el equipo y el cliente.

En conjunto, el análisis de métricas permitió mejorar la eficiencia del proceso y la calidad de la planificación, consolidando un sistema de gestión basado en evidencia y orientado a la mejora continua. Las métricas, más que un fin en sí mismas, se convirtieron en un reflejo del proceso: mostraron la evolución del equipo y cómo la observación sistemática de datos permitió alcanzar un equilibrio entre velocidad, calidad y sostenibilidad, adaptando progresivamente las prácticas de trabajo a la naturaleza real del producto y a las capacidades efectivas del equipo.

7.4. Gestión de la comunicación

La comunicación fue clave para coordinar al equipo, el cliente y el tutor académico, combinando instancias sincrónicas y asincrónicas según la urgencia y disponibilidad. El uso de distintos canales garantizó un flujo de información transparente, documentado y accesible para futuras consultas.

7.4.1. Comunicación entre el equipo

Durante los primeros meses del proyecto (octubre de 2024 a enero de 2025), la comunicación interna se gestionó principalmente mediante reuniones semanales por *Google Meet*, destinadas a revisar el estado de las tareas y coordinar las acciones a corto plazo. Para la organización y coordinación diaria, se utilizó *WhatsApp*, canal que facilitó el intercambio rápido y la resolución de temas operativos menores.

A medida que el proyecto avanzó y las necesidades técnicas aumentaron, en febrero de 2025, se incorporó *Slack* como espacio central de comunicación relacionado al desarrollo. Esta decisión surgió luego de detectar en las retrospectivas que los intercambios informales en *WhatsApp* no eran fácilmente accesibles, lo que dificultaba el seguimiento de temas técnicos. La adopción de *Slack* permitió estructurar la información en canales temáticos (*backend*, *frontend*, inteligencia artificial y actualizaciones) y favoreció la comunicación asincrónica mediante actualizaciones semanales que se realizaban los días lunes. Las reuniones de equipo se mantuvieron los días jueves, orientadas a la revisión general de avances, la detección de bloqueos y la definición de próximos pasos.

En temas vinculados a la gestión o al diseño, se trabajó de forma colaborativa sobre documentos compartidos y tableros en *Figma*, utilizando los sistemas de comentarios integrados para mantener discusiones ordenadas y contextualizadas. Cuando era

necesario, se notificaba por *WhatsApp* para asegurar que todos los integrantes estuvieran al tanto de los intercambios.

Este esquema permitió equilibrar la comunicación formal e informal, manteniendo la trazabilidad de las decisiones sin perder agilidad en el intercambio cotidiano. Gracias a esta estructura de comunicación, el equipo logró reducir los tiempos de respuesta ante bloqueos técnicos, mejorar la coordinación entre los subequipos y mantener informados al cliente y al tutor sin sobrecargar las reuniones sincrónicas.

7.4.2. Comunicación con el cliente

La relación con el cliente se estructuró a través de reuniones quincenales de actualización, donde se presentaban los avances del proyecto y se recogía retroalimentación sobre los resultados obtenidos. Estas instancias fueron clave para mantener la alineación entre las expectativas del cliente y las decisiones técnicas del equipo.

Además, se creó un grupo de *WhatsApp* compartido con el cliente, utilizado para coordinar reuniones, resolver dudas puntuales o comunicar avances que no requerían una reunión. Este canal permitió una interacción ágil y continua, favoreciendo la disponibilidad y la respuesta rápida ante consultas específicas.

Con el fin de mantener un registro estructurado se documentaron sobre las reuniones los principales temas tratados, las decisiones acordadas y las acciones pendientes. Este material sirvió como fuente de consulta y permitió evitar repeticiones o la pérdida de información relevante.

7.4.3. Comunicación con el tutor

La comunicación con el tutor académico siguió una dinámica similar a la establecida con el cliente. Se realizaron reuniones periódicas de seguimiento, orientadas a discutir los avances técnicos y metodológicos, revisar entregables y recibir orientación sobre aspectos académicos del proyecto.

En los casos en que la naturaleza de la consulta lo permitía, la comunicación se realizó a través de *WhatsApp*, lo que facilitó la resolución ágil de dudas menores sin necesidad de coordinar encuentros formales.

Esta modalidad flexible permitió mantener una interacción constante con el tutor, asegurando la alineación del proyecto con los lineamientos académicos y fomentando un acompañamiento continuo a lo largo de todas las etapas del desarrollo.

8. Gestión de riesgos

En este capítulo se aborda la gestión de riesgos, comenzando por la definición del concepto de riesgo y su relevancia en el desarrollo de *software*. A partir de esta base conceptual, se expone la estrategia de gestión de riesgos adoptada para este proyecto y, posteriormente, se detallan los principales riesgos identificados, junto con sus respectivas estrategias de identificación, análisis cuantitativo, mitigación y seguimiento.

Un riesgo, según lo plantea el autor Charette, “es el potencial de que se materialicen las consecuencias negativas e indeseadas de un evento.” [36]. Para el equipo, esta definición permite comprender el riesgo como la posibilidad de que un evento genere consecuencias negativas o no deseadas, en este caso sobre el proyecto.

8.1. Estrategia de gestión de riesgos

Considerando la definición de riesgo adoptada por el equipo, se decidió implementar un enfoque proactivo de gestión de los mismos. Aspirando a minimizar la incertidumbre que pudiera afectar el desarrollo, buscando maximizar la eficiencia del equipo al momento de tomar decisiones.

Dicha gestión consistió en la identificación, evaluación y seguimiento recurrente de los riesgos a lo largo de todo el ciclo de vida del proyecto, siguiendo los principios de gestión de riesgos planteada por el autor Charette en el libro *Software Engineering Risk Analysis and Management* [36]. El objetivo fue evitar un enfoque reactivo, integrando la gestión de riesgos al proceso de trabajo habitual, buscando fomentar así una cultura de anticipación y adaptación frente a posibles eventos adversos.

8.2. Identificación

Con el fin de identificar los riesgos, el equipo utilizó la técnica de *brainstorming* para poder evaluar en conjunto los principales elementos del proyecto en dónde se podrían generar eventos que pudieran tener un impacto negativo. A partir de ello, se determinaron

cuatro grandes áreas en dónde podrían ocurrir, a nivel tecnológico, a nivel de gestión, dentro del equipo y por elementos externos al proyecto.

En base a ello, se identificaron los siguientes riesgos pudiendo encontrar su correspondiente medida de mitigación dentro del Anexo 13.9:

Id	Riesgo	Descripción
<i>Riesgos Tecnológicos</i>		
1.1	Falta de conocimiento en LLM	El equipo puede enfrentar dificultades técnicas al integrar el modelo de lenguaje debido a la falta de experiencia previa.
1.2	Problemas de compatibilidad	Posibles conflictos entre el LLM y las tecnologías existentes.
1.3	Escalabilidad de la solución	El sistema puede no manejar adecuadamente un aumento significativo de usuarios o consultas.
1.4	Errores en el despliegue	Problemas durante la configuración de la infraestructura de despliegue (frontend y backend).
1.1.4	Falta de conocimiento sobre las herramientas de AWS	Riesgo de incurrir en errores operativos o de configuración, así como retrasos en los plazos de entrega por curvas de aprendizaje elevados en las herramientas.
1.5	Pérdida de acceso a tecnologías	Riesgo de quedar sin acceso a herramientas, cuentas o servicios críticos (correo electrónico, repositorios) que soportan el desarrollo de este proyecto.
1.5.1	Control de acceso a modelo de IA	Riesgo de no disponer de un mecanismo seguro y autónomo para gestionar credenciales o claves de API que permitan operar el modelo de IA sin depender únicamente de una sola API Key para poder realizar consultas.
<i>Riesgos de Gestión</i>		
2.1	Retrasos en entregas	Problemas en la coordinación del equipo que lleven a incumplir los plazos.
2.2	Desacuerdos con el Product Owner	Cambios frecuentes en los requerimientos o falta de alineación.
2.3	Falta de comunicación	Poca claridad en los objetivos o tareas asignadas.

2.4	Falta de claridad en los roles	Confusión sobre las responsabilidades de cada integrante, lo que podría generar ineficiencias y conflictos.
2.5	Desalineación con los objetivos del cliente	Cambios frecuentes en los requerimientos o desalineación con los objetivos del cliente, lo que puede provocar desvíos en las prioridades y afectar el valor del producto entregado.
<i>Riesgos del Equipo</i>		
3.1	Disponibilidad del equipo	Compromisos personales o profesionales que limiten el tiempo dedicado al proyecto.
3.2	Desbalance en la carga de trabajo	Un desbalance podría generar retrasos en áreas críticas si una persona tiene demasiadas tareas.
<i>Riesgos Externos</i>		
4.1	Cambios en las tecnologías usadas	Actualizaciones inesperadas en el LLM o las herramientas principales que introduzcan incompatibilidades.
4.2	Interrupciones en servicios externos	Fallos en servicios externos necesarios para el proyecto (APIs, servidores de despliegue).
4.3	Costos de infraestructura	Incrementos inesperados en el costo de los servicios utilizados, debido a un aumento en el uso o ineficiencias del sistema.
4.4	Legislación y regulaciones de datos	Manejo de información personal y sensible sobre los usuarios, asegurarse que se siguen los lineamientos planteados por las leyes sobre privacidad de datos y el uso de modelos de inteligencia artificial.

Tabla 5: Riesgos identificados

8.3. Análisis cuantitativo

Con el objetivo de cuantificar la severidad de los riesgos identificados, el equipo asignó a cada riesgo dos atributos numéricos para ilustrar su probabilidad de ocurrencia y su impacto dentro del proyecto.

Probabilidad de ocurrencia (P): valor numérico que indica la probabilidad de ocurrencia de dicho riesgo en un momento determinado del proyecto. Se comenzó utilizando una escala de 0 (siendo improbable), a 3 (probabilidad de ocurrencia alta). Pero se observó que esta escala no permitía reflejar realmente la probabilidad de ocurrencia de

los riesgos ya que el margen de evaluación era acotado. Por ello, para la segunda revisión de riesgos se decidió ampliar la escala de 0 (siendo improbable) a 5 (Probabilidad de ocurrencia alta).

Impacto (I): cuantificación numérica asignada por el equipo, utilizada para reflejar el impacto de las consecuencias que corresponde a la ocurrencia del riesgo. Para ilustrar con mayor fidelidad el efecto del riesgo sobre el proyecto, el equipo consideró dos factores complementarios al impacto, la gravedad de las consecuencias que causaría el riesgo en caso de ocurrir y el esfuerzo necesario para la resolución. El primero refiere a la pérdida de funcionalidades, degradación en la calidad del servicio, incumplimiento de requisitos, entre otros. Mientras que el segundo refiere a cuántas horas de trabajo ideales demandaría mitigar o resolver el problema generado por el riesgo.

Al igual que la probabilidad, se comenzó con una escala de 1 a 3 y luego se comenzó a utilizar una escala de 1 a 5, donde uno significaba que el impacto era insignificante y cinco que el mismo era crítico y ponía en riesgo los objetivos del proyecto.

Ambos atributos se utilizaron para calcular la severidad del riesgo. Para su cálculo se utilizó la siguiente ecuación:

$$Severidad = P * I$$

Este valor permitió al equipo ponderar entre los riesgos identificados, para categorizarlos como críticos, moderados y bajos. Esta categorización fue ilustrada en la siguiente Matriz de riesgos, en donde los riesgos bajos se ilustran en color verde, los riesgos moderados se ilustran en color amarillo, y los riesgos críticos en color rojo.

		<i>Probabilidad</i>				
		1 (Baja)	2 (Moderada/Baja)	3 (Moderada)	4 (Moderada/Alta)	5 (Alta)
<i>Impacto</i>	1 (Insignificante)	1	2	3	4	5
	2 (Menor)	2	4	6	8	10
	3 (Moderado)	3	6	9	12	15
	4 (Significativo)	4	8	12	16	20
	5 (Crítico)	5	10	15	20	25

Ilustración 51: Matriz de riesgos

A continuación se adjunta un ejemplo de asignación de valores de probabilidad de ocurrencia y de impacto, así como su correspondiente valor de severidad para el riesgo 1.5. *Pérdida de acceso a las tecnologías*, en el mes de mayo.

Riesgos	Probabilidad	Impacto	Severidad
1.5 Pérdida de acceso a tecnologías	2	5	10

Tabla 6: Valor establecido para el riesgo 1.5 durante la revisión de mayo de 2025.

8.4. Seguimiento

El seguimiento de los riesgos se realiza con el fin de evitar que aquellos riesgos predecibles, impredecibles, o desconocidos, pasen a ser riesgos conocidos, como plantea el autor Charette en su libro. Por tanto, el equipo decidió implementar un seguimiento recurrente de los mismos durante todos los ciclos del proyecto, como ya fue mencionado anteriormente.

Para ello, durante las fases iniciales del proyecto, cuando el equipo aún se encontraba familiarizándose con las tecnologías y dinámicas de desarrollo, se estableció una revisión quincenal de los riesgos. Estas instancias permitían analizar e identificar nuevos factores que generaban incertidumbre en el equipo y así ajustar la probabilidad y grado de impacto de los riesgos previamente identificados.

Una vez que el equipo logró alcanzar una mayor estabilidad en el desarrollo, la revisión principal de riesgos pasó a realizarse de forma mensual, complementada por una evaluación quincenal más breve, centrada en posibles incidencias recientes. En cada revisión se evaluaba el estado actual de los riesgos identificados, posibles variaciones en sus valores de probabilidad de ocurrencia e incidencia, y si era necesario ajustar o reforzar las medidas de mitigación previamente definidas. Además, en aquellos casos que se consideraban pertinentes, se procedía a la eliminación del riesgo ya que el valor de probabilidad de ocurrencia pasaba a ser improbable.

Las notas extraídas de cada una de estas reuniones de valoración se pueden encontrar en el siguiente apartado del Anexo 13.9. A continuación se muestra la gráfica en donde se evidencia la evolución de la severidad de los riesgos a lo largo del proyecto, las

variaciones en los valores permitió al equipo reflexionar sobre la evolución de la incidencia de dichos riesgos a lo largo del proyecto.

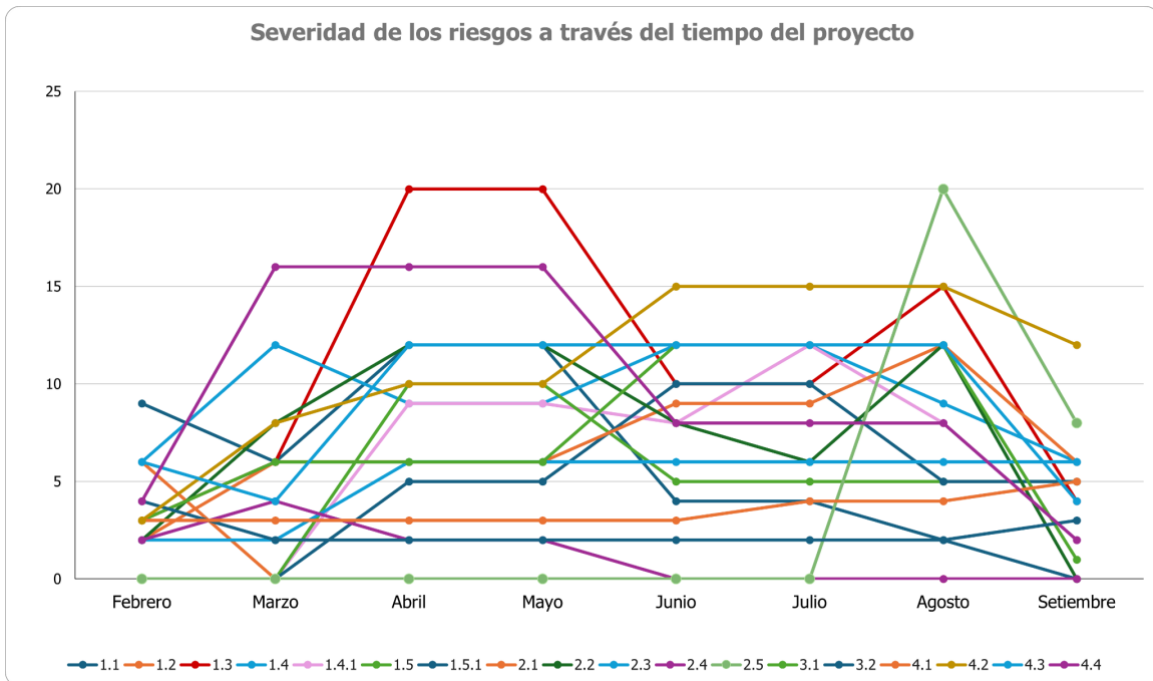


Ilustración 52: Evolución de la severidad de los riesgos a través del tiempo.

A pesar de los esfuerzos del equipo al realizar este seguimiento recurrente, con el fin de mantener un enfoque proactivo, durante el desarrollo surgieron incidentes que evidenciaron la incorporación de ciertos riesgos al proceso de gestión. Un ejemplo de ello fue el riesgo “1.5. Pérdida de acceso a tecnologías”, que se materializó cuando la cuenta de *Gmail* utilizada para la administración de credenciales, incluyendo servicios como *Pinecone* y *AWS*, fue bloqueada por *Google* debido a actividad sospechosa. La situación se resolvió tras enviar una apelación, que fue aceptada y permitió restaurar el acceso. Sin embargo, el riesgo se mantuvo activo, ya que la cuenta también estaba vinculada al usuario raíz de *AWS*, y una posible suspensión futura podría generar consecuencias críticas durante las etapas de despliegue.

Esta vivencia dejó como aprendizaje al equipo, que si bien es posible reducir significativamente la ocurrencia de un riesgo mediante una gestión proactiva, esto no garantiza la ocurrencia de eventos inesperados.

Se destacan dentro de dicho gráfico, aquellos riesgos que mantuvieron una severidad baja durante el desarrollo del proyecto (e.g. “3.2 *Desbalance en la carga de trabajo*”) que de

igual manera el equipo decidió mantener sus estrategias de mitigación con el fin de evitar un aumento en la probabilidad de ocurrencia e incidencia de los mismos, logrando mantenerlos con una severidad baja.

9. Gestión de calidad

La gestión de calidad en este proyecto se fundamenta en el principio establecido por Schulmeyer: "la calidad de un producto o servicio depende de la calidad del proceso usado para crearlo" [37]. Esta perspectiva reconoce que la calidad no es un elemento que se añade al final del desarrollo, sino que debe estar integrada en cada etapa del proceso, desde la definición de requerimientos hasta el despliegue y mantenimiento del sistema.

Para este proyecto, se adoptó la definición de calidad como "el grado al cual un objeto satisface un conjunto específico de atributos o requerimientos" [37]. Esta definición orienta tanto los objetivos del producto como los del proceso, estableciendo un marco conceptual para todas las actividades de aseguramiento de calidad. En el contexto del proyecto, esto se traduce en garantizar que el sistema cumpla con los requerimientos funcionales definidos.

La aplicación de estos principios se materializó a través de un Plan de Calidad estructurado el cual se detalla en el capítulo 9.3.

9.1. Objetivos del producto

Los objetivos de calidad del producto se enfocan en garantizar que el sistema cumpla con los requerimientos funcionales y no funcionales especificados en el Anexo 13.4, manteniendo niveles medibles de rendimiento, usabilidad, seguridad y escalabilidad. Adicionalmente, se estableció como objetivo verificar que la solución desarrollada satisficiera efectivamente las necesidades de los usuarios finales, evaluado mediante métricas relevantes.

9.2. Objetivos del proceso

Los objetivos de calidad del proceso se centraron en ejecutar todas las actividades de aseguramiento de calidad planificadas, mantener el cumplimiento de los estándares de codificación y testing establecidos, y gestionar efectivamente los defectos identificados mediante su detección, documentación y resolución sistemática. Se estableció además como objetivo la detección temprana de desviaciones en los procesos y la implementación oportuna de acciones correctivas.

9.3. Plan de calidad

El desarrollo del plan de calidad para este proyecto adoptó un enfoque estructurado que, según Fontdevila & Oliveros (2007) [38], comprende cuatro actividades fundamentales: identificar el ambiente y características del proyecto, seleccionar el proceso y actividades a realizar, documentar el plan de calidad, y mantenerlo actualizado a lo largo del ciclo de vida del proyecto.

Caracterización del proyecto

La caracterización inicial del proyecto identificó factores determinantes para la definición del proceso de calidad. Si bien se contaba con experiencia en desarrollo de software tradicional, se enfrentaba un desafío técnico significativo al trabajar con tecnologías emergentes como LLMs y bases de datos vectoriales. Esta limitación inicial, combinada con la volatilidad esperada en los requerimientos debido a la naturaleza exploratoria de la solución, determinó la necesidad de incorporar fases de investigación mediante spikes tecnológicos y adoptar un enfoque iterativo que permitiera validaciones tempranas y ajustes progresivos.

Se trata de un producto web *multi-tenant* basado en servicios con componentes desplegados en AWS, integrando un modelo de lenguaje mediante técnicas de RAG. Se requería garantizar atributos de calidad como seguridad, disponibilidad, usabilidad y flexibilidad.

Selección del proceso

Con base en las características identificadas, se optó por un enfoque de gestión ágil, como fue mencionado anteriormente, complementado con prácticas de integración continua, revisión de código y testing automatizado. Esta decisión se fundamentó en la necesidad de mantener flexibilidad ante cambios, favorecer la comunicación directa entre los miembros del equipo, y garantizar entregas incrementales validables con el cliente.

Plan de calidad

El plan de calidad se organizó en seis fases principales, cada una con actividades y mecanismos específicos de aseguramiento de calidad integradas al flujo de trabajo del proyecto.

Es importante destacar que estas fases no fueron ejecutadas de manera secuencial o en cascada, sino que se desarrollaron de forma paralela y cíclica, alineándose con la naturaleza ágil del proceso de desarrollo adoptado. Algunas actividades se ejecutaron simultáneamente, mientras que otras se repitieron durante diferentes etapas del proyecto. Por ejemplo, las actividades de discovery e ingeniería de requerimientos se realizaron en la fase inicial, pero se extendieron y refinaron durante la fase de desarrollo a través de ciclos iterativos. De manera similar, la investigación de tecnologías de LLM se ejecutó de manera concurrente con las actividades de desarrollo y despliegue, reflejando la naturaleza iterativa del enfoque adoptado.

Esta organización en fases sirvió como marco estructural para identificar y planificar las actividades específicas de aseguramiento de calidad necesarias en cada etapa, permitiendo al equipo gestionar de manera sistemática los diferentes aspectos de calidad del producto sin limitar la flexibilidad del proceso de desarrollo ágil. El plan de calidad puede observarse en el Anexo 13.10.1.

Fase de Investigación

Debido a las necesidades tecnológicas que plantea el cliente con las cuales el equipo no tenía experiencia previa se acordó realizar una fase de investigación orientada a establecer las bases tecnológicas del proyecto mediante el estudio de alternativas y la creación de pruebas de concepto que validaron las decisiones arquitectónicas fundamentales.

Fase de Ingeniería de Requerimientos

Centrada en la especificación, prototipado y validación de los requerimientos del sistema, asegurando la correcta comprensión de las necesidades del cliente y usuarios finales. Durante esta fase se logró una especificación comprensiva de los requerimientos del cliente.

Fase de Diseño

Enfocada en el análisis de costos, diseño de la arquitectura del sistema y su posterior revisión, garantizando que la solución propuesta sea técnicamente viable y escalable.

Fase de Desarrollo

Abarcó la planificación del desarrollo, codificación de funcionalidades y revisión sistemática del código, implementando estándares de calidad y buenas prácticas de programación.

Fase de Testing

Comprendió el diseño e implementación de diferentes tipos de pruebas, desde unitarias hasta de usabilidad, incluyendo pruebas específicas para sistemas de inteligencia artificial. Esta fase se caracterizó por la integración continua (CI) que ejecutaba automáticamente todas las pruebas con cada *Pull Request*, asegurando que solo el código que pasara exitosamente todas las validaciones pudiera ser integrado a la rama principal.

Fase de Despliegue

Incluyó la planificación del despliegue, configuración de la infraestructura en AWS y establecimiento de sistemas de monitoreo continuo para el entorno de producción.

Procesos de Gestión Transversales

Actividades independientes de las fases que se ejecutaron durante todo el ciclo de vida del proyecto, incluyendo planificación de actividades, gestión de riesgos y diseño del propio plan de calidad.

9.4. Aseguramiento de calidad

Las actividades de aseguramiento de calidad implementadas en este proyecto se diseñaron para integrarse naturalmente en el flujo de trabajo del equipo, proporcionando mecanismos efectivos de verificación y validación sin comprometer la agilidad del desarrollo.

9.4.1. Validación de requerimientos con usuarios

Como se detalla en el capítulo 2 , la fase de ingeniería de requerimiento fue marcada por la validación de requerimientos para la cual se realizó un proceso iterativo que incluyó la creación de prototipos en *Figma*, encuestas con usuarios potenciales y entrevistas con expertos del dominio. Este enfoque multi-método aseguró que los requerimientos especificados reflejaran fielmente las necesidades reales de los usuarios finales y que la solución propuesta fuera técnicamente viable y comercialmente atractiva.

9.4.2. Estudio de tecnologías y pruebas de concepto

Como se menciona en el capítulo 3, durante la fase de investigación, se implementó un proceso sistemático de evaluación de alternativas tecnológicas mediante la creación de pruebas de concepto (POCs). Cada POC fue diseñado para validar aspectos específicos de la arquitectura propuesta, desde la selección del proveedor de LLM hasta la elección de la base de datos vectorial. Este enfoque permitió tomar decisiones fundamentadas basadas en evidencia empírica, reduciendo significativamente los riesgos técnicos del proyecto.

9.4.3. Definición de criterios de aceptación

Para cada funcionalidad desarrollada, se establecieron criterios de aceptación claros y medibles antes del inicio de la implementación. Estos criterios sirvieron como base para las pruebas de aceptación y proporcionaron una definición objetiva de cuándo una funcionalidad podía considerarse completada. La definición temprana de estos criterios facilitó la comunicación entre los miembros del equipo y redujo la ambigüedad en los requerimientos.

BE - Add label to question

As a User,

I want to be able to add labels to questions,

In order to improve question categorization.

Endpoint: PUT /questions/{questionId}/labels

```
PUT /questions/{questionid}/label
```

```
body: {  
  "labelIds": [1, 2, 3]  
}
```

Acceptance Criteria:

- Correctly adds or removes the labels from the question
- Only users with admin role can modify labels
- Returns 200 with updated question including all labels
- Returns 401 if user is not authenticated
- Returns 403 if user is not an admin
- Returns 404 if question or any label doesn't exist
- Returns 400 if labelIds is not a valid array
- Returns descriptive error messages when errors occur
- Unit and integration tests with >90% coverage

Ilustración 53: Ejemplo de tarea en el backlog con sus criterios de aceptación.

9.4.4. Revisión de arquitectura

La arquitectura del sistema fue sometida a un proceso formal de revisión que incluyó la evaluación de aspectos técnicos, de escalabilidad y de mantenibilidad. Esta revisión se realizó con la participación de todo el equipo, permitiendo identificar potenciales problemas de diseño antes de la fase de implementación y asegurando que la arquitectura propuesta fuera robusta y extensible.

9.4.5. Revisión sistemática de código

Se implementó un proceso obligatorio de revisión de código que requería un mínimo de dos aprobaciones antes de realizar un merge a la rama principal. Este proceso no solo garantiza la calidad del código, sino que también facilita la transferencia de conocimiento

entre los miembros del equipo y la detección temprana de defectos. Las revisiones se enfocaron en aspectos de funcionalidad, legibilidad, mantenibilidad y adherencia a los estándares de codificación establecidos.

9.4.6. Implementación de pruebas automatizadas e integración continua

El proyecto implementó una estrategia integral de testing, articulada dentro de un pipeline automatizado de CI/CD, que permitió ejecutar de forma continua todas las actividades vinculadas al aseguramiento de la calidad.

Pruebas unitarias

Se adoptó un modelo de responsabilidad individual, según el cual cada desarrollador asumía no solo la implementación de una funcionalidad, sino también la creación de sus correspondientes pruebas unitarias. Este enfoque se fundamentó en el principio establecido por Draughtrey: "El ingeniero de software que escribe un programa es el que está en la mejor posición para encontrar y corregir sus defectos. Es fundamental que los ingenieros de software asuman la responsabilidad personal por la calidad de los programas que producen." [39]. Esta filosofía aseguró que las pruebas fueran diseñadas con un conocimiento profundo de la lógica implementada y los casos de uso específicos.

Para cada funcionalidad del backlog, se diseñaron casos de prueba (Anexo 13.10.2) detallados que contemplaban tanto escenarios de éxito como casos de error, documentados en tablas específicas que servían como guía durante la implementación. Esta planificación previa garantizó una cobertura exhaustiva de los diferentes caminos de ejecución del código.

Las pruebas unitarias fueron implementadas utilizando Jest como framework principal, alcanzando y manteniendo una cobertura superior al 90% del código desarrollado. Esta alta cobertura no sólo validaba el correcto funcionamiento del código nuevo, sino que también proporcionaba una red de seguridad que detectaba regresiones, asegurando que las modificaciones no afectarían funcionalidades previamente implementadas.

File	% Stmts	% Funcs	% Lines
All files	96.07	96.21	96.15
clients	100	100	100
rag.client.ts	100	100	100
controllers	96.96	97.82	96.81
answer.controller.ts	98.37	100	98.24
auth.controller.ts	100	100	100
data.controller.ts	93.24	83.33	94.02
health.controller.ts	100	100	100
question.controller.ts	92.78	100	92.13
statistics.controller.ts	100	100	100
tag.controller.ts	98.18	100	97.95
tenant.controller.ts	100	100	100
user.controller.ts	95.68	100	95.37
userToTenant.controller.ts	100	100	100
vote.controller.ts	100	100	100
errors	100	100	100
CustomErrors.ts	100	100	100
middlewares	96.42	86.66	97.46
auth.middleware.ts	96.42	86.66	97.46
repositories	94.72	93.75	94.61
answer.repository.ts	100	100	100
datachunk.repository.ts	90	75	83.33
question.repository.ts	89.91	89.65	90.26
tag.repository.ts	100	100	100
tenant.repository.ts	90.9	87.5	90
user.repository.ts	98.43	100	98.38
userToTenant.repository.ts	100	100	100
vote.repository.ts	94.44	80	93.75
services	94.76	97.59	95.1
answer.service.ts	100	100	100

Ilustración 54: Porcentaje de cobertura de los tests unitarios.

El pipeline ejecutaba automáticamente todas las pruebas en cada *commit*, proporcionando feedback inmediato sobre la calidad del código y bloqueando la integración de cambios que no cumplieran con los criterios de calidad establecidos.

Pruebas de integración

Para cada nueva funcionalidad desarrollada se añadieron las pruebas de integración correspondientes las cuales validan la correcta interacción entre módulos del sistema, asegurando que los componentes funcionen correctamente en conjunto.

Pruebas de usabilidad

Durante el desarrollo del sistema, se llevaron a cabo pruebas de usabilidad con usuarios reales en cada ciclo iterativo, siguiendo el enfoque incremental adoptado en el proyecto. Para facilitar la participación, se creó y compartió un enlace de Calendly donde personas interesadas en el sistema pudieran registrarse y programar entrevistas de máximo 30

minutos para probar las funcionalidades implementadas en cada incremento y proporcionar retroalimentación.

Estas pruebas de usabilidad siguieron la misma metodología aplicada durante la evaluación del prototipo (ver capítulo 4 - Ingeniería de requerimientos), utilizando la técnica *Think Aloud Protocol*. Se definió una lista de tareas con puntos de inicio y finalización claros, solicitando a los usuarios que ejecuten cada tarea mientras verbalizan su razonamiento y proceso de pensamiento.

Este enfoque de evaluación permitió analizar distintas dimensiones del sistema, como la facilidad de uso de la interfaz, la intuitividad de los flujos de navegación y la calidad percibida de las respuestas generadas por el modelo de lenguaje. Además, las sesiones aportaron información valiosa sobre posibles mejoras y nuevas funcionalidades que los usuarios consideraron relevantes para el producto.

Los resultados de estas pruebas no sólo validaron la usabilidad del sistema desarrollado, sino que también confirmaron el cumplimiento de los objetivos de calidad establecidos respecto a la facilidad de uso y la satisfacción del usuario final. Las métricas recabadas de esta actividad pueden observarse más adelante en el capítulo 9.5, Métricas de calidad.

Los documentos generados a partir de estas pruebas así como un resumen de los comentarios recibidos por los usuarios puede observarse en los Anexos 13.10.3 y 13.10.4.

9.4.7. Pruebas específicas para sistemas de IA

Dadas las características particulares del sistema de inteligencia artificial, se desarrollaron pruebas específicas para evaluar la precisión, relevancia y comportamiento ético del modelo de lenguaje. Estas pruebas incluyeron la evaluación manual de respuestas, el análisis de métricas automatizadas para detectar sesgos y alucinaciones, y la implementación de mecanismos de feedback continuo basados en la valoración de los usuarios.

9.4.8. Monitoreo continuo y feedback de usuarios

El sistema en producción fue equipado con mecanismos de monitoreo continuo que permitían detectar problemas de rendimiento, disponibilidad y calidad de las respuestas. Adicionalmente, se implementó un sistema de feedback de usuarios mediante botones de

valoración ("¿Fue útil?") que proporcionaba datos cuantitativos sobre la satisfacción del usuario y la calidad percibida del sistema. El feedback obtenido mediante este sistema se detalla en el capítulo 9.5 Métricas de calidad.

9.4.9. Gestión proactiva de riesgos de calidad

Se mantuvo una matriz de riesgos actualizada semanalmente que identificaba, evaluaba y mitigaba riesgos específicos relacionados con la calidad del software y, particularmente, con los sistemas de inteligencia artificial. Esta matriz prestó especial atención a riesgos como la falta de conocimiento en LLM, problemas de compatibilidad, errores en el despliegue y escalabilidad de la solución, el análisis detallado de los riesgos se puede ver en el capítulo 8.

9.4.10. Gestión de defectos y mejora continua

Como parte del proceso de mejora continua, se estableció un estándar para el reporte de bugs. Este estándar, permitió una gestión sistemática de los defectos identificados y facilitará el mantenimiento futuro del sistema.

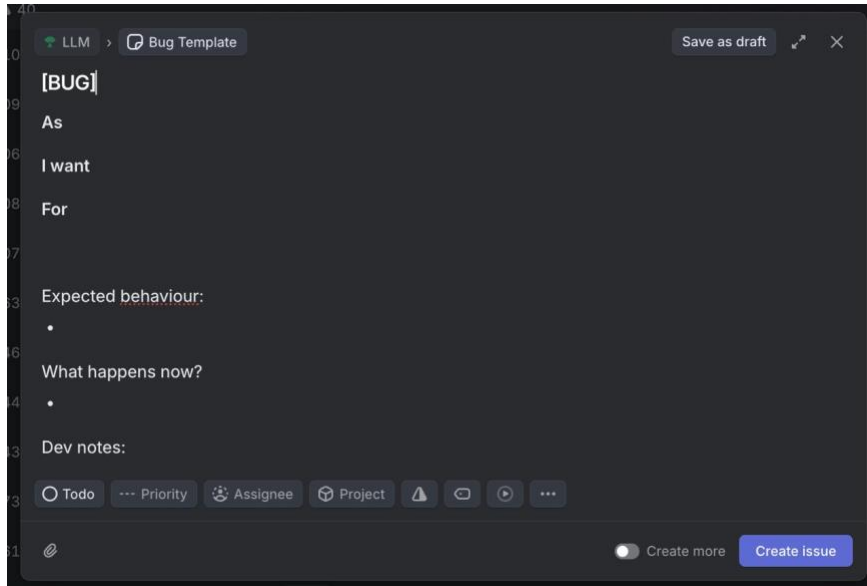


Ilustración 55: Template para Bugs

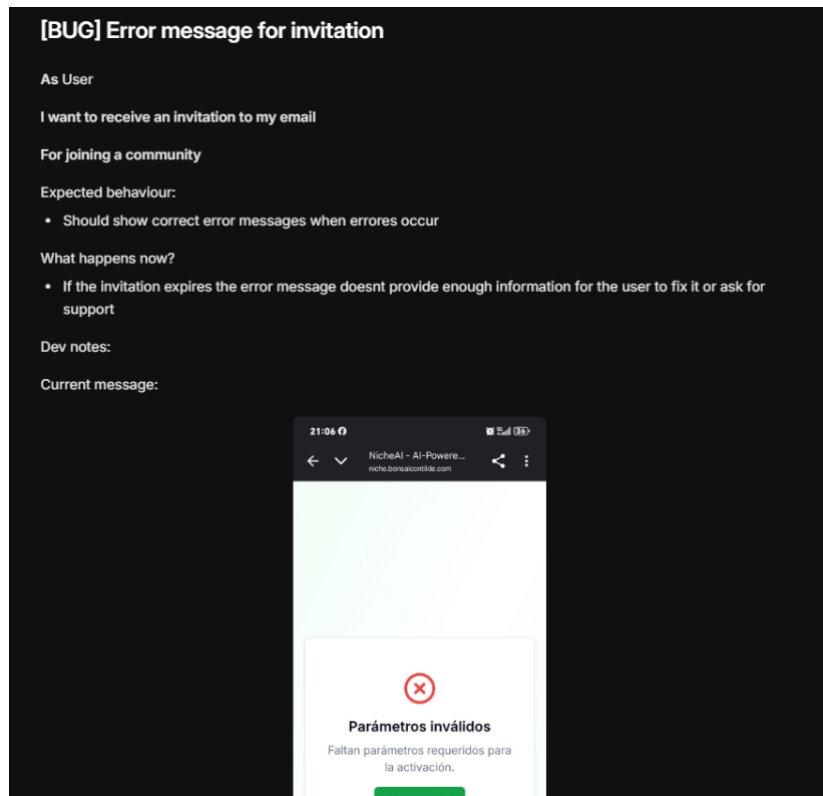


Ilustración 56: Bug creado con el template.

9.4.11. Auditorías y revisiones de calidad

Se realizaron tres tipos de auditorías a lo largo del proyecto:

Revisiones: Se llevaron a cabo tres revisiones formales con Gastón Mousqués, Rafael Bentancur y Amalia Álvarez, para presentar los avances del proyecto y obtener feedback. A partir de los comentarios recibidos se generó un documento de puntos fuertes y débiles del proyecto sobre los cuales aplicar acciones correctivas. Estas auditorías permitieron validar el enfoque técnico y metodológico del proyecto. Los resúmenes de los reportes de estas revisiones pueden verse en el Anexo 13.10.5.

Seguimiento del cronograma: Se implementó un sistema de monitoreo continuo del cronograma establecido mediante reuniones mensuales entre los integrantes, con acciones correctivas cuando se detectaron desviaciones.

Auditorías de proceso: A través de las retrospectivas y reuniones periódicas mencionadas en el capítulo 7, se evaluó continuamente la efectividad de los procesos implementados y se realizaron ajustes según fuera necesario.

9.5. Métricas de calidad

Además de las métricas definidas para el análisis del proceso en el capítulo 7. Se definió una lista de métricas relacionadas a la calidad del producto. A continuación se presentan las métricas utilizadas para asegurar la calidad del producto:

Heurísticas de nielsen

Las heurísticas de Nielsen constituyen una métrica esencial para evaluar la usabilidad de interfaces web desde la perspectiva del usuario. Desde el inicio del desarrollo, la implementación del *frontend* se realizó teniendo en cuenta estos principios para minimizar el retrabajo y garantizar una experiencia de usuario óptima.

Durante el análisis final de usabilidad, se evaluaron las diez heurísticas de Nielsen aplicadas a la plataforma, identificándose cuatro problemas puntuales distribuidos en diferentes categorías:

En la heurística Relación entre el sistema real y el mundo, el uso inicial del término técnico "*tenant*" debió ser corregido a "comunidad" para mejor comprensión.

Por otra parte, la ausencia de funcionalidad para eliminar preguntas propias afectaba el control y libertad del usuario.

La flexibilidad y eficiencia de uso se vio levemente afectada por problemas de responsive en algunas secciones específicas en modo móvil.

Por último, se identificó un mensaje de error no descriptivo durante la activación de usuarios con invitaciones vencidas lo cual no permitía a los usuarios recuperarse de ese error.

Estos *issues* fueron documentados y registrados como *bugs* en el sistema de gestión de proyectos para su resolución en futuras iteraciones. La cantidad reducida de problemas identificados se encuentra dentro de los parámetros esperados para un MVP en producción, y la mayoría de las heurísticas cumplieron satisfactoriamente con los estándares de usabilidad establecidos. Una versión detallada del análisis se puede encontrar en el Anexo 13.10.6, mostrando los problemas principales encontrados así como los aspectos positivos implementados para cada heurística.

Defectos y retrabajo

La gestión de calidad del proyecto se basó en un enfoque preventivo y de respuesta inmediata. Cada defecto detectado era asignado al desarrollador responsable y corregido en el momento, lo que permitió mantener la continuidad del flujo de trabajo y evitar la acumulación de errores pendientes.

El equipo mantuvo un seguimiento continuo mediante validaciones con usuarios y revisiones iterativas. Los defectos detectados durante el desarrollo fueron resueltos de forma inmediata, evitando su acumulación entre ciclos. Al cierre del proyecto, se documentaron únicamente tres bugs en el sistema Linear (Tabla 7): ninguno de severidad crítica, uno moderado y dos leves, todos corregidos antes de la entrega final.

Como oportunidad de mejora identificada, la incorporación de métricas cuantitativas complementarias habría enriquecido este proceso de aseguramiento de calidad. Específicamente, la medición sistemática de métricas como densidad de defectos por ciclo (bugs/funcionalidad entregada), tasa de retrabajo (esfuerzo correctivo vs. desarrollo de nuevas características) y tendencia de estabilidad (evolución de defectos entre sprints).

Estas métricas habrían proporcionado información cuantitativa adicional para validar con datos objetivos la percepción de estabilidad observada y optimizar estimaciones futuras basándose en patrones históricos del proyecto.

Descripción	Aplicación	Severidad	Categoría	Estimación en SP
Filtrar preguntas no tiene soporte para más de una etiqueta	Niche-API	Bajo	Funcional	1
Mensaje de error cuando vence la invitación no es descriptivo	Niche-WEB	Moderado	Estético	0.5
Errores estéticos debido a responsiveness	Niche-WEB	Bajo	Estético	0.5

Tabla 7: Registro de bugs y tech-debt.

Satisfacción de usuarios finales

Para evaluar la satisfacción de los usuarios, como se mencionó previamente se realizaron sesiones de pruebas de usabilidad, se registraron los tiempos de completitud de cada tarea para evaluar la eficiencia y facilidad de uso de la plataforma. La siguiente tabla presenta los tiempos promedio observados para las tareas principales:

Tarea	Tiempo Promedio	Observaciones
Iniciar sesión	<10 segundos	Todos los usuarios complementarios está tarea sin dificultades
Seleccionar Comunidad	< 5 segundos	Los usuarios comprendieron el concepto inmediatamente
Hacer una pregunta	1-3 minutos	El tiempo varió principalmente por la redacción de contenido, no por dificultades de navegación.
Ver mis preguntas	< 10 segundos	Acceso intuitivo desde el menú principal
Modificar perfil	< 1 minuto	Uno solo de los usuarios experimentó dificultades para encontrar la sección de perfil de usuario
Votar respuestas	< 10 segundos	Todos los usuarios identificaron los controles de votación inmediatamente
Buscar contenido	< 1 minuto	Encontraron y utilizaron la función de búsqueda sin dificultades.

Tabla 8: Medición de tiempo por tarea

Los tiempos de completitud registrados demuestran que la interfaz cumple con los estándares de usabilidad establecidos, con la mayoría de las tareas completadas en menos de 1 minuto, lo cual indica un diseño intuitivo y eficiente. Las tareas que requirieron más tiempo, como hacer una pregunta, estuvieron asociadas principalmente a la redacción del contenido por parte del usuario y no a dificultades de navegación o comprensión de la interfaz.

Al finalizar cada sesión, se solicitó a los usuarios que califiquen la facilidad de uso general de la plataforma en una escala de 1 a 5, donde 1 representa "muy fácil de usar" y 5 "muy difícil de usar". Los resultados obtenidos fueron consistentemente positivos, con todos los usuarios asignando calificaciones de 1 o 2, lo que refleja una experiencia de usuario

altamente satisfactoria. Los usuarios describieron la plataforma como "muy intuitiva", "clara", "sencilla" y destacaron que no encontraron dificultades significativas durante la navegación.

Por otra parte, se integró *Sprig* para que con ayuda del cliente Darío se pudiera compartir el sitio con más usuarios seleccionados y que puedan usarlo en su propio tiempo haciendo preguntas y dejando feedback opcional sobre las respuestas obtenidas.

Al momento de escribir este reporte la encuesta sobre la iteración más reciente del producto recibió 7 respuestas, de las cuales se obtuvieron los datos presentados en la siguiente tabla.

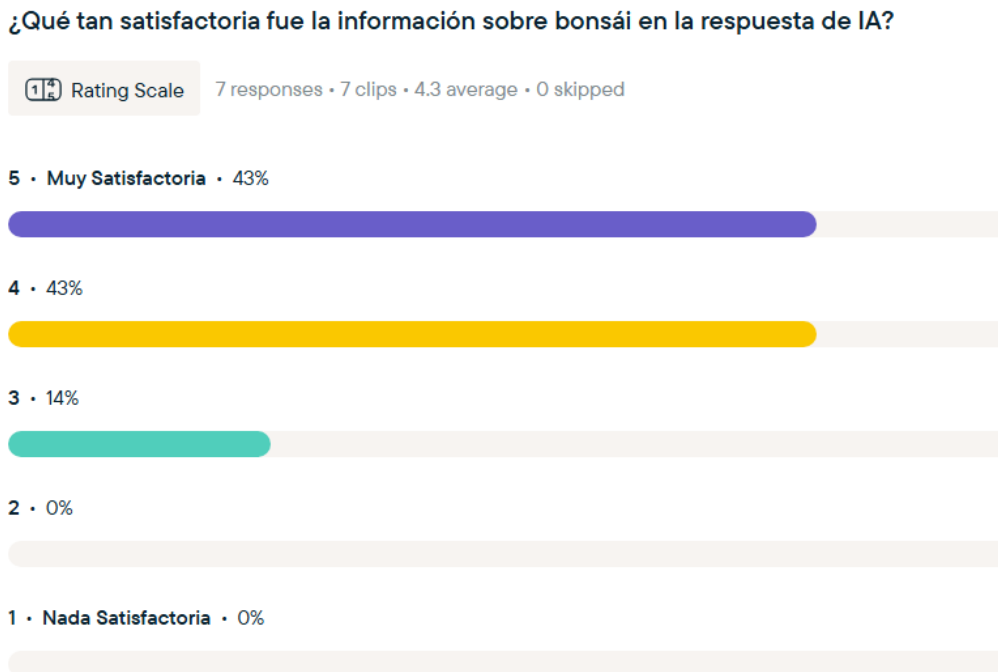


Ilustración 57: Resultados de la encuesta en Sprig.

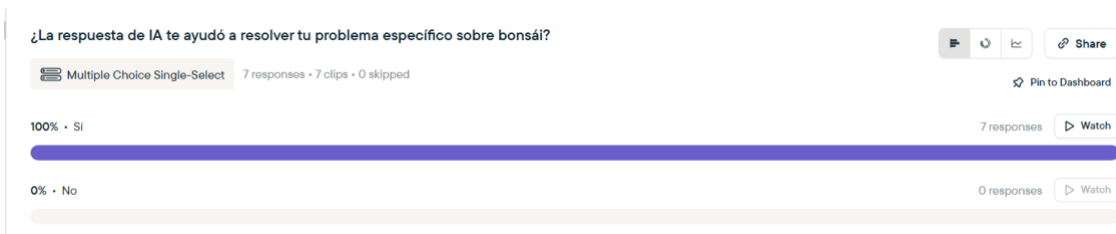


Ilustración 58: Resultados de la encuesta en Sprig.

Pregunta	Escala	Resultado	Objetivo
¿Qué tan precisa fue la información sobre bonsái en la respuesta de IA?	1-Muy inexacta al 5-Muy precisa	Promedio 4.4	> 4
¿Qué tan satisfactoria fue la información sobre bonsái en la respuesta de IA?	1- Nada Satisfactoria al 5-Muy satisfactoria	Promedio 4.3	> 4
¿Qué tipo de información faltó en la respuesta de IA?	Opciones: No faltó información, Detalles técnicos específicos, Pasos prácticos de implementación, información sobre casos específicos, otros.	4 personas contestaron: No faltó información. 2 personas: Información sobre casos específicos. 1 persona: Detalles técnicos específicos.	> 80% de las personas contesten no faltó información.
¿Qué tan confiado te sientes en aplicar los consejos de la IA a tu bonsái?	1- No confío en absoluto al 5-Confío completamente	Promedio 4.1	> 4
¿La respuesta de IA te ayudó a resolver tu problema específico sobre bonsái?	Si o No	100% de las personas contestaron SI	> 80% de las personas contestaran Si

Tabla 9: Resultados y objetivos de la encuesta en Sprig.

En la tabla 9 se puede observar el resultado versus el objetivo para cada pregunta de la encuesta.

Se puede observar que la única métrica que no alcanzó los objetivos establecidos corresponde a la pregunta 3, relacionada con el tipo de información faltante. Este resultado se atribuye a una insuficiencia de datos en la base vectorial, dado que al momento de liberar la encuesta a los usuarios, el sistema aún se encontraba a la espera de que el cliente cargara más *pills* de datos con información específica del nicho de bonsáis. Esta limitación impactó la capacidad del sistema para brindar respuestas más detalladas y especializadas basadas en conocimiento específico del dominio.

No obstante, los resultados de la encuesta a usuarios finales se consideran satisfactorios en todos los demás aspectos evaluados. A pesar de la limitada disponibilidad de pills durante la instancia de prueba, las respuestas generadas por el sistema fueron lo suficientemente precisas y útiles para asistir efectivamente a los usuarios en la resolución de sus consultas dentro del dominio de bonsáis.

Satisfacción del cliente

En cuanto a la satisfacción del cliente, durante todo el proceso hubo reuniones periódicas para recibir actualizaciones de las *User Stories*, obtener *feedback* de los avances del LLM y nuevas funcionalidades. Además se le envió una encuesta final con espacio para completar su satisfacción general con el producto así como cualquier comentario sobre el proyecto.

La evaluación del cliente mediante la encuesta resultó altamente positiva, manifestando estar muy satisfecho con la plataforma y considerando que cumple completamente con las expectativas acordadas.

En cuanto al equipo, el cliente evaluó positivamente el profesionalismo y la comunicación, destacando que:

"Se sintió que separaron tareas y tenían perfiles bien definidos. Eso hizo fácil la comunicación dado que cuando un integrante del equipo te chateaba/hablaba sabías de qué temática iba la charla. En todo momento se manejaron muy profesionalmente: cuando necesitaron algo, lo pidieron; siempre fueron super concretos en las reuniones, teniendo una agenda clara de lo que querían hablar."

Sobre el producto final el cliente concluyó que:

"El resultado es compacto, está alineado con lo que se pidió y más importante, con aquellos requerimientos implícitos por el contexto del proyecto (UI, comportamientos esperables de ciertas features, etc.). Hubo una gestión y de mi lado sentí que tenían el control del proyecto".

En conclusión, la evaluación del cliente confirma un alto nivel de satisfacción con el producto desarrollado, evidenciando que la plataforma cumple efectivamente con las expectativas establecidas y demuestra potencial para su adopción en comunidades reales

de bonsaistas. Los comentarios del cliente no solo validan la calidad técnica y funcional del sistema, sino que también reconocen la profesionalidad del equipo y la efectividad de la metodología de trabajo aplicada, estableciendo una base sólida para futuras colaboraciones y el desarrollo continuo de la plataforma.

Los resultados completos de la encuesta pueden observarse en el Anexo 13.10.7.

10. Gestión de la configuración

10.1. Desarrollo de software

Para el desarrollo de software, se adoptó *Git* como sistema de control de versiones y *GitHub* como plataforma de hospedaje y colaboración. Se estableció una arquitectura de repositorios distribuida, creando un repositorio independiente para cada aplicación del sistema: *niche-api* para el backend, *niche-rag* para el servicio de inteligencia artificial, *niche-web* para el *frontend* y *niche-infrastructure* para la infraestructura como código. Esta separación permitió gestionar de manera independiente el ciclo de vida de cada componente, facilitando la implementación de *Pull Requests* para revisión de código, el versionado específico de cada aplicación y la configuración de pipelines automatizados de integración y despliegue continuo mediante *GitHub Actions* para los componentes de *backend* e infraestructura, y *Vercel* para el *frontend*.

10.1.1. Control de versiones y cambios

El control de versiones se implementó mediante un sistema de tags de release que proporciona trazabilidad completa de las versiones desplegadas en producción. Cada release se identifica mediante un tag semántico que sigue el formato *x.y.z*, donde *x* representa cambios mayores, *y* cambios menores y *z* correcciones de errores. Para documentar los cambios entre versiones, se implementó un archivo *Changelog.md* en los repositorios de *Backend* lógico y *Frontend*, donde se registran de manera estructurada todas las modificaciones, nuevas funcionalidades, correcciones de errores y cambios que rompen la compatibilidad. Esta documentación se mantiene actualizada durante el desarrollo y se consolida al momento de crear cada release, proporcionando una referencia clara de la evolución del sistema.

10.1.2. Nombramiento de ramas

Se estableció una convención de nombramiento de ramas que facilita la identificación del propósito y origen de cada rama de desarrollo. La rama *main* se reservó exclusivamente para el código de producción, manteniendo siempre una versión estable y desplegable del sistema, mientras que la rama *develop* funciona como rama de integración para el entorno de *staging*. Para las ramas de desarrollo de funcionalidades, se implementó una

convención que combina las iniciales del desarrollador, el número del ticket de *Linear* y una descripción breve, ejemplificada como *"ml/llm-123/user-authentication"*. Esta convención se integró directamente con el sistema de gestión de proyectos *Linear*, permitiendo la automatización del movimiento de tarjetas entre columnas del tablero *Kanban*: cuando un desarrollador crea una rama siguiendo esta convención, el sistema automáticamente mueve la tarjeta correspondiente de *"To Do"* a *"In Progress"*, luego a *"Code Review"* cuando se crea el *Pull Request*, y finalmente a *"Done"* cuando se completa el merge.

10.1.3. Procedimientos de release

El proceso de release se diseñó como un flujo estructurado que garantiza la calidad y estabilidad de cada versión liberada a producción. El procedimiento comienza con la creación de una rama de release desde *develop* siguiendo la convención *"release-x.x.x"*, donde se actualiza el número de versión en el archivo *package.json* y se ejecuta un proceso de testeo funcional completo que abarca todas las funcionalidades del sistema.

Para realizar pruebas en producción sin afectar las comunidades existentes, se creó una comunidad de demostración denominada *"Demoland"* que replica la funcionalidad completa del sistema en un entorno controlado. Una vez completadas las pruebas, se procede a la publicación de la rama mediante un *Pull Request* hacia *main* con el nombre *"Promoting develop to main as release-x.x.x"*, el cual requiere la resolución de conflictos y la aprobación de al menos dos miembros del equipo.

Cuando se aprueba y mergea el *Pull Request*, se ejecuta automáticamente la creación de un nuevo tag de release y la generación de documentación en *GitHub* con los cambios implementados, activando los workflows de *GitHub Actions* para el despliegue automatizado.

En caso de modificaciones en variables de entorno, se actualiza previamente el código de *Terraform* en el entorno correspondiente. Para *niche-api*, después del autodespliegue se accede mediante *Session Manager* a la instancia liberada para ejecutar las migraciones de base de datos, mientras que para *niche-web* se verifica manualmente en la página web los principales flujos de funcionamiento una vez finalizado el despliegue en *Vercel*.

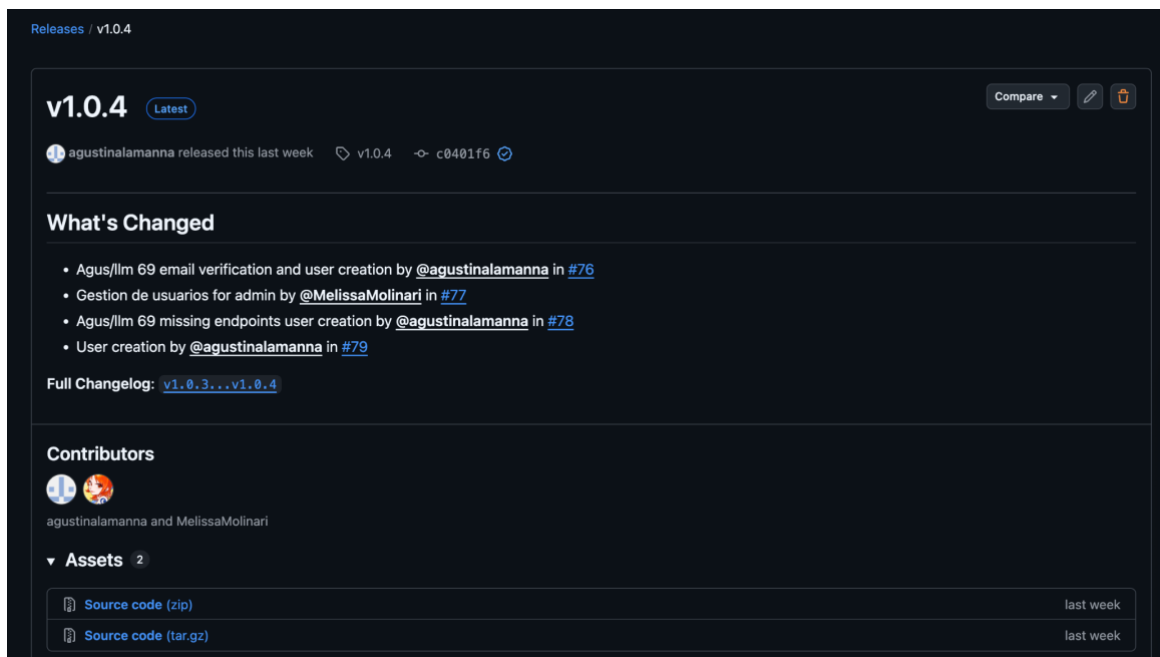


Ilustración 59: Documentación generada para el release de la versión 1.0.4 de Niche-API

10.1.4. Gestión de dependencias

La gestión de dependencias se implementó de manera diferenciada según las tecnologías utilizadas en cada componente del sistema, aprovechando las herramientas específicas de cada ecosistema para optimizar la resolución de dependencias y la seguridad del sistema.

En el repositorio *niche-api*, que utiliza *Node.js* con *TypeScript*, se implementó *pnpm* como gestor de paquetes principal, proporcionando beneficios significativos en términos de velocidad de instalación, eficiencia en el uso del espacio en disco y resolución optimizada de dependencias mediante su sistema de almacenamiento centralizado.

Para el componente *niche-rag* desarrollado en *Python*, se adoptó *pip* como gestor de paquetes estándar junto con un archivo *requirements.txt* que especifica versiones exactas de todas las dependencias, asegurando la consistencia entre entornos de desarrollo y producción.

En el *frontend* *niche-web*, construido con *Next.js* y *TypeScript*, también se implementó *pnpm* como gestor de paquetes principal, aprovechando sus capacidades de resolución de dependencias optimizada.

10.2. Documentación y gestión del proceso

10.2.1. Linear

Se utilizó Linear como herramienta central para la gestión del tablero *Kanban* y la administración de historias de usuario del proyecto. Esta plataforma permitió asignar tareas específicas a cada miembro del equipo, agregar estimaciones de esfuerzo en puntos de historia y, fundamentalmente, automatizar el movimiento de tarjetas mediante su integración directa con las ramas de *GitHub*.

Cuando un desarrollador creaba una rama siguiendo la convención establecida, la tarjeta correspondiente se movía automáticamente entre las columnas del tablero, eliminando la necesidad de actualización manual.

Se implementó un proceso de respaldo mensual del *backlog* completo, exportando desde *Linear* toda la información de tareas, historias de usuario y métricas acumuladas para garantizar la preservación del trabajo realizado.

Adicionalmente, *Linear* proporcionó capacidades valiosas para generar métricas rápidas del ciclo de desarrollo, permitiendo visualizar el trabajo asignado versus el completado, los puntos de esfuerzo asignados a cada integrante del equipo y la velocidad de desarrollo del proyecto, lo cual facilitó la identificación de cuellos de botella y el ajuste de estimaciones para futuras iteraciones.

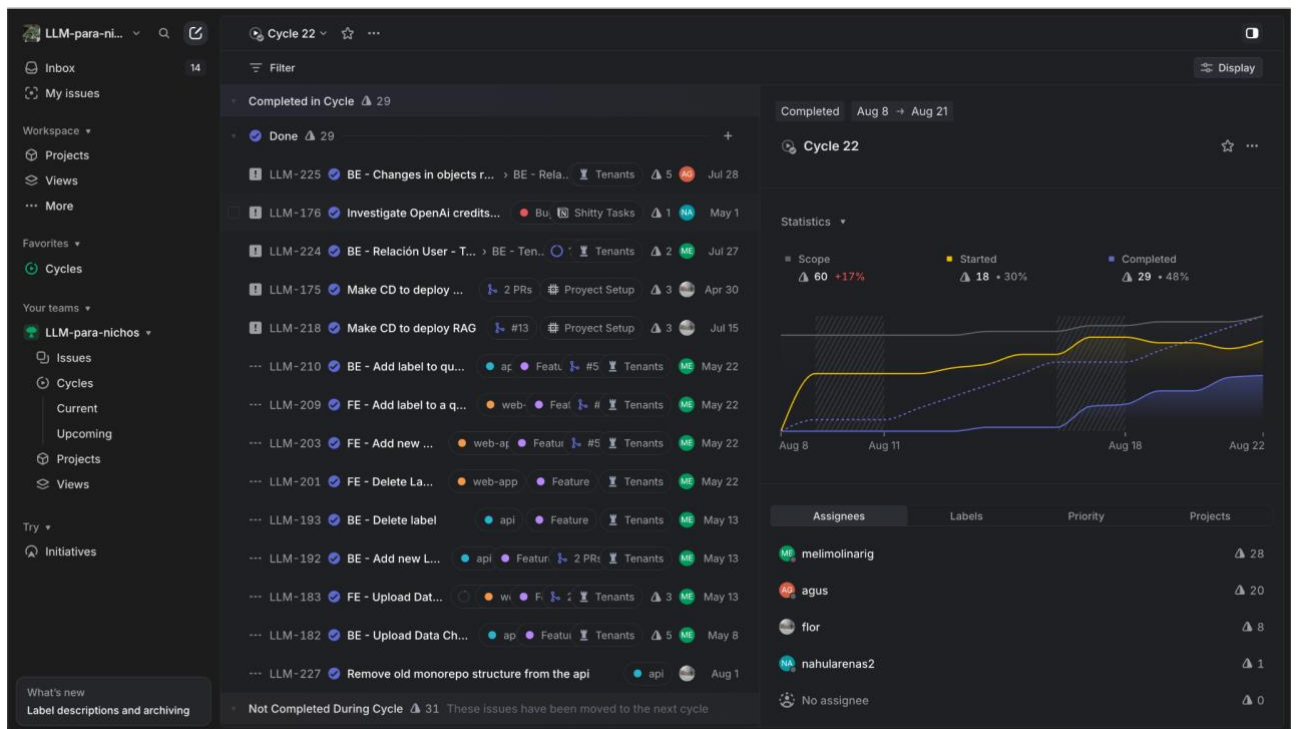


Ilustración 60: Análisis del ciclo 22 en Linear.

10.2.2. Google Drive

Google Drive se estableció como el repositorio central para la documentación del proyecto, aprovechando su capacidad para soportar una amplia variedad de formatos de documentos.

Esta flexibilidad fue particularmente valiosa para un proyecto que requería la gestión de diferentes tipos de documentación, incluyendo archivos de *Microsoft Word* para documentos formales y especificaciones, hojas de cálculo de *Excel* para casos de prueba y seguimiento de métricas, y herramientas de diagramación para la creación de diagramas técnicos y de arquitectura.

La colaboración en documentos se facilitó mediante las capacidades de edición simultánea de *Google Docs*, permitiendo que múltiples miembros del equipo trabajaran en la misma documentación de manera coordinada y en tiempo real.

10.2.3. Documentación generada

A continuación se presenta una tabla que especifica los documentos creados durante la duración del proyecto así como el código entregado al finalizar el proyecto.

Tipo	Elemento	Descripción
Documentos	Especificación de requerimientos	Documento que contiene los requerimientos funcionales y no funcionales del sistema.
Documentos	Informes de revisiones	Documentos con las conclusiones, puntos fuertes, débiles y accionables correspondientes a las tres revisiones del proyecto.
Documentos	Validación con usuarios	Documentos que contienen encuestas, análisis de resultados y conclusiones extraídas a partir del proceso de validación con usuarios potenciales y bonsaistas expertos.
Documentos	Entrevistas con usuarios	Documentos con transcripciones y conclusiones de entrevistas realizadas con expertos en bonsái y usuarios del nicho.
Documentos	Prototipos de UX/UI	Prototipos en Figma
Documentos	User Personas	Perfiles detallados del usuario objetivo
Documentos	Presentaciones de revisiones	Presentaciones utilizadas en cada revisión académica.
Documentos	Plan de calidad	Documento que detalla las fases del proyecto, actividades por fase, roles responsables, productos resultantes y mecanismos de aseguramiento de calidad.
Documentos	Gestión de riesgos	Documento que contiene los riesgos identificados, matriz de probabilidad e impacto, y estrategias de mitigación y acción para cada riesgo.
Documentos	Documentación de arquitectura	Documento que detalla la arquitectura del sistema, decisiones arquitectónicas, tácticas aplicadas y justificación técnica.
Documentos	Product Backlog y Tablero	Backlog en Linear priorizado de historias de usuario, tareas técnicas y bugs pendientes.

Documentos	Reuniones con clientes	Notas, acuerdos y conclusiones de reuniones periódicas con el cliente (Ing. Darío Macchi).
Documentos	Cronogramas	Planificación temporal del proyecto con hitos, entregables y seguimiento de avances.
Documentos	Casos de prueba	Planillas con casos de prueba diseñados para validación de funcionalidades, incluyendo pruebas unitarias, de integración y de usabilidad.
Documentos	Bibliografía	Listado de papers, artículos, documentación técnica y recursos utilizados como fuente bibliográfica para el proyecto.
Código	POCs y Spikes tecnológicos	Pruebas de concepto realizadas para validar decisiones tecnológicas (proveedores LLM, bases de datos vectoriales, frameworks).
Código	Niche-API	Código del backend en Node.js/TypeScript con Express, Prisma ORM, estructura modular (controllers, services, repositories), Dockerfile, dependencias (package.json), workflows de CI con GitHub Actions y pipelines de testing automatizado.
Código	Niche-RAG	Código del servicio RAG en Python, implementación de agentic graph, integración con LLM (OpenAI), <i>embeddings</i> , cliente de Pinecone, evaluación de respuestas con Opik, Dockerfile y dependencias.
Código	Niche-Infrastructure	Código de Infrastructure as Code con Terraform, módulos para VPC, ECS, RDS, ALB, ECR, Session Manager, configuración de ambientes y scripts de despliegue.
Código	Niche-FrontEnd	Código del <i>frontend</i> en <i>Next.js</i> con TypeScript, componentes React, integración con API backend, sistema de temas <i>multi-tenant</i> , Tailwind CSS y dependencias.

Tabla 10: Documentación generada durante el proyecto.

11. Conclusiones del proyecto

El proyecto demuestra la viabilidad de integrar modelos de lenguaje de gran escala (LLMs) dentro de una arquitectura avanzada de recuperación aumentada (RAG), ofreciendo una solución efectiva ante la falta de información precisa y contextualizada en dominios de conocimiento altamente especializados.

Como resultado, se desarrolló un Producto Mínimo Viable (MVP) robusto, funcional y listo para su uso, que cumple ampliamente con los objetivos académicos y de desarrollo propuestos. Este MVP establece una base tecnológica sólida sobre la cual pueden construirse futuras extensiones y mejoras, tanto en términos de capacidad de respuesta como de escalabilidad y adaptación a nuevos dominios de conocimiento.

11.1. Conclusiones sobre Objetivos

En esta sección se resumen los resultados obtenidos en relación con los objetivos académicos, del producto y del proceso.

11.1.1. Objetivos Académicos

Desde una perspectiva académica, el proyecto constituyó una experiencia formativa integral, en la que los conocimientos adquiridos a lo largo de la carrera de Ingeniería en Sistemas se aplicaron de manera coordinada y orientada a la resolución de un problema real. La experiencia permitió no solo poner en práctica competencias técnicas vinculadas al diseño e implementación de sistemas complejos, sino también a habilidades de gestión, liderazgo y trabajo colaborativo que resultaron determinantes para alcanzar los objetivos propuestos.

La autonomía del equipo fue un factor central, los integrantes asumieron la responsabilidad de organizar el cronograma de trabajo, priorizar tareas y tomar decisiones críticas sobre la selección de tecnologías, la arquitectura del sistema y la metodología de desarrollo. Este proceso de autogestión favoreció la madurez profesional del grupo, fortaleciendo capacidades de análisis, planificación y resolución de problemas en contextos reales de incertidumbre.

Asimismo, se aplicaron las buenas prácticas de ingeniería de software y gestión de proyecto aprendidas durante la carrera, integrando conceptos de análisis de requerimientos, modelado de sistemas, aseguramiento de calidad y métricas de desempeño. La articulación de estas prácticas dentro de un marco metodológico ágil evidenció la importancia de combinar rigor técnico con la flexibilidad necesaria para adaptarse a la evolución natural de un proyecto.

Por otra parte, el proyecto implicó una profundización técnica en el campo de los LLMs y las arquitecturas de recuperación aumentada de generación (RAG). El equipo logró una comprensión integral y actualizada del estado del arte en modelos de lenguaje y enfoques RAG, resultado de un proceso de investigación sistemática y de validación práctica. Esto permitió comprender de forma empírica las capacidades, limitaciones y oportunidades de cada arquitectura. Este aprendizaje contribuyó no solo a la consolidación del MVP desarrollado, sino también al crecimiento académico y profesional del equipo en el ámbito emergente de la inteligencia artificial aplicada.

En conjunto, el trabajo representó una experiencia formativa completa, donde la aplicación de conocimientos teóricos, metodológicos y tecnológicos se tradujo en resultados concretos y medibles, reforzando la preparación profesional de los integrantes para enfrentar proyectos reales en el ámbito tecnológico.

11.1.2. Objetivos del Producto

Desde la perspectiva del producto, la plataforma desarrollada responde de manera efectiva a la problemática identificada por la comunidad de bonsaistas, logrando un equilibrio adecuado entre funcionalidad, precisión y usabilidad. El resultado es una plataforma operativa, medible y confiable, capaz de generar respuestas especializadas basadas en información validada.

El mecanismo de evaluación y trazabilidad implementado permitió cuantificar objetivamente la calidad de las respuestas, evidenciando una reducción significativa de alucinaciones y ambigüedades, típicas de los modelos de lenguaje. Además, la incorporación de variables contextuales al modelo, como la ubicación del usuario, mejoró de manera notable la pertinencia y utilidad de las recomendaciones ofrecidas.

La adopción de *Infrastructure as Code* (IaC) mediante *Terraform* aseguró la escalabilidad, flexibilidad y reproducibilidad del MVP. Gracias a su arquitectura *multi-tenant*, la plataforma puede incorporar nuevos dominios de conocimiento sin necesidad de modificar su estructura central, lo que facilita su expansión hacia otros nichos especializados, como la apicultura o la enología.

La validación empírica realizada con usuarios finales y expertos confirmó la pertinencia, usabilidad y valor práctico del sistema. Este proceso de coevaluación iterativa permitió refinar el prototipo antes de su despliegue definitivo, asegurando que la solución no solo alcanzara solidez técnica, sino que también resultara relevante y significativa para su comunidad de usuarios.

Finalmente, el interés demostrado por un reconocido bonsaista y docente en explorar el uso de la plataforma en su escuela, refuerza el potencial de adopción del sistema en contextos reales de conocimiento especializado, validando su utilidad más allá del ámbito académico.

11.1.3. Objetivos del Proceso

El cumplimiento exitoso de los objetivos técnicos y funcionales se logró gracias a una metodología de gestión incremental iterativa, que combinó rigor técnico con flexibilidad organizacional.

Durante las fases de desarrollo, se aplicó un enfoque híbrido de gestión ágil: el proyecto comenzó bajo *Scrum*, promoviendo la iteración temprana y la retroalimentación continua, y posteriormente evolucionó hacia *Kanban*, priorizando un flujo constante y la entrega progresiva de valor. La planificación se apoyó en métricas objetivas, como *Cycle Time* y *Throughput*, que permitieron evaluar la productividad y tomar decisiones fundamentadas en evidencia.

La gestión proactiva de riesgos se mantuvo como una práctica constante, orientada a minimizar la incertidumbre tecnológica y a controlar la complejidad arquitectónica inherente a los sistemas basados en IA. Se realizaron spikes técnicos y se elaboró un Plan de Calidad exhaustivo, que sirvió como guía para evaluar decisiones críticas de diseño e integración.

Asimismo, la coordinación efectiva del equipo y la sinergia entre sus integrantes resultaron determinantes para el éxito del proyecto. La clara asignación de roles y la especialización en infraestructura cloud e IaC permitieron enfrentar con solvencia los desafíos del despliegue, garantizando una implementación robusta, escalable y alineada con los estándares de calidad establecidos.

11.2. Estado actual

La plataforma se desarrolló exitosamente como un Producto Mínimo Viable (MVP), orientado inicialmente a la creación de una comunidad para bonsaistas, pero diseñado con la flexibilidad necesaria para adaptarse a otros nichos especializados. Su arquitectura garantiza tanto la segregación de datos como la escalabilidad, permitiendo la incorporación de nuevas comunidades de conocimiento sin requerir modificaciones estructurales significativas.

El sistema se fundamenta en un *Large Language Model* (LLM) como componente central y combina dos enfoques de recuperación de información: RAG, basado en grafos de estado jerárquicos, y *Agentic RAG*, orientado a mejorar la precisión y trazabilidad de las respuestas dentro del dominio.

La arquitectura implementada contenerizada y orquestada en la nube asegura portabilidad, mantenibilidad y rendimiento, cumpliendo con los objetivos planteados para esta etapa del proyecto y estableciendo una base sólida para su futura expansión hacia otros dominios especializados.

11.3. Lecciones aprendidas

El desarrollo de la plataforma, dada su naturaleza exploratoria, permitió generar aprendizajes clave que abarcan aspectos metodológicos, técnicos y de gestión, consolidando la experiencia adquirida a lo largo del proyecto.

Entre las primeras lecciones, se destaca el valor de la interacción directa con los usuarios finales del nicho, que permitió comprender de manera más profunda las necesidades reales del dominio. Este contacto reveló la importancia de validar los requerimientos desde la experiencia concreta de uso, ajustando funcionalidades y prioridades en función de la práctica y no solo de las suposiciones iniciales del cliente o del equipo. De haberse

seguido únicamente los requerimientos definidos en las etapas iniciales, el producto difícilmente habría alcanzado el nivel de pertinencia y utilidad logrado tras este proceso de validación continua.

A su vez, el equipo tuvo la oportunidad de vivir de cerca la importancia de la gestión de procesos y la definición temprana de un plan de calidad para asegurar el proceso y seguirlo rigurosamente. La implementación del módulo de inteligencia artificial requirió constante experimentación, lo que evidenció que las metodologías deben adaptarse al contexto específico del proyecto: cuando las tareas son más exploratorias e inciertas, un enfoque de flujo continuo como *Kanban* puede ser más efectivo que *sprints* fijos.

En este proceso, el equipo identificó oportunidades de mejora, especialmente en la necesidad de fortalecer el seguimiento sistemático de métricas de calidad y los procesos de mejora continua. Esta lección permitió reconocer el valor de establecer mecanismos tempranos de control y evaluación para optimizar la trazabilidad y la eficiencia en futuras implementaciones.

Además, el equipo valora el conocimiento técnico adquirido durante el desarrollo del proyecto, especialmente considerando que se trataba de tecnologías emergentes en las cuales ningún integrante tenía experiencia previa. La experiencia permitió identificar la capacidad de adaptación y aprendizaje del equipo para dominar tecnologías como bases de datos vectoriales, sistemas RAG, e IaC con *Terraform*.

Por otra parte, esta experiencia constituyó la primera oportunidad del equipo de trabajar con un cliente real, lo que generó aprendizajes significativos en la gestión de relaciones cliente-equipo. Se desarrolló la capacidad de interactuar profesionalmente con *stakeholders* externos, manejando situaciones de desacuerdo sobre el producto de manera diplomática y constructiva.

11.4. Proyección a futuro

La proyección del proyecto se centra en la escalabilidad y expansión hacia nuevos nichos de conocimiento, validando el diseño *multi-tenant* y la modularidad de la arquitectura como pilares para su evolución.

Expansión a nuevos nichos

La plataforma fue concebida como un MVP diseñado para adaptarse a diferentes dominios, como apicultura, viticultura, astronomía amateur o normativas profesionales. Este modelo de Comunidades, entendido como silos de conocimiento independientes, permite la incorporación de nuevos dominios sin requerir cambios estructurales significativos en la arquitectura, facilitando así la rápida escalabilidad y diversificación de la plataforma.

Evolución del sistema de IA

Se proyecta mejorar de manera continua la precisión y confiabilidad del LLM, aprovechando la implementación experimental del *Agentic RAG* como base para desarrollar capacidades de razonamiento autónomo más sofisticadas. La arquitectura modular, con adaptadores bien definidos (RNF8), garantiza la posibilidad de sustituir proveedores de servicios, como LLMs, bases vectoriales o *embeddings*, para optimizar costos y rendimiento, asegurando que la plataforma pueda mantenerse tecnológicamente actualizada.

Refinamiento funcional

La estructura modular (RNF9) asegura que la incorporación de nuevas funcionalidades, tales como herramientas de administración de contenido o personalización de la experiencia de usuario, pueda realizarse con un bajo impacto en el código base. Esto proporciona flexibilidad para evolucionar la plataforma de manera sostenida y segura, respondiendo a necesidades emergentes y a la diversificación de las comunidades de conocimiento.

12. Referencias bibliográficas

- [1] FAO, “Manual técnico de apicultura: abeja (*Apis mellifera*),” *fao.org*, 2022. [En línea]. Disponible: <https://www.fao.org/family-farming/detail/es/c/1640691/>. [Accedido: 14-oct-2025].
- [2] P. Chan, *Bonsai: The Art of Growing and Keeping Miniature Trees*. Skyhorse, 2014.
- [3] R. C. Martin, *Clean Architecture: A Craftsman’s Guide to Software Structure and Design*. Pearson Education, 2017.
- [4] T. Brown, *Change by Design: How Design Thinking Transforms Organizations and Inspires Innovation*. HarperBusiness, 2009.
- [5] M. Cagan, *INSPIRED: How to Create Tech Products Customers Love*, 2nd ed. Wiley, 2017.
- [6] J. Barnard, “What Is Embedding?,” *IBM Think*, 2025. [En línea]. Disponible: <https://www.ibm.com/think/topics/embedding>. [Accedido: 10-oct-2025].
- [7] *Figma*, “FigJam,” 2025. [En línea]. Disponible: <https://www.figma.com/figjam/>. [Accedido: 10-oct-2025].
- [8] Ministerio de Agricultura, Ganadería y Alimentación de Guatemala (MAGA), “¿Qué es la horticultura?,” *MAGA*, 2025. [En línea]. Disponible: <https://www.maga.gob.gt/que-es-la-horticultura/>. [Accedido: 11-oct-2025].
- [9] K. Morris, *Infrastructure as Code*. O’Reilly, 2016.
- [10] *LangChain*, “LangChain Documentation,” 2025. [En línea]. Disponible: <https://python.langchain.com/>. [Accedido: 04-oct-2025].
- [11] B. Clark, “What is LangGraph?,” *IBM Think*, 2025. [En línea]. Disponible: <https://www.ibm.com/think/topics/langgraph>. [Accedido: 13-oct-2025].
- [12] *IBM*, “What Are Large Language Models?,” *IBM Think*, 2025. [En línea]. Disponible: <https://www.ibm.com/think/topics/large-language-models>. [Accedido: 12-oct-2025].

- [13] *Atlassian*, “Minimum Viable Product (MVP),” *Atlassian Agile Coach*, 2025. [En línea]. Disponible: <https://www.atlassian.com/agile/product-management/minimum-viable-product>. [Accedido: 01-oct-2025].
- [14] *ICZN*, “International Code of Zoological Nomenclature (Online),” *iczn.org*, 2025. [En línea]. Disponible: <https://www.iczn.org/the-code/the-international-code-of-zoological-nomenclature/>. [Accedido: 01-oct-2025].
- [15] *Pinecone*, “Pinecone Docs,” *docs.pinecone.io*, 2025. [En línea]. Disponible: <https://docs.pinecone.io/>. [Accedido: 02-oct-2025].
- [16] K. Schwaber y J. Sutherland, *The 2020 Scrum Guide*, 2020. [En línea]. Disponible: <https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-US.pdf>. [Accedido: 02-oct-2025].
- [17] *Glossaire Wein-Plus*, “Procesos enológicos — vinificación,” 2021. [En línea]. Disponible: <https://glossaire.wein.plus/procesos-enologicos>. [Accedido: 06-oct-2025].
- [18] *Google Cloud*, “Retrieval-Augmented Generation — use cases,” 2025. [En línea]. Disponible: <https://cloud.google.com/use-cases/retrieval-augmented-generation>. [Accedido: 05-oct-2025].
- [19] R. S. Pressman y B. R. Maxim, *Software Engineering: A Practitioner’s Approach*, 8th ed. McGraw-Hill, 2015.
- [20] M. Cohn, *Agile Estimating and Planning*. Prentice Hall, 2005.
- [21] *University of California, Berkeley*, “Clasificación según Linneo,” *Understanding Evolution*, 2025. [En línea]. Disponible: <https://evolution.berkeley.edu/glossary-spanish/clasificacion-linneo/>. [Accedido: 11-oct-2025].
- [22] *HashiCorp*, “Terraform (Docs),” 2025. [En línea]. Disponible: <https://developer.hashicorp.com/terraform/docs>. [Accedido: 13-oct-2025].
- [23] *PMI*, “The Objective of Time-Boxing,” *Disciplined Agile*, 2025. [En línea]. Disponible: <https://www.pmi.org/disciplined-agile/the-objective-of-time-boxing>. [Accedido: 06-oct-2025].

[24] *OpenAI*, “What are tokens and how to count them,” *OpenAI Help Center*, 2025. [En línea]. Disponible: <https://help.openai.com/en/articles/4936856-what-are-tokens-and-how-to-count-them>. [Accedido: 06-oct-2025].

[25] C. LeRouge, J. Ma, S. Sneha y K. Tolle, “User profiles and personas in the design and development of consumer health technologies,” *Int. J. Med. Inform.*, vol. 82, no. 11, pp. e251–e268, nov. 2013.

[26] N. Mulla y S. Girase, “A New Approach to Requirement Elicitation Based on Stakeholder Recommendation,” *Int. J. Softw. Eng. Appl.*, vol. 3, no. 3, pp. 51–57, may. 2012. [En línea]. Disponible: <https://d1wqtxts1xzle7.cloudfront.net/104509184/006d498b0c928db7dcb87a4c7b30155c72f6-libre.pdf>. [Accedido: 07-oct-2025].

[27] Redacción Clarín, “Secretos de la jardinería oriental: ¿cada cuánto debe regarse un bonsái?,” *Clarín*, 07-ene-2024. [En línea]. Disponible: https://www.clarin.com/internacional/secretos-jardineria-oriental-debe-regarse-bonsai_0_qptL4LWRCO.html. [Accedido: 15-sept-2025].

[28] “¿Cuántas veces se le echa agua a un bonsái? Descubre el secreto para un árbol sano y feliz,” *Habibi*, 2025. [En línea]. Disponible: https://www.habibiplantitas.com/blogs/news/cuantas-veces-se-le-echa-agua-a-un-bonsai-descubre-el-secreto-para-un-arbol-sano-y-feliz?srsId=AfmBOoold3W_IMQwVBTTu2k0zSENkjflWYI7VThutlST6D3Pywu9mE2e. [Accedido: 15-sept-2025].

[29] *PMI, A Guide to the Project Management Body of Knowledge (PMBOK® Guide) – Seventh Edition*. Project Management Institute, 2021.

[30] N. Prajapati, “RAG vs. Fine-Tuning: LLM adaptation for businesses,” *Softweb Solutions*, 06-nov-2023. [En línea]. Disponible: <https://www.softwebsolutions.com/resources/rag-vs-fine-tuning/>. [Accedido: 10-oct-2025].

[31] M. Andersen, “How I Create a Cloud Architecture Diagram: My Step-by-Step Guide,” *Medium*, 11-ago-2025. [En línea]. Disponible:

<https://medium.com/@andersenmikkel/how-i-create-a-cloud-architecture-diagram-my-step-by-step-guide-df044aacf4b5>. [Accedido: 16-sept-2025].

[32] R. C. Martin, *Clean Architecture*. Prentice Hall, 2017.

[33] *Prisma.io*, “Blocked Page,” 2025. [En línea]. Disponible: <https://www.prisma.io/docs/orm/overview/prisma-in-your-stack/rest>. [Accedido: 19-sept-2025].

[34] *Microsoft*, “GitHub - microsoft/tsyringe: Lightweight dependency injection container for JavaScript/TypeScript,” *GitHub*, 2017. [En línea]. Disponible: <https://github.com/microsoft/tsyringe>. [Accedido: 01-oct-2025].

[35] J. Shore, “Fail Fast,” *Martin Fowler*, sep. 2004. [En línea]. Disponible: <https://martinfowler.com/ieeeSoftware/failFast.pdf>. [Accedido: 15-sept-2025].

[36] R. N. Charette, *Software Engineering Risk Analysis and Management*. McGraw-Hill, 1989.

[37] G. G. Schulmeyer, *Handbook of Software Quality Assurance*. Artech House, 2008.

[38] A. Álvarez y M. López, “Elaboración de planes de la calidad en proyectos de software.” [En línea]. Disponible: https://sedici.unlp.edu.ar/bitstream/handle/10915/23062/Documento_completo.pdf?sequence=1. [Accedido: 15-sept-2025].

[39] S. Carroll y T. Daughtrey, *Fundamental Concepts for the Software Quality Engineer*. Quality Press, 2007.

[40] *OpenAI*, “OpenAI Platform,” *Openai.com*, 2024. [En línea]. Disponible: <https://platform.openai.com/docs/overview>. [Accedido: 06-oct-2025].

[41] *Google*, “Gemini API Overview,” *Ai.google.dev*, 2024. [En línea]. Disponible: <https://ai.google.dev/>. [Accedido: 06-oct-2025].

[42] *Google*, “Gemini API Pricing,” *Ai.google.dev*, 2024. [En línea]. Disponible: <https://ai.google.dev/pricing>. [Accedido: 06-oct-2025].


- [43] *Anthropic*, “Claude Overview,” *Anthropic.com*, 2024. [En línea]. Disponible: <https://www.anthropic.com/claude>. [Accedido: 06-oct-2025].
- [44] *Meta*, “LLaMA: Open Foundation and Research Models,” *Llama.com*, 2024. [En línea]. Disponible: <https://www.llama.com/>. [Accedido: 06-oct-2025].
- [45] *Cohere*, “Command R API Documentation,” *Cohere.com*, 2024. [En línea]. Disponible: <https://docs.cohere.com/v2/reference/about>. [Accedido: 06-oct-2025].
- [46] *Google*, “Gemini API Model Tuning Guide,” *Ai.google.dev*, 2024. [En línea]. Disponible: <https://ai.google.dev/gemini-api/docs/model-tuning>. [Accedido: 06-oct-2025].
- [47] *Pandas*, “pandas - Python Data Analysis Library,” *pandas.pydata.org*, 2024. [En línea]. Disponible: <https://pandas.pydata.org/about/index.html>. [Accedido: 06-oct-2025].
- [48] M. Ali, “Top Python Libraries for Data Science in 2022,” *Datacamp.com*, ene. 2024. [En línea]. Disponible: <https://www.datacamp.com/blog/top-python-libraries-for-data-science>. [Accedido: 06-ene-2025].
- [49] M. Ali, “Top Python Libraries for Data Science in 2022,” *Datacamp.com*, ene. 2024. [En línea]. Disponible: <https://www.datacamp.com/blog/top-python-libraries-for-data-science>. [Accedido: 06-ene-2025].
- [50] *Matplotlib*, “Matplotlib: Python Plotting — Matplotlib 3.1.1 Documentation,” *Matplotlib.org*, 30-may-2024. [En línea]. Disponible: <https://matplotlib.org/>. [Accedido: 06-ene-2025].
- [51] “What are the advantages of Scikit-learn over other machine learning frameworks?,” *LinkedIn*, 2025. [En línea]. Disponible: <https://www.linkedin.com/advice/1/what-advantages-scikit-learn-over-other-machine-learning-d8kse>. [Accedido: 07-ene-2025].
- [52] *PyPI*, “scikit-learn,” *PyPI*, mar. 2019. [En línea]. Disponible: <https://pypi.org/project/scikit-learn/>. [Accedido: 07-ene-2025].
- [53] *NLTK*, “Natural Language Toolkit — NLTK 3.4.4 documentation,” *Nltk.org*, 2009. [En línea]. Disponible: <https://www.nltk.org/>. [Accedido: 07-ene-2025].

- [54] P. en, “Programación en Python - 3. Natural Language Toolkit,” *Google.com*, 2025. [En línea]. Disponible: <https://sites.google.com/view/programacion-en-python/home/3-natural-language-toolkit>. [Accedido: 07-ene-2025].
- [55] M. Ali, “Tutorial de análisis NLTK del sentimiento para principiantes,” *Datacamp.com*, 22-ago-2024. [En línea]. Disponible: <https://www.datacamp.com/es/tutorial/text-analytics-beginners-nltk>. [Accedido: 21-ene-2025].
- [56] *Comet-ml*, “GitHub - comet-ml/opik: Debug, evaluate, and monitor your LLM applications, RAG systems, and agentic workflows,” *GitHub*, 15-oct-2025. [En línea]. Disponible: <https://github.com/comet-ml/Opik>. [Accedido: 21-ene-2025].
- [57] F. Gamba, “Open-Source LLM Evaluation Platform | Opik by Comet,” *Comet*, 10-sep-2024. [En línea]. Disponible: <https://www.comet.com/site/products/opik/>. [Accedido: 21-ene-2025].
- [58] “Opik LLM Development Platform | Observability, Evaluation & Security,” *Opik Documentation*, 2025. [En línea]. Disponible: <https://www.comet.com/docs/opik/>. [Accedido: 21-ene-2025].
- [59] *Comet-ml*, “opik/sdks/python/README.md at main · comet-ml/opik,” *GitHub*, 2023. [En línea]. Disponible: <https://github.com/comet-ml/opik/blob/main/sdks/python/README.md>. [Accedido: 21-ene-2025].
- [60] J. Verre, “Javascript SDK,” *GitHub*, 12-dic-2024. [En línea]. Disponible: <https://github.com/comet-ml/opik/issues/876>. [Accedido: 07-ene-2025].
- [61] “Aprenda a usar Gensim, una popular biblioteca de Python para el modelado de temas y el análisis de similitud en NLP,” *LinkedIn*, 14-mar-2024. [En línea]. Disponible: <https://www.linkedin.com/advice/0/how-can-you-use-gensim-topic-modeling-similarity-0e2tf?lang=es>. [Accedido: 07-ene-2025].
- [62] *Google for Developers*, “TensorFlow.js: Crea un sistema de detección de spam y comentarios,” 2021. [En línea]. Disponible: <https://developers.google.com/codelabs/tensorflowjs-comment-spam-detection?hl=es-419#0>. [Accedido: 07-ene-2025].

[63] *TensorFlow*, “TensorFlow.js | Aprendizaje automático para desarrolladores de JavaScript,” 2025. [En línea]. Disponible: <https://www.tensorflow.org/js?hl=es-419>. [Accedido: 07-ene-2025].

[64] *Google Developers*, “Presentamos Danfo.js, una biblioteca similar a Pandas en JavaScript,” 2020. [En línea]. Disponible: <https://developers-latam.googleblog.com/2020/09/presentamos-danfojs-una-biblioteca.html>. [Accedido: 07-ene-2025].

[65] K. Jagota, “7 JavaScript Libraries for Data Manipulation, Visualization and Machine Learning,” *PlainEnglish.io*, 2025. [En línea]. Disponible: <https://plainenglish.io/blog/7-javascript-libraries-for-data-manipulation-visualization-and-machine-learning>. [Accedido: 07-ene-2025].

[67] “Build a Retrieval Augmented Generation (RAG) App |  LangChain,” *Langchain.com*, 2023. [En línea]. Disponible: <https://python.langchain.com/docs/tutorials/rag/>. [Accedido: 25-nov-2024].

[68] “Quick Start,” *Haystack*, 2025. [En línea]. Disponible: <https://haystack.deepset.ai/overview/quick-start>. [Accedido: 25-nov-2024].

[69] “Creating Your First QA Pipeline with Retrieval-Augmentation | Haystack,” *Haystack*, 2025. [En línea]. Disponible: https://haystack.deepset.ai/tutorials/27_first_rag_pipeline. [Accedido: 25-nov-2024].

[70] “Always Retrieval Augment Your Large Language Models,” *Deepset.ai*, 2025. [En línea]. Disponible: <https://www.deepset.ai/blog/llms-retrieval-augmentation>. [Accedido: 25-nov-2024].

[71] M. Pietsch *et al.*, “Haystack: the end-to-end NLP framework for pragmatic builders,” *GitHub*, 01-nov-2019. [En línea]. Disponible: <https://github.com/deepset-ai/haystack>. [Accedido: 25-nov-2024].

[72] *AssemblyAI*, “Vector Databases simply explained! (Embeddings & Indexes),” *YouTube*, 06-may-2023. [En línea]. Disponible: <https://www.youtube.com/watch?v=dN0lsF2cvm4>. [Accedido: 25-nov-2024].

- [73] *MyScale*, “Understanding Vector Indexing: A Comprehensive Guide,” *Medium*, 13-feb-2024. [En línea]. Disponible: <https://medium.com/@myscale/understanding-vector-indexing-a-comprehensive-guide-d1abe36ccd3c>. [Accedido: 25-nov-2024].
- [74] “Build RAG with Milvus,” *Milvus Documentation*, 2025. [En línea]. Disponible: <https://milvus.io/docs/build-rag-with-milvus.md>. [Accedido: 25-nov-2024].
- [75] “RAG battle: vector database vs knowledge graph,” *FalkorDB*, 04-oct-2023. [En línea]. Disponible: <https://www.falkordb.com/blog/rag-battle-vector-database-vs-knowledge-graph/>. [Accedido: 25-nov-2024].
- [76] “Construct Knowledge Graphs From Unstructured Text,” *Neo4j Developer Blog*, 2024. [En línea]. Disponible: <https://neo4j.com/developer-blog/construct-knowledge-graphs-unstructured-text/>. [Accedido: 10-ene-2025].
- [77] “Constructing knowledge graphs | LangChain,” *Langchain.com*, 2024. [En línea]. Disponible: https://python.langchain.com/v0.1/docs/use_cases/graph/constructing/. [Accedido: 10-ene-2025].
- [78] *Neo4j*, “NaLLM,” *GitHub*, 05-feb-2024. [En línea]. Disponible: <https://github.com/neo4j/NaLLM>. [Accedido: 10-ene-2025].
- [79] *LangChain*, “Introduction | LangChain,” *Langchain.com*, 2024. [En línea]. Disponible: <https://python.langchain.com/docs/introduction/>. [Accedido: 19-ene-2025].
- [80] “Graphs | LangChain,” *Langchain.com*, 2025. [En línea]. Disponible: <https://python.langchain.com/docs/integrations/graphs/>. [Accedido: 19-ene-2025].
- [81] “Pinecone | LangChain,” *Langchain.com*, 2025. [En línea]. Disponible: <https://python.langchain.com/docs/integrations/providers/pinecone/>. [Accedido: 19-ene-2025].
- [82] “Qdrant | LangChain,” *Langchain.com*, 2025. [En línea]. Disponible: <https://python.langchain.com/docs/integrations/providers/qdrant/>. [Accedido: 19-ene-2025].

[83] “Weaviate | LangChain,” *Langchain.com*, 2025. [En línea]. Disponible: <https://python.langchain.com/docs/integrations/providers/weaviate/>. [Accedido: 19-ene-2025].

13. Anexos

13.1. Validación de la propuesta de valor

Pregunta	Escala	Respuesta más representativa
<i>Preguntas demográficas</i>		
¿Cuál es su edad?	18-17+	41-60 años (46,8%)
¿Con qué género se identifica?	Hombre; Mujer; Otro; Prefiero no decirlo	Hombre (83,8%)
¿Dónde reside actualmente?	Uruguay; Argentina; Brasil; Chile; Otro	Argentina (38,7%); Otro (32,4%)
¿Qué nivel educativo posee?	Secundaria completa / incompleta; Estudios terciarios / universitarios completos /incompletos; Estudios de posgrado	Estudios terciarios / universitarios completos (32,4%)
¿A qué se dedica actualmente?	Estudiante; trabajador en actividad; Jubilado; Otro	Trabajador en actividad (73,9%)
¿Qué tan familiarizado se encuentra con la tecnología o el uso de plataformas en línea?	Muy; Algo; Poco o Nada Familiarizado.	Muy familiarizado (53,2%)
¿Con qué frecuencia navega por foros o comunidades en línea?	Varias veces al día; Una vez al día; Varias veces a la semana; Una vez a la semana o menos	Varias veces al día (51,4%)
¿Qué dispositivos utiliza con mayor frecuencia para acceder a internet? Puedes seleccionar más de una opción	Celular; Computadora de escritorio; Laptop; Tablet	Celular (108 votos); Laptop (38 votos)
<i>Conocimiento y experiencia sobre Bonsáis</i>		
¿Hace cuánto tiempo los bonsáis son un tema de interés para ti, ya sea porque tienes uno o porque te gustaría aprender sobre su cuidado?	Menos de 1 año; Entre 1 y 3 años; Más de 3 años; Solo recientemente me empezó a interesar	Más de 3 años (84,7%)
¿Cuál es tu nivel de experiencia con el cuidado de plantas en general?	Principiante; Intermedio; Avanzado; Experto	Intermedio (46,8%)
¿Consideras que cuidar bonsáis requiere conocimientos especializados?	Si, es un tema complejo y específico; No, es relativamente sencillo; No estoy seguro	Si, es un tema complejo y específico (86,5%)

¿Con qué frecuencia buscas información sobre cómo cuidar bonsáis?	Frecuentemente; Ocasionalmente; Raramente; Nunca	Frecuentemente (79,3%)
<i>Fuentes actuales de información</i>		
¿Qué tan difícil es encontrar información de calidad sobre el cuidado de bonsáis?	1 (Muy sencillo) - 6 (Muy difícil)	4 (31,5%)
¿Qué te motiva a buscar más información sobre el cuidado de bonsáis?	Aprender más sobre un tema que me interesa; Resolver problemas con mis bonsais; Evitar cometer errores que afecten mis bonsais; Otros	Aprender más sobre un tema que me interesa (94 votos)
¿Qué fuentes sueles utilizar para buscar información sobre el cuidado de bonsáis?	Redes sociales (Facebook, Instagram, etc); Foros especializados; Videos de Youtube; Libros o bibliografía especializada; Amigos o expertos en bonsais; Asistentes de inteligencia artificial; Otros	Videos de YouTube (89 votos); Amigos o expertos en bonsáis (87 votos)
¿Qué beneficios esperas obtener al mejorar tu conocimiento en el cuidado de bonsáis?	Tener árboles más sanos; Mejorar mi técnica; Disfrutar más el proceso de cuidado; Compartir mi experiencia con otros; Otros	Mejorar mi técnica (82%)
¿Cómo calificarías la calidad de la información que encuentras en estas fuentes?	Muy buena, generalmente confiable; Buena, aunque a veces inconsistente; Regular, con frecuencia encuentro errores; Mala, es difícil encontrar información confiable	Buena, aunque a veces inconsistente (58,6%)
¿Has encontrado problemas al buscar información sobre bonsáis en las fuentes que actualmente consultas?	Si; No	Si (60,4%)
¿Qué problemas has encontrado al buscar información sobre bonsáis en las fuentes actuales?	Respuestas inconsistentes o contradictorias; Falta de profundidad en las explicaciones; Información poco confiable o sin respaldo; Dificultad para encontrar información específica	Falta de profundidad en las explicaciones (63 votos); Respuestas inconsistentes o contradictorias (53 votos)
¿Alguna vez aplicaste información que resultó ser incorrecta o dañina para tus bonsáis?	Si, y afectó negativamente mis bonsais; Si, pero no tuvo un impacto significativo; No, nunca he tenido este problema	Si, pero no tuvo un impacto significativo (43,2%)
Si aplicaste información incorrecta, ¿cómo te diste cuenta del error?	Note un deterioro en mis bonsais; Consulte con un experto y me corrigieron; Revise otras fuentes y encontré contradicciones; Otros	Revisé otras fuentes y encontré contradicciones (35,6%)

<i>Percepción de un asistente especializado</i>		
¿Te resultaría útil un asistente basado en inteligencia artificial que proporcione respuestas específicas y fundamentadas sobre el cuidado de bonsáis?	Si, definitivamente; No, no creo que lo necesite; Tal vez, pero dependería de su calidad	Sí, definitivamente (53,2%)
¿Qué tan útil te parecería que una comunidad experta pueda complementar o mejorar las respuestas proporcionadas por el asistente sobre el cuidado de bonsáis?	1 (Muy útil) - 5 (Nada útil)	1 (70,3%)
¿Qué características te gustaría que tenga un asistente especializado en bonsáis? Puedes seleccionar más de una opción	Respuestas claras y detalladas; Referencias confiables; Personalización según el tipo de bonsai; Otros	Respuestas claras y detalladas (88 votos); Referencias confiables que respalden la información (76 votos)
¿Qué nivel de confianza necesitas tener en las respuestas de un asistente para usarlo regularmente?	Muy alta (respuestas siempre precisas y fundamentadas); Alta (respuestas confiables, aunque ocasionalmente puedan tener errores); Media (información útil, pero con margen de error)	Muy alta (62,2%)
¿Te interesaría contribuir a la comunidad respondiendo preguntas de otros usuarios en la plataforma?	Si; No; Dependería de la recompensa	Si (82.9%)
¿Estarías dispuesto a pagar por un servicio con respuestas confiables y especializadas sobre bonsáis?	Si; No; Dependería del costo	Dependerá del costo (52.3%) Si (18.9%) No (28.8%)
Si estuvieras dispuesto a pagar, ¿qué modelo de pago preferirías?	Pago Único; Pago por consulta; Suscripción mensual; Otros	Pago único (32.3%) Suscripción mensual (34.4%) Pago por consulta (22.9%)
<i>Futuras etapas del proyecto</i>		
¿Te gustaría participar en futuras etapas de nuestro proyecto? Estas podrían incluir entrevistas o pruebas del sistema para ayudarnos a mejorar. Si estás interesado/a, por favor déjanos tu dirección de correo electrónico.		Se recibieron 53 respuestas.

Tabla 11: Validación de la propuesta de valor

13.2. Usuario objetivo



The illustration shows a user profile card for Raúl Martínez. On the left is a cartoon illustration of an elderly man with glasses, wearing a green sweater. The card contains the following information:

RAÚL MARTÍNEZ
68 años, Jubilado

Residencia: Paraguay
Educación: nivel terciario incompleto

Raúl esta jubilado hace un par de años. Comenzó una carrera la cual nunca terminó. Usa su celular o laptop para visitar foros o comunidades en línea y aprender de sus intereses.

"La información de internet está acorde, pero a veces se me hace poco confiable."

RELACIÓN CON BONSAÍIS

- Comenzó con el bonsaísmo cómo hobby hace tres años.
- Cometió errores leves en el cuidado de los bonsáis, solventado por expertos.

TECNOLOGÍAS

- Algo familiarizado con la tecnología
- Navega "una vez a la semana o menos"

NECESIDADES

- Poder encontrar información confiable.
- Encontrar información detallada con respaldo teórico.

QUÉ BUSCA EN EL ASISTENTE

- Respuestas de calidad, que estén basadas en bibliografía fiable.
- Respuestas completas, que logren ayudarlo a resolver sus dudas.

Ilustración 61: Usuario objetivo

13.3. Listado de requerimientos iniciales del cliente

13.3.1. Glosario/Definiciones

La idea es comentar qué es cada uno de estos términos en el contexto de este documento.

Tenant: un dominio de conocimiento; un silo de datos perteneciente a una institución/persona que no quiere que su información se “mezcle” con la información de otros silos.

Entrenar: estoy bastante seguro que no vamos a entrenar ningún modelo sino que vamos a utilizar otras estrategias para hacer llegar información que mejore la precisión de las respuestas de un modelo. A este ejercicio de “hacer llegar” dicha información le llamaremos entrenar.

Estados de una pregunta:

- AI MISS: estado en el que queda una pregunta si la AI no puede responder.
- AI ANSWERED: estado en el que queda una pregunta si la AI puede responder.
- AI HIT: estado en el que queda una pregunta cuando un bonsaista-junior acepta la respuesta a una pregunta.
- PUBLIC AVAILABLE: estado en el que queda una pregunta si el dev-admin no puede contestar la pregunta
- PUBLIC ANSWERED: estado en el que queda una pregunta cuando un bonsaista-junior contesta una pregunta.
- PUBLIC HIT: estado en el que queda una pregunta cuando un bonsaista-junior acepta la respuesta a una pregunta.

13.3.2. Actores

dev-admin: usuario técnico que llevará a cabo tareas de administración de la solución.

bonsaista-junior: usuario novato en bonsai que tiene dudas sobre cómo mantener vivo su árbol o tiene algunos conocimientos básicos que puede incluso compartir con otros usuarios tipo bonsaista-junior.

13.3.3. Funcionalidades

13.3.3.1. Funcionalidades CLI (o backend administrativo)

1. Ingest de datos

Para proveer a un tenant de datos, como dev-admin quiero cargar los chunks de datos desde una folder (en formato markdown) a la BD.

2. Hacer pregunta

Para hacer QA de respuestas, como dev-admin quiero escribir una pregunta en lenguaje natural que será respondida, enviando el contexto como un json (o string) y el tenant como parámetros.

3. Listar preguntas

Para hacer QA, analizar respuestas e identificar preguntas que necesiten edición, como dev-admin quiero ver un listado de las preguntas/respuestas obtenidas, ofreciendo filtros por estado.

4. Responder pregunta como si fuera la AI

Para no dejar preguntas sin responder correctamente/completamente, como dev-admin quiero poder editar la respuesta dada por la AI o responder a una pregunta que no pudo ser respondida automáticamente (estado “AI MISS”).

5. Extraer todos los chunks relacionados

Para hacer QA de datos, como dev-admin quiero escribir un texto y que el CLI me retorne los chunks que están relacionados en la BD, en orden descendente de distancia hasta cierto threshold.

6. Wishlist

6.1. Generar invite para una cuenta de email

Para invitar a nuevos bonsaista-juniors a un tenant, como dev-admin quiero recibir una lista de emails y devolver un json con email y link de invitación ver qué formato se necesita para que esto sea un input para algo como mailchimp o similar.



6.2. Generar QR para especie

Para poder visualizar info sobre una especie en particular y hacer preguntas únicamente sobre esa especie, como dev-admin quiero generar un QR que contenga una URL especial (url base más un token que contenga la info del vivero + la especie + algún elemento único que evite su reuso sin pagar por nuevos QRs).

6.3. Notificación de respuesta pendiente

Para poder cerrar el ciclo pregunta-respuesta, como sistema quiero notificar a bonsaista-junior que hay una respuesta sin aceptar (o rechazar). Esta notificación debería de dispararse luego de un tiempo prudencial (a definir) dado que el bonsaista-junior no sabe de inmediato si una respuesta dada fue correcta o no.

6.4. Tablero de QA

Para poder evaluar la exactitud de las respuestas, como dev-admin quiero tener un dashboard tipo "tabla" que muestre todo el contexto del usuario en una columna, la pregunta en otra y la respuesta en otra (tipo un log). Incluso esa tabla podría tener una columna más de  , o un dropdown para darle un puntaje del 1-5.

6.5. Tablero de costos

Para poder entender la estructura de costos de la solución, como dev-admin quiero poder hacer un seguimiento de los costos desglosados por servicios.

13.3.3.2. Funcionalidades Web

1. Hacer pregunta

Para aumentar el conocimiento (general o sobre una especie particular), como bonsaista-junior quiero escribir una pregunta en lenguaje natural. La pregunta será respondida utilizando el perfil del usuario como parte del contexto. No tiene porque ser respondida en el momento. Va a actuar como un foro... la pregunta se va a responder en algún momento (ver [ejemplos a soportar](#)). Si la pregunta es respondida, el estado de la pregunta pasará a "AI ANSWERED". En caso de no poder ser respondida por la AI, la pregunta será marcada como "AI MISS".

2. Disparar notificación al responder pregunta

Para saber cuándo una pregunta fue respondida por la IA, como bonsaista-junior quiero recibir un email con un link. Al momento de enviar el email se descontará 1 del crédito del bonsaista-junior.

3. Visualizar respuesta

Para ver la respuesta generada, como bonsaista-junior quiero ver la respuesta a una pregunta siguiendo el link recibido por mail. La respuesta puede tener texto o urls. Las urls deberán:

- Renderizarse como imágenes sin son imágenes
- Renderizarse como embed de youtube si es un link de youtube
- Renderizarse como un link clickable en otro caso

4. Signup con Google

Para tener acceso a ciertas funcionalidades, como bonsaista-junior quiero crear un usuario usando una cuenta de Google

5. Signup con Email&PWD

Para tener acceso a ciertas funcionalidades, como bonsaista-junior quiero crear un usuario usando una cuenta de email y un password

6. Ver/editar perfil del usuario

Para tener control de mis créditos, ver mis preguntas y ver mi info de perfil (país por ahora), como bonsaista-junior quiero ver/editar mi perfil.

7. Mostrar info de especie (según QR)

Para informar cada uno de los elementos de la taxonomía Bonsai information Taxonomy sobre un especie específica, como bonsaista-junior quiero leer un QR y que te lleve a la misma UI del "chat" pero con un resumen de generalidades de la especie (con todos los nombres comunes por país que el sistema conoce o ajustado al país si el usuario no es anónimo). El usuario anónimo podrá hacer hasta 3 preguntas (luego tendrá que pagar por otras 3).

8. Integrar con Trefle API (o similar)

Para proveer más información sobre una especie (luego de leer el QR), como bonsaista-junior quiero ver información técnica de la especie.

9. Comprar créditos

Para poder hacer nuevas preguntas, como bonsaista-junior quiero comprar créditos usando una pasarela de pagos.

10. Wishlist

10.1. Ver listado de preguntas sin contestar/rechazadas

Para que todas las preguntas realizadas por un bonsaista-junior puedan ser respondidas, como usuario anónimo quiero ver una lista de aquellas preguntas que no hayan podido ser respondidas por la AI o que hayan sido rechazadas (estado “AI MISS” o “PUBLIC AVAILABLE”)

10.2. Responder pregunta

Para contribuir con la comunidad (y potencialmente ganar créditos), como bonsaista-junior quiero responder una pregunta, indicando si quiero contribuir con esa respuesta a entrenar la AI. En caso de que la pregunta haya sido rechazada, debería poder ver dicha respuesta en el “thread”. La pregunta pasará al estado “PUBLIC ANSWERED”.

10.3. Aceptar respuesta

Para "cerrar" ciclo pregunta-respuesta y asignar, 1 crédito al autor de la respuesta (o 2 si marco que quería contribuir con la AI) y 1 al autor de la pregunta (cómo reward por tomarse el trabajo de marcar resp como la mejor), como bonsaista-junior quiero marcar una respuesta como "aceptada". La pregunta pasará al estado “PUBLIC HIT” si estaba en “PUBLIC ANSWERED” o a “AI HIT” si estaba en “AI ANSWERED”.

10.4. Rechazar respuesta

Para "cerrar" el ciclo pregunta-respuesta e indicar la no conformidad con la respuesta, como bonsaista-junior quiero marcar una respuesta como “rechazada”. Si la pregunta se encontraba en estado “AI ANSWERED” pasa a “AI MISS”, si estaba en “PUBLIC ANSWERED” pasa a “PUBLIC AVAILABLE”.

10.5. Sugerir etiquetas a preguntas

Para definir áreas de interés de las preguntas, como *bonsaista-junior* quiero recibir sugerencias de etiquetas para utilizar luego de escribir una pregunta.

13.3.3.3. Requisitos no funcionales

1. Datos de contexto

En algunos casos mencionamos que el país de un usuario es parte de su perfil y que se usará como dato de contexto. Otros datos de contexto deberán inferirse del país (ej.: hemisferio, “tipo” de clima -tropical, seco, templado, etc. a definir-).

Internacionalización

El sistema deberá soportar i18n (usando cualquiera de los approaches estándares). El único idioma que deberá soportar por ahora es Español siendo este el idioma por defecto. Respecto a las rutas de las páginas y para evitar problemas de desarrollo, serán definidas en Inglés. Las mismas se intentará que sean cortas. Por ejemplo /q/como-se-fertiliza-un-coronilla) donde el /q/ indica que es una question.

13.3.3.4. Restricciones

1. Proyecto web

Dada la diversidad de usuarios, dispositivos y formas de acercamiento con la tecnología (ej.: celulares, tablets, laptops y PC de escritorio), la solución deberá implementarse con tecnología web. Se deberán tener en cuenta los dispositivos mencionados y las resoluciones más comunes del mercado en cada una.

2. Tecnología Javascript (en la medida de lo posible)

El cliente tiene un amplio expertise en el stack Javascript. Dicho esto, no tiene problemas en adaptarse/utilizar alguna tecnología distinta en alguna parte de la solución, pero siempre que se muestre acompañe de evidencias que justifiquen su uso.

13.3.4. Ejemplos de preguntas a soportar

Estos son ejemplos de preguntas que deberá poder contestar la AI. Las mismas representan los distintos casos que encontré hasta ahora. De cada uno hago una generalización de ellos, pero lo acompaño con al menos 1 ejemplo.

Para evitar cualquier tipo de ambigüedad, es importante mencionar que toda la información utilizada para entrenar la solución se basará en nombres científicos.

Caso 1 - Coronilla

Generalización: mismo nombre científico --> distinto nombre común por país

Scutia buxifolia --> Coronilla en UY

Scutia buxifolia --> Coronillo en AR

Q: ¿Cómo se fertiliza un coronilla?

A: Para fertilizar un Scutia buxifolia suele utilizarse un fertilizante rico en nitrógeno para lograr un crecimiento explosivo.

Q: ¿Cómo se fertiliza un coronillo?

A: Para fertilizar un Scutia buxifolia suele utilizarse un fertilizante rico en nitrógeno para lograr un crecimiento explosivo.

Caso 2 - Arrayán

Generalización: mismo nombre común por país --> distinto nombre científico

Arrayán en UY --> Blepharocalyx salicifolius

Arrayán en AR --> Luma apiculata

Q: Qué sustrato utilizar para un arrayán?

A (UY): Para un Blepharocalyx salicifolius, conviene usar un sustrato algo ácido, aireado y no conviene que se encharque.

A (AR): El sustrato de un Luma apiculata conviene que siempre esté húmedo (recordar que este árbol crece al borde de los lagos)

Caso 3 – Ficus

Generalización: Cuando se pregunta por una especie, si no hay información más precisa, se puede usar la respuesta del género (primera parte del nombre científico) dado que especies de un mismo género “normalmente” se comportan de la misma forma.

Ficus pumila

Ficus microcarpa

Ficus benjamina

Q: ¿Cuándo se trasplanta un Ficus benjamina?

A: La mayoría de los Ficus sp. se trasplantan en primavera avanzada por ser tropicales

Caso 4 - Tipo de clima

Generalización: cuando se pregunta por una especie y no tengo un dato "concreto" pero sé en qué clima se "da mejor" puedo dar consejos genéricos tomando ese dato como base.

Q: ¿Cuándo se trasplanta un palo borracho?

A: El Ceiba speciosa es un árbol tropical, por lo que es recomendable trabajarlo cuando el clima ya es cálido en su zona, preferiblemente a finales de la primavera o comienzos del verano.

Caso 5 - Generalidades

Generalización: cuando se pregunta por una especie no tengo dato concreto, pero hay generalidades que aplican a cualquier especie.

Q: Cuáles son los cuidados que conviene tener luego de trasplantar un ombú?

A: Luego de trasplantar un *Phytolacca dioica* (al igual que la gran mayoría de los árboles) conviene colocarlos en un lugar iluminado pero no con sol directo, donde no corra mucho aire para evitar la pérdida de humedad.

13.4. Validación del prototipo con usuarios finales

Resultados de las entrevistas de usabilidad del prototipo

Las sesiones de validación con usuarios finales permitieron evaluar la usabilidad del prototipo, observar su interacción con las principales funcionalidades del sistema y recabar impresiones cualitativas sobre la propuesta. Participaron usuarios con distintos perfiles y niveles de experiencia en el cuidado de bonsáis. Este abanico de perfiles permitió obtener una visión equilibrada que combina perspectivas técnicas, pedagógicas y de usuario final.

Experiencia general y percepción de valor

En todos los casos, los participantes destacaron la pertinencia del sistema y el potencial de la herramienta como espacio de intercambio confiable para la comunidad bonsaísta. Se reconoció especialmente el valor de contar con una plataforma que centralice la información y permita acceder a respuestas contextualizadas, una necesidad recurrente en el dominio. Los usuarios coincidieron en que la propuesta resultaría útil tanto para principiantes como para aficionados avanzados, aunque subrayaron la importancia de adaptar el nivel de profundidad de las respuestas al nivel de conocimiento del usuario.

Usabilidad e interacción con el sistema

El flujo de creación de preguntas fue comprendido con facilidad por todos los entrevistados. La interfaz fue percibida como intuitiva y clara, destacándose el diseño de la caja de texto y el sistema de etiquetas. No obstante, surgieron sugerencias para mejorar

la navegación y la retroalimentación inmediata: varios usuarios esperaban que, una vez publicada la pregunta, el sistema los dirigiera directamente a la visualización de la misma en lugar de mostrar solo una notificación (“toast”). También se propuso autogenerar etiquetas basadas en el contenido, dado que no todos los usuarios dominan su uso.

En cuanto a la navegación, algunos entrevistados expresaron que la ubicación de ciertas opciones —como “Mis preguntas” o “Cambiar contraseña”— no era del todo intuitiva, sugiriendo incorporar accesos alternativos o etiquetas más explícitas. En general, las tareas de perfil y configuración se realizaron correctamente, aunque se destacó la necesidad de simplificar los datos solicitados en los formularios y de permitir un mayor control sobre la información personal compartida.

Recomendaciones y aportes funcionales

Entre las recomendaciones más relevantes surgieron ideas orientadas a incrementar la personalización de la experiencia. Los usuarios propusieron:

- Incorporar información contextual como el clima o la localización geográfica para ajustar las recomendaciones (“no realizar podas o trasplantes durante olas de calor”).
- Permitir especificar el nivel de detalle deseado en las respuestas (breve o desarrollada).
- Mostrar la fuente de las respuestas, especialmente en aquellas generadas por la IA, para aumentar la confianza.
- Añadir mecanismos de reputación para los usuarios que respondan, como sistemas de puntuación o insignias.
- Habilitar notificaciones sobre respuestas nuevas o actividad en temas seguidos, así como una sección de “favoritos”.
- Explorar la integración con herramientas de reconocimiento de especies o generación de imágenes de diseño de bonsáis.

Aspectos de accesibilidad y diseño visual

Se identificaron algunos ajustes necesarios para mejorar la accesibilidad, particularmente en la legibilidad de textos y etiquetas. En el caso de usuarios con baja visión, el contraste entre fondo y tipografía dificultaba la lectura, por lo que se recomendó revisar los colores

y tamaños de fuente. También se sugirió incluir íconos más representativos para las secciones principales y mejorar la jerarquía visual entre los distintos tipos de contenido (preguntas, respuestas y comentarios).

Conclusiones generales

Las entrevistas confirmaron que el prototipo cumple con su propósito central de ofrecer una experiencia de interacción fluida y significativa para los usuarios interesados en bonsáis. Las observaciones obtenidas no solo validan la usabilidad del sistema, sino que aportan insumos valiosos para su evolución, principalmente en torno a la personalización de las respuestas, la mejora de la navegación y la transparencia de la información generada por la IA. En conjunto, estas instancias evidenciaron una recepción positiva de la propuesta y reforzaron su alineación con las necesidades reales del público objetivo.

13.5. Listado de Requerimientos Funcionales

RFT1. Creación de comunidad

El sistema deberá proporcionar un endpoint para la creación de nuevos tenants. Cada tenant deberá disponer de un espacio de datos independiente y de credenciales de acceso propias, asegurando así la adecuada segregación y aislamiento respecto de los demás tenants.

RFT2. Modo de operación de la comunidad

El sistema deberá permitir que cada comunidad opere en dos modos configurables: libre o moderado.

En el modo moderado, cada respuesta generada por la IA o por los usuarios deberá recibir inicialmente el estado PENDIENTE, hasta ser ACEPTADA o RECHAZADA por un administrador. Este mecanismo asegura que las respuestas cuenten con validación previa, reforzando la calidad de la información.

En el modo libre, las respuestas se publicarán directamente sin requerir la intervención del administrador, priorizando la inmediatez y la fluidez de la interacción.

En ambos modos, las respuestas deberán ser siempre visibles para los usuarios, independientemente de su estado. El modo de operación deberá definirse al momento de crear la comunidad y podrá modificarse posteriormente por el administrador.

RFT3. Selección de mejor respuesta

El sistema deberá permitir que cada comunidad configure la habilitación de la funcionalidad de selección de la mejor respuesta. Cuando esta funcionalidad esté activa, cada usuario podrá seleccionar una única mejor respuesta entre todas las disponibles para una pregunta específica.

En comunidades que operen en modo moderado, la selección de la mejor respuesta deberá restringirse exclusivamente a aquellas respuestas que hayan sido aceptadas por un administrador, asegurando que solo el contenido validado pueda destacarse como la opción más relevante.

En comunidades que operen en modo libre, cualquier respuesta podrá ser seleccionada como la mejor, dado que no existe un proceso previo de moderación. Esta funcionalidad deberá aplicarse tanto a respuestas generadas por la IA como a respuestas proporcionadas por los usuarios, y deberá ser configurable por el administrador de cada comunidad.

RFA1. Carga de datos

El sistema deberá brindar al administrador la posibilidad de realizar la carga de datos a la comunidad, contemplando distintos formatos de entrada según la naturaleza de la información. Por un lado, se permitirá la incorporación de archivos en formato Markdown, utilizados para alimentar la base de conocimiento específica de cada comunidad. Por otro lado, se habilitará la carga de datos estructurados en formato Excel, destinados principalmente al nomenclátor provisto por el cliente.

Una vez cargados, los datos deberán integrarse a la base de conocimiento de la comunidad correspondiente, asegurando la correcta segregación de la información entre distintos dominios..

Este requerimiento funcional se encuentra alineado con el documento de requerimientos iniciales del cliente, específicamente en el apartado 4.1.1. Ingest de datos, el cual

establece la necesidad de que los administradores dispongan de un mecanismo seguro y flexible para abastecer de datos al sistema.

RFA2. Invitar miembro a la comunidad

El sistema deberá brindar al administrador la capacidad de invitar nuevos miembros a la comunidad, generando enlaces de invitación únicos que se envíen automáticamente a las direcciones de correo electrónico proporcionadas. Este mecanismo debe permitir incorporar usuarios de manera segura y controlada, asegurando que cada invitación sea válida únicamente para el destinatario correspondiente y evitando accesos no autorizados.

Este requerimiento funcional está alineado con lo definido en el documento de requerimientos iniciales del cliente, en el apartado 4.1.6.1. Generar invite para una cuenta de email, donde se establece la necesidad de contar con un proceso formalizado para la incorporación de nuevos miembros a la comunidad.

RFA3. Administrar miembros de la comunidad

El sistema deberá proporcionar al administrador un conjunto de herramientas que permitan gestionar de manera eficiente a los miembros de la comunidad. Entre estas herramientas se incluyen la visualización del listado de usuarios y la desactivación de cuentas cuando sea necesario, garantizando un control adecuado sobre el acceso y las acciones dentro de la comunidad.

Si bien este requerimiento no se encontraba contemplado en el documento de requerimientos iniciales del cliente, se incorporó posteriormente tras diversas conversaciones con el cliente. La decisión de agregarlo respondió a la necesidad de moderar y mantener organizada la comunidad en el contexto de una plataforma interactiva orientada a la participación de los usuarios.

RFA4. Listar preguntas y respuestas

El sistema deberá permitir al administrador acceder a un listado de preguntas y respuestas realizadas dentro de la comunidad, ofreciendo filtros por estado, etiqueta u otros criterios relevantes para facilitar la revisión y el análisis de la información. Esta funcionalidad permitirá identificar preguntas que necesiten edición o corrección, asegurando un adecuado control de calidad sobre las respuestas generadas.

Este requerimiento funcional se basa en el requerimiento 4.1.3 Listar preguntas del documento de requerimientos iniciales del cliente, que especifica la necesidad de visualizar y filtrar las preguntas y respuestas para realizar QA y asegurar la consistencia de la información.

RFA5. Dejar comentario en una respuesta de IA

El sistema deberá permitir al administrador registrar comentarios sobre las respuestas generadas por la IA, con el objetivo de aportar retroalimentación o contexto adicional. Esta funcionalidad asegura que las respuestas puedan ser evaluadas y mejoradas de manera ética, sin modificar directamente la salida de la IA.

Este requerimiento funcional se origina a partir del requerimiento 4.1.4. Responder pregunta como si fuera la AI del documento de requerimientos iniciales del cliente, que originalmente permitía editar respuestas de la IA. Tras conversaciones con el cliente, se acordó que modificar las respuestas no era éticamente adecuado, por lo que se definió esta solución alternativa de dejar comentarios como mecanismo de retroalimentación.

RFA6. Rechazar respuesta brindada por la IA en modo moderado

El sistema deberá permitir al administrador rechazar respuestas generadas por la IA cuando estas no cumplan con los criterios de calidad o precisión establecidos para la comunidad. Al rechazar una respuesta, la pregunta asociada será marcada como RECHAZADA, asegurando que los usuarios no interpreten información incorrecta como válida. Esta funcionalidad es aplicable únicamente cuando la comunidad opera en modo moderado, en el cual cada respuesta recibirá inicialmente el estado PENDIENTE hasta ser ACEPTADA o RECHAZADA por un administrador, garantizando la depuración y calidad de la información.

Este requerimiento funcional se adapta del requerimiento 4.1.4. Responder pregunta como si fuera la AI del documento de requerimientos iniciales del cliente, que originalmente permitía al administrador editar respuestas de la IA. Tras analizar aspectos éticos y de consistencia, se determinó que rechazar respuestas inapropiadas en lugar de modificarlas ofrece un mecanismo más transparente de control y retroalimentación, alineado con las prácticas de moderación de la comunidad.

RFA7. Búsqueda semántica con extracción de datos

El sistema deberá permitir al administrador realizar búsquedas semánticas dentro de la comunidad, extrayendo los fragmentos de información (chunks) que estén relacionados con un texto de consulta. Los resultados deberán presentarse en orden descendente de relevancia, considerando un umbral definido que limite la inclusión de información menos relacionada.

Este requerimiento funcional se adapta del requerimiento 4.1.5. Extraer todos los chunks relacionados del documento de requerimientos iniciales del cliente, que originalmente permitía al administrador extraer todos los chunks relacionados para tareas de QA. La adaptación extiende la funcionalidad hacia un mecanismo de búsqueda semántica, orientado a facilitar el análisis y verificación de datos dentro de la comunidad.

RFA8. Administración de etiquetas

El sistema deberá permitir al administrador gestionar las etiquetas utilizadas dentro de la comunidad, incluyendo la creación, edición, eliminación y categorización de etiquetas. Esto permitirá organizar de manera eficiente las áreas de interés y asegurar consistencia en la clasificación de preguntas y respuestas.

Este requerimiento funcional se adapta de la contraparte 4.2.10.5 Sugerir etiquetas a preguntas del documento de requerimientos iniciales del cliente, que originalmente permitía a los usuarios recibir sugerencias de etiquetas. La adaptación traslada la capacidad de administración al administrador, garantizando que la taxonomía de etiquetas se mantenga coherente y controlada dentro de la comunidad.

RFA9. Dashboard de estadísticas de la comunidad

El sistema deberá contar con un dashboard destinado al administrador, que permita visualizar de manera clara la actividad y el desempeño de la comunidad. A través de este tablero, será posible obtener información sobre la participación de los usuarios, la evolución de las preguntas y el funcionamiento de la inteligencia artificial en la gestión de respuestas.

Este requerimiento surge de la necesidad de proporcionar al administrador un medio rápido y eficiente para visualizar estadísticas generales, permitiéndole evaluar de manera inmediata la salud y el desempeño de la comunidad.

RFA10. Configuración de comunidad

El sistema deberá permitir al administrador definir aspectos clave del funcionamiento de cada comunidad, incluyendo el modo de operación y la posibilidad de seleccionar la mejor respuesta. Las comunidades podrán operar en modo moderado, donde las respuestas son revisadas y validadas por un administrador antes de mostrarse, o en modo libre, donde las respuestas se publican directamente, favoreciendo una interacción más ágil.

Además, se podrá habilitar la funcionalidad de selección de la mejor respuesta, que permitirá a los usuarios resaltar la opción más relevante para cada pregunta. En comunidades moderadas, únicamente podrán marcarse como mejores las respuestas previamente validadas por un administrador, mientras que en comunidades libres cualquier respuesta podrá ser seleccionada. Esta configuración podrá modificarse en cualquier momento según las necesidades de la comunidad.

RFU1. Hacer pregunta

El sistema deberá permitir al usuario formular preguntas en lenguaje natural sobre cualquier tema relacionado con el nicho específico de la comunidad en la que se encuentra. Las preguntas se enviarán considerando el perfil del usuario como parte del contexto, y recibirán respuesta por parte del sistema de IA . En comunidades en modo moderado, la respuesta se mostrará como pendiente de aprobación hasta su aceptación por un administrador.

Este requerimiento funcional se adapta de 4.2.1. Hacer pregunta del documento inicial, que originalmente permitía a los usuarios realizar preguntas. La adaptación considera ahora los modos de operación de la comunidad, garantizando consistencia entre visibilidad de respuestas y control de calidad.

RFU2. Visualizar respuestas

El sistema deberá permitir que los usuarios consulten las respuestas asociadas a cada pregunta, mostrando el contenido de manera clara y estructurada. Las respuestas podrán incluir texto o enlaces.

Este requerimiento funcional se adapta de 4.2.3. Visualizar respuesta del documento de requerimientos iniciales del cliente, que originalmente permitía a los usuarios visualizar respuestas.

RFU3. Feed de preguntas

El sistema deberá mostrar un feed con las preguntas recientes y populares de la comunidad.

Este requerimiento responde a la necesidad de orientar la plataforma hacia un enfoque más social, permitiendo que los usuarios visualicen no solo sus propias preguntas, sino también las de otros miembros, con el fin de fomentar la interacción y el intercambio dentro de la comunidad.

RFU4. Filtrar preguntas

El sistema deberá permitir al usuario filtrar preguntas por criterios como estado, etiquetas o contenido, facilitando la navegación y priorización de información relevante dentro de la plataforma.

Este requerimiento funcional se adapta de las funcionalidades 4.1.3. Listar preguntas y 4.2.10.1. Ver listado de preguntas sin contestar/rechazadas del documento inicial, que originalmente estaban orientadas a la revisión y listado de preguntas por parte del administrador o de usuarios específicos. La adaptación amplía el acceso de filtrado a todos los usuarios, mejorando la experiencia de búsqueda y consulta de información.

RFU5. Agregar etiquetas a preguntas

El sistema deberá permitir al usuario agregar etiquetas sugeridas al momento de crear una pregunta, asegurando que la información se clasifique de manera coherente y sea fácilmente recuperable.

Este requerimiento funcional se adapta al 4.2.10.5 del documento inicial, que originalmente permitía a los usuarios recibir sugerencias de etiquetas. La adaptación mantiene la funcionalidad principal, enfocándola a la experiencia general del usuario para organizar preguntas y optimizar búsquedas pero bajando la complejidad de manera de poder centrar el esfuerzo en otros requerimientos.

RFU6. Agregar respuesta

El sistema deberá permitir que un usuario agregue respuestas a preguntas, complementando o corrigiendo la información proporcionada por la IA. Esto asegura que todas las preguntas puedan ser respondidas de manera completa, incluso cuando la IA no genere una respuesta adecuada, y permite a los usuarios involucrarse activamente en la comunidad.

Este requerimiento funcional se adapta del requerimiento 4.2.10.2. Responder preguntas del documento inicial, garantizando la continuidad del ciclo pregunta-respuesta y la participación de los miembros.

RFU7. Seleccionar mejor respuesta

El sistema deberá permitir que el usuario seleccione una única mejor respuesta para su pregunta, siempre considerando las restricciones del modo de operación. En modo moderado, sólo se podrá seleccionar entre respuestas previamente ACEPTADAS; en modo libre, cualquier respuesta disponible podrá ser marcada como la mejor.

Este requerimiento funcional se adapta de 4.2.10.3. Aceptar respuesta y deriva del RFT3. Selección de mejor respuesta, integrando la dependencia del modo de operación de la comunidad para garantizar que solo contenido validado pueda ser reconocido como la mejor respuesta.

RFU8. Votar respuestas

El sistema deberá permitir que los usuarios califiquen la utilidad o calidad de una respuesta mediante un sistema de votos.

Este requerimiento funcional se adapta de los requerimientos 4.1.6.4. Tablero de QA y 4.2.10.3. Aceptar respuesta del documento entregado inicialmente por el cliente, La adaptación extiende la capacidad de votar a todos los usuarios, no solo para que la comunidad pueda auto-regularse, sino también para recopilar retroalimentación sobre las respuestas otorgadas por la IA.

RFU9. Aceptar invitación desde el link del email

El sistema deberá permitir que los usuarios nuevos acepten invitaciones mediante enlaces de correo electrónico, garantizando autenticidad y completando el registro de forma segura.

Este requerimiento funcional se adapta de 4.1.6.1. Generar invite para una cuenta de email, permitiendo la aceptación de invitaciones a todos los usuarios, manteniendo la seguridad y trazabilidad.

RFU10. Iniciar sesión con mail y contraseña

El sistema deberá permitir que los usuarios registrados accedan con correo y contraseña, asegurando protección de la información personal.

Este requerimiento funcional se adapta del requerimiento 4.2.5. Signup con Email&PWD del documento inicial entregado por el cliente.

RFU11. Editar perfil

El sistema deberá permitir que los usuarios vean y modifiquen su perfil, incluyendo país e información personal. Esto asegurará que las respuestas y recomendaciones se adapten al contexto del usuario.

Este requerimiento funcional se adapta del requerimiento 4.2.6. Ver/editar perfil del usuario del documento inicial entregado por el cliente.

RFU12. Seleccionar comunidad

El sistema deberá permitir que los usuarios visualicen un listado de las comunidades a las que pertenecen y seleccionen autenticarse en una de ellas.

Este requerimiento funcional surge de la necesidad de gestionar usuarios que forman parte de múltiples comunidades, garantizando que puedan acceder de manera diferenciada a cada una según su membresía. Además, facilita la navegación, mantiene la coherencia de la información presentada y asegura que la experiencia de usuario se adapte al contexto específico de la comunidad seleccionada.

RFU13. Guardar preguntas

El sistema deberá permitir que los usuarios guarden preguntas de interés para revisarlas más tarde.

Este requerimiento funcional surge de la necesidad de que los usuarios puedan organizar y gestionar la información relevante de manera personal, facilitando el acceso posterior a preguntas importantes o de mayor interés personal, y mejorando la experiencia de navegación dentro de la comunidad.

RFU14. Buscar preguntas

El sistema deberá permitir que los usuarios busquen preguntas mediante palabras clave, etiquetas o estados.

Este requerimiento funcional se inspira en el requisito 4.2.10.1. Ver listado de preguntas sin contestar/rechazadas del documento inicial, y responde a la necesidad de facilitar la localización rápida de información específica, optimizando la experiencia de navegación y consulta dentro de la comunidad.

RFU15. Listar preguntas del usuario

El sistema deberá permitir que los usuarios visualicen un listado completo de sus preguntas, incluyendo estados y respuestas asociadas.

Este requerimiento funcional surge de la necesidad de que los usuarios puedan hacer un seguimiento integral de sus interacciones dentro de la comunidad, facilitando la revisión

de sus preguntas, el control sobre el estado de cada una y el acceso a las respuestas recibidas, mejorando así la gestión personal de la información.

RFU16. Completar encuesta sobre la respuesta de la IA.

El sistema deberá permitir que los usuarios completen una encuesta después de recibir la respuesta de la IA, con el fin de obtener retroalimentación inmediata sobre la calidad, utilidad y precisión de las respuestas proporcionadas durante la etapa MVP del producto.

Este requerimiento surge como consecuencia del requerimiento no funcional RNF3. Precisión de respuestas, que establece la necesidad de garantizar un nivel suficiente de exactitud en las respuestas del modelo LLM para cumplir con las expectativas de los usuarios. La encuesta permitirá recolectar datos directos sobre la percepción de precisión, relevancia y confiabilidad de las respuestas, complementando los mecanismos de evaluación automatizada y votación de usuarios, y asegurando así que los criterios de precisión y satisfacción definidos en RNF3 se puedan validar y ajustar en el contexto real de uso.

RFU17. Cambiar contraseña

El sistema deberá permitir que los usuarios modifiquen su contraseña de manera segura, protegiendo la cuenta y la información personal.

13.6. Investigaciones del sistema de IA

13.6.1. Proveedores de modelos de IA

Propósito y alcance

Este anexo resume la evaluación de opciones de LLMs para el asistente conversacional, tomando como caso de prueba la consigna:

“¿Cuáles son los cuidados que conviene tener luego de trasplantar un ombú?”

El análisis prioriza cinco ejes para decidir el modelo final: Task, Performance, Cost, Ease of Use y Customization. Se incluyen pros/contras, enlaces a documentación oficial y un cuadro ejecutivo para una lectura rápida.

Resumen ejecutivo (criterios de decisión)

Criterio	Hallazgos relevantes
Task (Adecuación a la tarea)	Ambos (OpenAI y Google Gemini) resuelven preguntas clínicas del dominio con buen “reasoning”. Gemini aporta multimodalidad nativa útil si se proyecta trabajar con imágenes (p. ej., fotos del bonsái).
Performance (Precisión/Eficiencia)	Gemini (1.5/2.x) ofrece contextos muy amplios (hasta orden de millones de tokens en versiones avanzadas) y buena latencia para consultas largas. GPT-4/4-Turbo mantiene consistencia y coherencia muy sólida en texto extenso y citación de evidencias.
Cost (Modelo de precios)	OpenAI: pricing por tokens documentado y estable; GPT-4 Turbo suele ser más económico/rápido que GPT-4. Gemini: esquema por uso; ventaja potencial en sesiones largas por su gran ventana de contexto, pero con variabilidad según versión/región.
Ease of Use (Integración)	OpenAI: SDKs maduros (Python/Node), documentación extensa y comunidad activa. Gemini: buena DX, con fuerte integración a Google Cloud y Google AI Studio para experimentar y compartir prototipos.
Customization (Ajuste/afinación)	OpenAI: fine-tuning disponible de forma acotada según modelo. Gemini: soporte de tuning para ciertas versiones; Gemini Flash 1.5–8B descartado en este proyecto por restricciones de fine-tuning .

Tabla 12: Criterios de decisión de proveedores de modelo de IA

Tabla 1 — ChatGPT (OpenAI) vs. Google Gemini (Google AI)

Atributo	ChatGPT (OpenAI)	Google Gemini
Precio (alta-nivel)	Free (modelos base) y Plus con acceso a GPT-4/4-Turbo; API con costos por 1k tokens.	Basic/Advanced; precios y cupos vinculados a cuenta/organización; API con uso medido.
Performance	Muy buena coherencia y estabilidad; excelente para redacción larga y estructurada.	Respuestas ágiles y buena calidad; ventanas de contexto grandes para casos largos.
Multimodalidad	GPT-4 con imagen (según plan y endpoint); 3.5 texto-only.	Multimodal nativo (texto, imagen, audio según versión).
Integración	SDKs oficiales, ejemplos, comunidad grande.	Profunda integración con Google Cloud / Workspace y AI Studio.
Context Window	3.5: ~4k; 4: ~8k; 4-Turbo: ~32k.	1.5 Pro/2.x: hasta 1M (según versión).
RAG	Soporta RAG con embeddings + vector DB (p. ej., Pinecone) o tool-use.	Arquitectura con retrieval bien integrado; útil para reducir alucinación.

Tabla 13: ChatGPT (OpenAI) vs. Google Gemini (Google AI)

Tabla 2 — Modelos y versiones

Atributo	OpenAI	Google Gemini
Variantes	GPT-3.5, GPT-4, GPT-4 Turbo (prioriza coste/latencia).	Gemini 1.x, 1.5, 2.x (multimodal avanzado).
Disponibilidad	Free/Plus y API.	Basic/Advanced y API.
Notas de proyecto	4-Turbo: buen equilibrio calidad/costo/latencia.	Flash 1.5-8B: descartado por no permitir fine-tuning acorde a nuestras necesidades.

Tabla 14: Modelos y versiones

Tabla 3 — Esquema de precios (alto nivel)

La comparación es cualitativa: los costos concretos dependen de versión, región, plan y volumen de tokens/requests del proyecto.

Atributo	ChatGPT (OpenAI)	Google Gemini
Free tier	Acceso a GPT-3.5 (limitado).	Gemini Basic (limitado).
Plan usuario	ChatGPT Plus con GPT-4/4-Turbo.	Gemini Advanced (precio varía).
API (tokens/uso)	Transparente por tokens (input/output); 4-Turbo más económico/rápido que 4.	Uso medido; favorece contextos grandes; variabilidad por versión.
Cost drivers	Modelo elegido + volumen de tokens + herramientas (embeddings, etc.).	Versión (1.5/2.x), ventana de contexto y multimodalidad.

Tabla 15: Esquema de precios (alto nivel)

Tabla 4 — Perspectiva desarrollador(a)

Atributo	ChatGPT (OpenAI)	Google Gemini
API/SDKs	REST, SDKs oficiales, estándares claros, errores/documentación madura.	REST, SDKs y Google AI Studio para iterar; integración GCP sólida.
Token pricing	Claridad en input/output por 1k tokens.	Por uso, con impacto de contextos largos y modalidades.
Context handling	Hasta 32k en 4-Turbo.	Hasta 1M tokens (según versión).
Multimodal	Imagen en GPT-4; 3.5 texto.	Texto/imagen/audio nativo (según versión).
RAG	Requiere vector DB/embeddings externos (Pinecone, etc.).	Retrieval integrado; reduce dependencia externa.

DX/Comunidad	Muy amplia.	Creciente, con ventajas si ya usas Google Cloud.
--------------	-------------	--

Tabla 16: Perspectiva desarrollador(a)

Tabla 5 — Mirada ampliada (opciones alternativas)

Atributo	Claude (Anthropic)	LLaMA (Meta)	Cohere Command R
Disponibilidad	API con foco safety/long context.	Open-weights; requiere self-hosting o vendor.	API enfocada a RAG y enterprise.
Ventajas	Muy buen razonamiento y contexto largo.	Costo controlado (infra propia), custom total.	Buenas embeddings/RAG, coste competitivo.
Desafíos	Costos por uso, disponibilidad por región.	Operación/MLOps y seguridad a cargo.	Menor ecosistema que OpenAI/Google.

Tabla 17: Opciones alternativas de proveedores

Tabla 6 — Gestión de datos personales y cumplimiento

Atributo	OpenAI	Google Gemini	Claude	LLaMA (self-host)	Cohere
Retención	Abuso 30 días; sin training por defecto en planes pagos.	Integrado al marco de privacidad Google.	Foco privacy, retención mínima.	A elección del implementador (autoservicio).	Política clara, orientada a enterprise.
Cumplimiento	GDPR/CCPA; HIPAA no por defecto.	GDPR/CCPA vía Google Cloud.	Fuerte enfoque compliance.	Depende del entorno/hiperparámetros.	Orientado a GDPR/enterprise.

Control del dato	Borrado a demanda; aislamiento enterprise.	Herramientas de gestión de datos en ecosistema Google.	Sin reutilización sin consentimiento.	Control total (con costo operativo).	Sin reuso por defecto; políticas claras.
------------------	--	--	---------------------------------------	--------------------------------------	--

Tabla 18: Gestión de datos personales y cumplimiento.

Pros y contras destacados (según pruebas y docs)

Gemini (Google AI)

Pros: Multimodalidad nativa; ecosistema Google; contexto extenso; AI Studio para prototipado/compartir (ej.: link de tu prueba).

Contras: Variabilidad de planes/precios por versión/región; ecosistema más nuevo que OpenAI en comunidad; **Flash 1.5–8B descartado** por restricciones de **fine-tuning** para este proyecto.

OpenAI (GPT-4 / 4-Turbo)

Pros: Calidad de texto y coherencia; SDKs/documentación maduras; comunidad grande; pricing predecible por tokens.

Contras: Costos pueden crecer en cargas altas; fine-tuning más acotado en modelos tope.

Evidencia de pruebas y artefactos

Google AI Studio

Nota: En nuestras corridas de la “pregunta comparadora” el foco estuvo en exactitud de recomendaciones, uso de contexto recuperado (RAG) y estilo de redacción. La evaluación cualitativa mostró buen desempeño en ambos; Gemini aporta ventaja potencial si en el futuro se incorporan imágenes (hojas, sustratos, raíces) para diagnóstico visual, mientras que GPT-4-Turbo ofrece excelente costo-rendimiento para texto + RAG.

Recomendación práctica (cómo elegir “el final”)

Si el roadmap incluye imágenes (p. ej., analizar fotos del bonsái post-trasplante) y contextos extensos (protocolos, fichas, series de chat largas): Gemini 1.5/2.x es una apuesta sólida.

Si el foco es texto con RAG (vector DB + evidencias) buscando calidad estable y costes predecibles: GPT-4-Turbo es un default muy bueno.

Personalización: si se requiere fine-tuning estricto a corto plazo, validar disponibilidad real por versión (descartado Flash 1.5–8B en este proyecto).

Costeo: proyectar el volumen de tokens (input/output) y la longitud del contexto de las sesiones reales; elegir el modelo según el punto dulce de coste/latencia/calidad.

DX/Operación: elegir en función del stack actual (GCP vs. multi-cloud) y el flujo de prototipado (AI Studio vs. SDKs OpenAI).

Las conclusiones sintetizan documentación oficial [40] - [46] y las pruebas del repositorio citado; los precios y disponibilidades pueden variar por versión/región y deben confirmarse al momento de contratación.

13.6.2. Javascript vs Python

Python para Ciencia de Datos

Python es uno de los lenguajes dominantes dentro de la ciencia de datos debido a la facilidad de uso que presenta y la gran cantidad de bibliotecas que dispone.

A continuación, se presentan algunas bibliotecas que pueden resultar útiles en el contexto del proyecto que se busca desarrollar:

Pandas

Esta biblioteca de código abierto se utiliza principalmente para el análisis, manipulación y depuración de datos. Permite realizar operaciones de modelado y análisis de datos de forma sencilla. Según se describe en su página web, Pandas es “una herramienta rápida, potente, flexible y fácil de usar para el análisis y manipulación de datos desarrollada sobre el lenguaje de programación Python”.

Como mencionan diversas fuentes, esta biblioteca se caracteriza por la disposición de herramientas para la lectura y escritura de datos en múltiples formatos, sobre los que se incluye SQL, Excel, CSV, entre otros. También se destaca por su manejo de datos y tratamiento de valores faltantes, facilitando el procesamiento de datos que no se encuentran ordenados para convertirlos en información estructurada.

La flexibilidad que brinda esta biblioteca permite realizar operaciones como reestructuración y pivoteo de conjunto de datos, slicing basado en etiquetas, indexación avanzada y creación de subconjunto incluso en grandes volúmenes de datos. Además, ofrece la capacidad de agregar o transformar datos. Por otra parte, permite insertar y eliminar columnas en estructuras de datos para manejar el tamaño dinámicamente, así como realizar uniones y combinaciones de alto rendimiento entre conjuntos de datos.

Estos factores, hacen que esta biblioteca sea utilizada para el análisis de datos y la preparación de datos para modelos de aprendizaje automático. Etapas claves para el análisis de la información que entrenarán al modelo elegido para este proyecto.

Matplotlib

Matplotlib es una biblioteca de visualización de datos que permite crear gráficos estáticos, animados e interactivos. Existen múltiples paquetes que extienden las funcionalidades que ofrece Matplotlib. Esto permitirá presentar los datos utilizando una amplia variedad de gráficos, como diagramas de dispersión, histogramas, gráficos de barras, gráficos de errores y diagramas de caja, entre otros. Además, todas estas visualizaciones pueden implementarse con tan solo unas pocas líneas de código, lo que facilita su uso tanto para principiantes como para usuarios avanzados.

Scikit-Learn

Esta biblioteca de código abierto, implementa una gran cantidad de algoritmos de aprendizaje automático, preprocesamiento, validación cruzada y visualización utilizando una interfaz unificada. Proporciona una gran cantidad de herramientas para diversas tareas de aprendizaje automático, como clasificación, regresión, clustering, entre muchas otras.

Alguna de las características principales de esta biblioteca es que es fácil de utilizar e integrar con otras bibliotecas de Python, como las mencionadas anteriormente. Por otra

parte, brinda herramientas para la validación cruzada y la selección de modelos. Es una herramienta que puede ser útil para construir y evaluar modelos de aprendizaje automático. Además, un gran beneficio que presenta esta biblioteca es su rendimiento y escalabilidad. Al estar basado en bibliotecas optimizadas (como NumPy y SciPy) se aceleran los cálculos y la gestión de memoria. Scikit-Learn ofrece opciones de paralelización y computación distribuida para el mejor manejo de conjuntos de datos y modelos de gran tamaño y complejidad.

Bibliotecas para procesamiento del lenguaje natural

NLTK (Natural Language Toolkit)

Este conjunto de librerías permite llevar a cabo diversas tareas relacionadas con el procesamiento del lenguaje natural. Proporciona un conjunto de bibliotecas que permiten el procesamiento de texto para su clasificación, tokenización, lematización, etiquetado, análisis sintáctico y razonamiento semántico.

Gensim

Gensim es una biblioteca para modelado de temas, indexación de documentos y recuperación de similitudes con grandes corpus.

Sus principales características incluyen el modelado de temas y el análisis de similitud. El modelado de temas es una técnica que permite identificar los temas principales en un corpus de documentos. El uso de esta biblioteca para el modelado de temas, permite organizar, resumir y explorar datos de texto, así como reconocer brechas y tendencias. El análisis de similitud es un método que permite medir qué tan similares o diferentes son dos o más documentos en función de su contenido. Gensim ofrece varios métodos para el análisis de similitud y la distancia del motor de palabras.

Bibliotecas adicionales

Opik

Opik es una plataforma de código abierto, diseñada para evaluar, probar y monitorear aplicaciones que utilizan Modelos de Lenguaje de Gran Escala (LLM). Entre sus principales funcionalidades se incluye trazabilidad que permite registrar y dar seguimiento a cada interacción y proceso que realiza el LLM tanto en entornos de

desarrollo como en producción. Esta trazabilidad permite registrar todas las llamadas realizadas al LLM, así como los pasos intermedios en la generación de respuestas, logrando brindarnos una visión completa del funcionamiento en el flujo de trabajo del modelo. Esto reduce los tiempos empleados en la identificación y resolución de problemas, ya que al tener un registro detallado de lo que ocurre paso a paso permite localizar con facilidad errores y comportamientos inesperados del modelo. Por otra parte, considerando las etapas de desarrollo y definición de los parámetros a utilizar para el modelo, el hecho de poseer un detalle de los pasos internos que siguió el modelo para llegar a una respuesta, nos permitirá descartar hipótesis o mejorar el funcionamiento del modelo.

Además, Opik brinda un playground, sería una especie de laboratorio interactivo en dónde se permite probar y ajustar los prompts y modelos deseados. A nivel de experimentación, permite escribir y modificar diversos prompts, como instrucciones o preguntas, con el fin de ver cómo responden distintos modelos de lenguaje. Esto permite probar de forma rápida distintas configuraciones sin la necesidad de implementar cambios permanentes. A nivel de este proyecto esto es de gran beneficio ya que una parte fundamental del mismo es conseguir una configuración de parámetros para el modelo final a utilizar. Esta herramienta permite comparar cómo responden distintos modelos al mismo prompt, permitiendo identificar fortalezas y debilidades en cada modelo para ciertas situaciones específicas para lograr elegir el modelo que mejor se adapte a la solución.

Además, Opik se integra a pipelines de Integración Continua y Despliegue Continuo a través de PyTest. Esto permite incluir tests automatizados como parte del proceso de despliegue en diversos entornos a lo largo del proceso de desarrollo. Logrando que cada cambio en el código o el modelo pueda ser exhaustivamente testeado antes de ser implementado.

Finalmente, esta herramienta permite mantener un monitoreo del modelo en la etapa de producción. Opik está diseñado para manejar grandes volúmenes de datos, permitiendo registrar cada interacción que se generan en el entorno de producción. Esta herramienta provee paneles de control dónde se pueden revisar puntajes de retroalimentación, las trazas que se generan. Esto permite una visión detallada y en tiempo real del rendimiento

del modelo, permitiendo identificar y solucionar potenciales problemas que puedan surgir de forma rápida. Logrando garantizar de esta manera, que el modelo logre operar de forma continua y efectiva.

Hasta la fecha (21/01/2025), Opik ofrece un SDK en Python para la integración con aplicaciones que utilizan LLM. Este kit de desarrollo de software (SDK) facilita la interacción entre Opik y el modelo a utilizar.

Notas adicionales

Actualmente, Opik no cuenta con un SDK para JavaScript aunque sus desarrolladores se encuentran trabajando en la misma. Estiman que se publicará próximamente, pero aún no se tiene una fecha estimada de publicación. Además, no existe una herramienta análoga para JavaScript. Se podría sustituir por un conjunto de librerías para lograr funcionalidades similares a las ofrecidas por Opik.

JavaScript for Data Science

JavaScript es un lenguaje comúnmente utilizado para el desarrollo web, pero últimamente comenzó a ganar terreno en el campo de la ciencia de datos. A pesar de no ser tan comúnmente utilizado como Python en este ámbito, JavaScript ofrece diversas ventajas y herramientas que lo hacen una opción viable para ciertos proyectos de ciencia de datos.

A continuación, se presentan algunas bibliotecas que pueden resultar útiles en el contexto del proyecto que se busca desarrollar:

TensorFlow.js

Esta biblioteca permite entrenar y desplegar modelos de aprendizaje automático en el navegador del cliente, permitiendo eliminar la necesidad de servidores potentes y la reducción de la latencia.

Dicha biblioteca permite entrenar los modelos y utilizarlos en tiempo real, presentando una gran ventaja de esta herramienta para aplicaciones interactivas. Por otra parte, ofrece una variedad de modelos pre entrenados que se pueden utilizar para tareas como la clasificación de imágenes, así como el procesamiento del lenguaje natural.

Danfo.js

Danfo.js es una biblioteca de código abierto que proporciona estructuras de datos de alto rendimiento utilizadas para manipular y procesar datos estructurados. Entre las mismas se encuentran funcionalidades como filtrados, agrupamientos, agregación y transformación de datos. Permite manejo de datos faltantes y realización de operaciones de limpieza de datos, logrando preparar los datos antes de su análisis o modelado.

Esta biblioteca está inspirada principalmente en Pandas.

Bibliotecas para procesamiento de lenguaje natural

NLP.js

NLP es una biblioteca de procesamiento del lenguaje natural para Node.js. Está diseñada para construir aplicaciones que requieren capacidades de procesamiento de lenguaje natural.

Natural

Esta biblioteca de procesamiento de lenguaje natural ofrece diversas funcionalidades que permiten aplicar técnicas de procesamiento de lenguaje natural de manera eficiente y efectiva. Estas funcionalidades incluyen la tokenización, que divide los textos en unidades más pequeñas como palabras o frases. Además, se incluye stemming para reducir las palabras a su raíz o forma base, para lograr normalizar el texto y mejorar la precisión en tareas de análisis de texto.

Comparación de Python vs JavaScript

En términos de capacidad de procesamiento, Python se destaca por su amplia variedad de bibliotecas optimizadas específicamente para el procesamiento de datos y el aprendizaje automático. Las bibliotecas detalladas anteriormente, Pandas, Scikit-learn, junto con NumPy hacen de Python una opción eficiente para tareas complejas dentro de la ciencia de datos, análisis estadísticos y modelos predictivos. Además, herramientas potentes que permiten evaluar, monitorear y probar un LLM, como Opik brindan SDK para la integración con este lenguaje.

Por su parte, JavaScript ofrece bibliotecas de gran utilidad, como TensorFlow.js pero no cuenta con una infraestructura tan robusta ni especializada como ocurre con Python para el procesamiento de grandes volúmenes de datos o la implementación de algoritmos avanzados de machine learning.

En cuanto al soporte de la comunidad a las bibliotecas mencionadas anteriormente, Python tiene una comunidad activa y una abundante cantidad de recursos disponibles, lo que facilita tanto el aprendizaje como la resolución de problemas, así como la continuidad y mantenibilidad de las mismas al tratarse de bibliotecas de código abierto. Aunque JavaScript también posee una comunidad activa, especialmente en el ámbito del desarrollo web, su uso en el campo de la ciencia de datos es más limitado, en comparación con Python.

Conclusiones finales

Luego de analizar las capacidades de ambos lenguajes, se concluye que Python es la opción adecuada para este proyecto por su robustez, la gran cantidad de recursos disponibles, y su especialización en áreas claves como el procesamiento de datos y el aprendizaje automático, y la comunidad activa que existe en torno a este lenguaje la cual garantiza un soporte continuo y recursos actualizados.

Si bien JavaScript cuenta con ciertas bibliotecas útiles como TensorFlow.js, su infraestructura para el procesamiento de grandes volúmenes de datos y la implementación de algoritmos avanzados es más limitada en comparación con Python. Un factor decisivo en esta elección fue la disponibilidad de herramientas de gran utilidad, como lo es Opik el cuál ofrece actualmente un SDK para Python pero carece de ecosistema para JavaScript. Esta herramienta proporciona funcionalidades clave para evaluar, monitorear y probar modelos de lenguaje de manera eficiente.

Frente a lo antes comentado, se considera que Python es la opción más adecuada para utilizar durante este proyecto.

Esta sección se desarrolló con base en la bibliografía consultada en [47] - [65].

13.6.3. Fine Tuning vs RAG

Restricciones del Fine-Tuning

Aunque el fine-tuning podría parecer una solución interesante en algunos casos, presenta varias limitaciones que lo hacen menos adecuado para nuestro caso de uso:

Falta de Flexibilidad con Datos Dinámicos: El fine-tuning implica ajustar los pesos del modelo incorporando datos específicos al proceso de entrenamiento. Además, si los usuarios actualizan frecuentemente sus datos (por ejemplo, reglas de negocio), sería necesario reentrenar el modelo continuamente. Esto genera un ciclo costoso y poco escalable, especialmente con muchos usuarios.

Costo de Entrenamiento: El entrenamiento de un modelo ajustado requiere infraestructura costosa, como GPUs o TPUs. Por otro lado, para cada usuario o conjunto de datos actualizados, debería realizarse un nuevo entrenamiento, lo que incrementa exponencialmente los costos operativos.

Dificultad para Mostrar Referencias: Una vez que el modelo está ajustado, no existe una manera directa de vincular las respuestas generadas con los datos originales. Esto limita la transparencia y dificulta que los usuarios confíen plenamente en las respuestas del modelo.

Escalabilidad Limitada: Gestionar modelos personalizados o ajustados para cada usuario se vuelve impráctico a medida que la base de usuarios crece. Alternativamente, crear un modelo general ajustado con datos de todos los usuarios podría causar problemas de aislamiento de datos y seguridad.

Comparativa con RAG

Dadas estas restricciones que hacen que Fine Tuning no sea lo más adecuado para nuestro caso de uso, se analizó una técnica (más profundamente analizada en su propia spike) llamada RAG.

RAG combina un LLM preentrenado con un sistema de recuperación de información, ofreciendo una solución dinámica y escalable para nuestro caso. Estas son sus principales ventajas:

Flexibilidad con Datos Dinámicos: Los datos de cada usuario se almacenan externamente (por ejemplo, en una base de datos o un vector store). Entonces, cada vez que alguien interactúa con el modelo, los datos relevantes se recuperan y se pasan como contexto al LLM para generar una respuesta. Esto asegura que las respuestas sean siempre precisas y reflejen la última información subida por los usuarios.

Costo Eficiente: Al no necesitar reentrenar el modelo, los costos operativos se reducen drásticamente. Además, el trabajo pesado se realiza en la etapa de recuperación, que es significativamente menos costosa que el entrenamiento.

Capacidad para Mostrar Referencias: Como las respuestas se generan a partir de datos recuperados en tiempo real, es posible mostrar claramente las referencias (por ejemplo, "esta información proviene del documento X o la regla Y"). Esto aumenta la transparencia y mejora la confianza del usuario final.

Escalabilidad: RAG escala bien con una base de usuarios grande, ya que los datos de cada usuario pueden gestionarse de manera aislada en un sistema de recuperación. Por ejemplo, podríamos tener índices separados por usuario o incluir identificadores únicos que permitan asociar cada consulta con un conjunto específico de datos.

Velocidad de Implementación: Aprovechar un modelo pre entrenado reduce el tiempo necesario para implementar el sistema. La configuración de un vector store y la integración con un LLM es menos compleja que un pipeline completo de fine-tuning.

Conclusión

Con base en las limitaciones del fine-tuning y las ventajas de RAG, pareciera ser que este último es más adecuado para nuestro caso de uso. En particular:

1. RAG es más adecuado para manejar datos específicos por usuario, especialmente si estos cambian con frecuencia.
2. La capacidad de mostrar referencias en tiempo real mejora significativamente la experiencia y confianza de los usuarios finales.
3. Los costos y tiempos de desarrollo son mucho más razonables al no depender de un reentrenamiento constante.

13.6.4. Retrieval Augmented Generation

RAG (Generación Aumentada por Recuperación, por sus siglas en inglés, Retrieval-Augmented Generation) es una técnica para aumentar el conocimiento de un modelo de lenguaje grande (LLM, por sus siglas en inglés, Large Language Model) mediante datos adicionales. Combina técnicas de recuperación de información y generación de texto para mejorar la precisión, relevancia y contexto de las respuestas generadas por sistemas de inteligencia artificial.

Los modelos de lenguaje grande (LLMs) poseen limitaciones en cuanto a su conocimiento. En primer lugar, su conocimiento se basa únicamente en los datos disponibles hasta el momento de su entrenamiento. Asimismo, debido a la falta de acceso a datos privados o dinámicos, no pueden razonar directamente sobre información que no haya sido incluida en su conjunto de entrenamiento.

En casos donde se requiere que el modelo procese datos nuevos o específicos, es necesario brindarle estos datos de manera dinámica. Para esto, la técnica RAG permite aumentar el conocimiento del modelo con información externa relevante.

Enfoques actuales para la implementación de RAG

Actualmente, los enfoques más destacados para la implementación de RAG se basan en frameworks especializados como *LangChain* y *Haystack*, los cuales permiten integrar modelos generativos con técnicas avanzadas de recuperación de información. A continuación, se describen sus principales características.

Haystack

Haystack es un framework integral diseñado para desarrollar pipelines robustos y listos para producción utilizando LLMs y otras tecnologías avanzadas de procesamiento de lenguaje natural. Este framework permite desarrollar soluciones de búsqueda personalizadas para una amplia variedad de casos de uso, combinando recuperación de información y generación de texto.

Ventajas

Flexibilidad en casos de uso: Haystack se adapta a múltiples escenarios, desde responder preguntas hasta realizar búsquedas documentales semánticas, lo que permite una experiencia personalizada.

Implementación end-to-end: Proporciona una infraestructura completa para construir soluciones que van desde la recuperación de datos hasta la generación de respuestas precisas y contextualizadas.

LangChain

LangChain es un framework para desarrollar aplicaciones impulsadas por LLMs. Este framework pone su enfoque en el manejo del contexto y la integración de datos externos, permitiendo que los modelos generativos sean más dinámicos y relevantes para diversos casos de uso.

Ventajas

Flexibilidad en el diseño de aplicaciones: Permite construir aplicaciones personalizadas que combinen múltiples pasos de procesamiento de datos y generación de respuestas.

Enfoque modular: Cada componente puede configurarse y adaptarse individualmente, proporcionando un control granular sobre el flujo de trabajo.

Escalabilidad y adaptabilidad: Es ideal para casos complejos donde los modelos deben trabajar con grandes volúmenes de datos o requerir consultas específicas y detalladas.

Comparación

Aunque tanto *Haystack* como *LangChain* ofrecen capacidades sólidas para implementar sistemas basados en RAG, existen diferencias importantes que influyen en la elección de un framework u otro, dependiendo del caso de uso y los requisitos del proyecto.

Haystack

Mejor desempeño general: En pruebas comparativas, *Haystack* suele demostrar un rendimiento superior en términos de recuperación y generación de información.

Facilidad de uso: Su sistema es más intuitivo y sencillo para los desarrolladores, lo que facilita su adopción y reduce tiempos de implementación.

Documentación completa y clara: Ofrece una documentación extensa que facilita su uso.

Ideal para chatbots simples: Si el objetivo es desarrollar o mejorar un chatbot sin demasiadas complejidades, *Haystack* brinda todas las herramientas necesarias eficientemente y sin complicaciones adicionales.

LangChain

Integración con agentes complejos: Su marco de agentes está diseñado específicamente para orquestrar múltiples servicios y procesos.

Configuración avanzada: Es ideal para escenarios en los que es necesario manejar flujos de trabajo sofisticados que combinan múltiples servicios o etapas de procesamiento.

Cuando el proyecto se centra en soluciones como chatbots o sistemas básicos de búsqueda y generación, *Haystack* ofrece una experiencia más directa y eficiente. En situaciones donde es necesario orquestrar numerosos servicios o integrar flujos avanzados, *LangChain* proporciona una mayor flexibilidad y capacidades de personalización.

En conclusión, mientras que *Haystack* destaca por su facilidad de uso y desempeño general, *LangChain* se posiciona como la mejor opción para configuraciones avanzadas que requieren una integración sofisticada.

Proof of Concept (PoC)

Para la implementación del Proof of Concept, se decidió utilizar *Haystack* como framework principal debido a que se alinea más con las necesidades actuales del proyecto. Esto asegura un desarrollo ágil y un enfoque claro en la evaluación de la efectividad del modelo. Se evaluaron las mismas preguntas tanto utilizando RAG como consultando directamente al modelo empleado, para poder comparar las respuestas.

Comparación de Respuestas: RAG vs. Consultas Directas al Modelo

Respuestas con RAG:

Pregunta: ¿Cómo se riega un coronilla?

Respuesta: El *Scutia buxifolia* no requiere mucho riego en ningún momento del año.

Pregunta: ¿Dónde crece mejor un coronilla?

Respuesta: El *Scutia buxifolia* prefiere los lugares iluminados, con algo de sol directo en algún momento del día; no sol pleno.

Pregunta: ¿Qué pasa si corto una rama de un coronilla?

Respuesta: Se activan las yemas latentes y de cada corte salen 2 nuevas ramas.

Respuestas directas al modelo:

Pregunta: ¿Cómo se riega un bonsái coronilla?

Respuesta: Un bonsái coronilla se riega profundamente y menos frecuentemente, preferiblemente durante la mañana o la tarde, con agua limpia y sin contaminantes... *(respuesta larga con información genérica y errónea).*

Pregunta: ¿Dónde crece mejor un coronilla?

Respuesta: Un coronilla (*Cornus mas*) crece mejor en climas templados y frescos con veranos suaves e inviernos moderados... *(confunde la especie y proporciona información incorrecta).*

Pregunta: ¿Qué pasa si corto una rama de un coronilla?

Respuesta: Si se corta una rama de un coronilla (*Cornus mas*), se puede esperar que se regeneren nuevas ramas desde la base de la planta... *(respuesta incorrecta y sin relación directa con el contexto real del *Scutia buxifolia*).*

Conclusión

El uso de RAG (Retrieval-Augmented Generation) proporciona respuestas mucho más precisas y alineadas con el contexto específico de los documentos proporcionados. Mientras que el modelo consultado directamente responde con información genérica, e

incluso incorrecta, debido a que se basa únicamente en su conocimiento preentrenado, RAG permite enriquecer las respuestas con datos relevantes y actualizados extraídos de fuentes confiables.

Por lo tanto, en casos donde la precisión y relevancia contextual son críticas, como al trabajar con dominios específicos o información de nicho, RAG demuestra ser una herramienta indispensable.

Esta sección se desarrolló con base en la bibliografía consultada en [66] - [71].

13.6.5. Bases de datos vectoriales

Una base de datos vectorial es un tipo de sistema diseñado para almacenar y buscar información en forma de vectores, es decir, representaciones numéricas de datos (como texto, imágenes o audio). Cada vector puede entenderse como una “cajita de información” que contiene el significado de un elemento. Cuanto más cercanas están dos cajas en el espacio vectorial, más relacionadas están entre sí en cuanto a su contenido o contexto.

Esto permite realizar búsquedas semánticas, encontrando resultados no solo idénticos, sino también conceptualmente similares.

Métodos de indexación

Las bases de datos vectoriales utilizan distintos métodos de indexación para organizar los vectores y acelerar las búsquedas de similitud. A continuación, se resumen los principales enfoques.

Locality Sensitive Hashing (LSH)

Agrupar los vectores en buckets según su similitud, utilizando funciones hash diseñadas para preservar la cercanía entre vectores.

Cuando se consulta un vector nuevo, se calcula su hash para ubicarlo en el bucket más similar, y solo se busca dentro de ese grupo.

Una analogía útil es la de una biblioteca: los libros están clasificados por género, por lo que si buscamos uno en particular, solo revisamos su sección y no toda la biblioteca. Este método es eficiente y mantiene agrupados los elementos parecidos.

IVF (Inverted File Index)

Método aproximado que divide el espacio vectorial en clústeres mediante un algoritmo de agrupamiento. Permite crear índices rápidamente y limita el conjunto de vectores a explorar durante la búsqueda, mejorando la velocidad sin perder demasiada precisión.

HNSW (Hierarchical Navigable Small World)

Basado en grafos jerárquicos, construye una estructura de varios niveles que permite encontrar vecinos cercanos en espacios de alta dimensionalidad. Es uno de los métodos más precisos y rápidos, aunque su creación y mantenimiento son intensivos en recursos, especialmente cuando los datos cambian con frecuencia.

ANNOY (Approximate Nearest Neighbors Oh Yeah)

Utiliza árboles binarios como estructura base, dividiendo recursivamente el espacio vectorial. Ofrece un buen equilibrio entre velocidad y precisión para conjuntos estáticos de datos.

Comparación de métodos de indexación

Método	Precisión	Velocidad	Complejidad	Costo	Mejor caso de uso
FLAT	100%	Lento	Bajo	Bajo	Conjuntos pequeños o cuando se requiere precisión total
IVF_FLAT	Moderada-Alta	Rápida	Media	Bajo-Medio	Grandes volúmenes, equilibrio entre velocidad y precisión
HNSW	Alta	Muy rápida	Alta	Media	Consultas en tiempo real sobre datos de alta dimensión

ANNOY	Moderada- Alta	Rápida	Media	Bajo	Datos estáticos con buena relación velocidad/precisión
PQ	Baja- Moderada	Muy rápida	Alta	Media	Escenarios con miles de millones de vectores, donde prima la velocidad
IVF_PQ	Moderada	Muy rápida	Alta	Media	Conjuntos masivos con limitaciones de recursos
LSH	Baja- Moderada	Media- Rápida	Media	Bajo	Búsquedas aproximadas en espacios de baja dimensión
Híbrido	Alta	Media	Muy alta	Alta	Consultas complejas con filtros y relevancia combinada

Tabla 19: Comparación de métodos de indexación

Opciones de bases de datos vectoriales

Pinecone

Servicio cerrado (*closed-source*) y de alto rendimiento. Como ventajas tiene buen desempeño y escalabilidad. Como desventajas puede ser costoso y con escasas opciones de personalización.

Milvus

Es una Base de datos *open-source* diseñada para manejar grandes volúmenes de datos vectoriales. Como ventajas, soporta múltiples métodos de indexación (IVF, HNSW, ANNOY), ofrece búsquedas rápidas y escalables, e integra fácilmente con frameworks de IA como *TensorFlow*, *PyTorch* y *Kubernetes*. Sin embargo, requiere infraestructura propia, aunque es completamente gratuita para auto alojar.

Weaviate

Es una Base vectorial *open-source* orientada a procesamiento de lenguaje natural y búsqueda semántica. Como ventaja, incluye API *GraphQL*, módulos listos para texto,

imagen y video, y permite crear módulos personalizados. Cuenta con uso gratuito en modo *self-host*, pero con costos por dimensión en la nube o al usar el servicio oficial.

Destaca por su precisión y su capacidad de entender el contexto de datos no estructurados.

Qdrant

Es una Base vectorial *open-source* enfocada en rendimiento en tiempo real. Permite filtrado y búsqueda híbrida, combinando similitud vectorial con criterios tradicionales.

Es gratuita para auto alojar, pero con costos mensuales al usar versiones en la nube o híbridas. Según la comunidad es ideal para entornos de datos dinámicos y sistemas de recomendación de alta escala.

Conclusión y elección de tecnología

Para este proyecto se decidió iniciar las pruebas con Milvus, aprovechando su integración con Node.js y su soporte para múltiples índices vectoriales. En caso de no cumplir con los requerimientos de rendimiento o administración, se considerará Weaviate como alternativa viable.

Esta sección se desarrolló con base en la bibliografía consultada en [72] - [74].

13.6.6. Comparativa entre Knowledge Graphs y Bases de Datos Vectoriales

En el contexto de sistemas de inteligencia artificial y recuperación de información, los Knowledge Graphs (grafos de conocimiento) y las Bases de Datos Vectoriales representan dos enfoques complementarios para almacenar y relacionar información. Mientras los grafos de conocimiento modelan relaciones explícitas entre entidades, las bases vectoriales se centran en medir similitud semántica entre fragmentos de información representados como vectores.

Sin embargo, las bases vectoriales, al almacenar información como vectores discretos de fragmentos individuales de texto, tienden a perder las relaciones complejas y los vínculos contextuales que existen entre esos fragmentos.

Comparativa general

Aspecto	Knowledge Graph	Vector Database
Estructura de datos	Basada en grafos, con nodos (entidades) y aristas (relaciones).	Basada en vectores de alta dimensionalidad que representan similitud semántica.
Fortalezas	- Representación clara de relaciones.- Ideal para jerarquías y trazabilidad.- Alta explicabilidad.	- Búsquedas por similitud y emparejamiento aproximado.- Maneja bien datos no estructurados (texto, imágenes).- Escalable para grandes volúmenes.
Consulta	Usa lenguajes estructurados como SPARQL .Excelente para consultas relacionales complejas.	Usa medidas de similitud vectorial (p. ej., <i>cosine similarity</i>).Ideal para consultas semánticas y contextuales.
Casos de uso	- Ontologías, taxonomías y jerarquías.- Seguimiento de procedencia o trazabilidad.	- Recomendaciones y sistemas de búsqueda.- Casos donde se prioriza la recuperación semántica .
Explicabilidad	Alta: las relaciones son explícitas y trazables.	Limitada: los puntajes de similitud no siempre son interpretables para el usuario.
Flexibilidad	Rígido: requiere esquemas definidos y datos estructurados.	Flexible: admite datos no estructurados sin esquema fijo.
Escalabilidad	Moderada: puede degradarse con grafos muy grandes y complejos.	Alta: diseñada para operar sobre conjuntos masivos y distribuidos.
Integración con LLMs	Facilita consultas estructuradas para razonamiento sobre relaciones.	Se integra de forma natural con RAG (Retrieval-Augmented Generation) para búsquedas semánticas.

Complejidad de implementación	Alta: requiere modelado de grafos y dominio de lenguajes como SPARQL o Cypher.	Media: requiere generar embeddings previos; herramientas como Pinecone o Weaviate simplifican el proceso.
Costo	Moderado: depende del tamaño del grafo y la complejidad de las consultas.	Variable: depende del volumen de embeddings y almacenamiento vectorial.
Mantenimiento	Requiere actualización continua del esquema y validación de relaciones.	Necesita reentrenar embeddings cuando cambian los datos, pero depende menos de esquemas rígidos.

Tabla 20: Comparación entre Knowledge Graphs y Bases de Datos Vectoriales

Recomendaciones de uso

Usar un Knowledge Graph cuando:

- Los datos son estructurados y dependen de relaciones explícitas u ontologías.
- Se requiere explicabilidad y trazabilidad de las respuestas.
- El dominio se beneficia de un modelado jerárquico (por ejemplo, especies, enfermedades o técnicas en bonsáis).

Usar una Base de Datos Vectorial cuando:

- Los datos incluyen elementos no estructurados (texto, imágenes, audio).
- La similitud semántica es clave (por ejemplo, consultas en lenguaje natural o recomendaciones).
- La escalabilidad y el rendimiento en grandes volúmenes de información son esenciales.

Tecnologías recomendadas

Categoría	Tecnologías
Graph Database	Neo4j Community Edition (open-source, intuitiva).
Bibliotecas RDF	RDFLib (Python). Apache Jena (Java).
Extracción de entidades y relaciones	SpaCy (gratuita, NER). Hugging Face Transformers (BERT, T5). GPT-4 (OpenAI API) para extracción avanzada.
Embeddings opcionales	Sentence Transformers para similitud semántica.
Lenguajes de consulta	SPARQL (RDF) y Cypher (Neo4j).
Visualización	Neo4j Bloom (interactiva) y Gephi (análisis estático).
Ejemplos de código	Repositorio: https://github.com/neo4j/NaLLM Ejemplo en LangChain: https://python.langchain.com/v0.1/docs/use_cases/graph/constructing/

Tabla 21: Tecnologías recomendadas

Conclusión

En síntesis, los Knowledge Graphs y las Bases Vectoriales ofrecen aproximaciones diferentes pero complementarias:

- Los Knowledge Graphs aportan estructura, explicabilidad y razonamiento sobre relaciones explícitas.
- Las Bases Vectoriales destacan en búsquedas semánticas, flexibilidad y escalabilidad para datos no estructurados.

En sistemas complejos, la integración de ambos enfoques —por ejemplo, combinar un grafo para la jerarquía taxonómica de especies con una base vectorial para búsqueda semántica— ofrece resultados más ricos, precisos y trazables.

Esta investigación se basó en las fuentes [75] - [78].

13.6.7. LangChain

LangChain y el preprocesamiento de datos

LangChain es un framework para desarrollar aplicaciones impulsadas por LLMs. Este framework pone su enfoque en el manejo del contexto y la integración de datos externos, permitiendo que los modelos generativos sean más dinámicos y relevantes para diversos casos de uso.

LangChain busca simplificar todas las etapas del ciclo de vida de una aplicación LLM, desde el desarrollo y la preparación para producción hasta el despliegue. Su enfoque principal, por lo tanto, es facilitar la integración de LLMs en aplicaciones, proporcionando herramientas y componentes que permiten manejar la interacción con estos modelos de manera eficiente.

Sin embargo, LangChain no está especialmente orientado al preprocesamiento de datos. Si bien incluye algunas herramientas que permiten transformar y estructurar datos en el contexto de interacciones con modelos de lenguaje, su enfoque principal es optimizar la interacción y el manejo de contexto, no el procesamiento o limpieza de datos. Para tareas de preprocesamiento complejas, es recomendable utilizar otras herramientas especializadas en procesamiento de lenguaje natural.

Integración con bases vectoriales y grafos de conocimiento

LangChain es un framework para desarrollar aplicaciones impulsadas por LLMs. Este framework pone su enfoque en el manejo del contexto y la integración de datos externos, permitiendo que los modelos generativos sean más dinámicos y relevantes para diversos casos de uso.

Como se investigó en el spike correspondiente a RAG (Retrieval Augmented Generation), RAG es una técnica que mejora los modelos de lenguaje al combinarlos con bases de conocimiento externas. Al realizar una consulta, los sistemas RAG primero buscan información relevante en una base de conocimiento. El sistema luego incorpora esta información recuperada en el prompt del modelo. El modelo usa el contexto proporcionado para generar una respuesta a la consulta.

Para lograr comprender cómo se usa LangChain, se comenzó realizando una prueba de concepto de una aplicación sencilla de preguntas y respuestas sobre una fuente de datos de texto.

Esta investigación se basó en las fuentes [79] - [83].

13.7. Evaluaciones del sistema

13.7.1. Resultados de pruebas

Para entender cómo fueron impactando las decisiones de diseño en la calidad del sistema, se seleccionaron cuatro iteraciones principales, considerados los puntos de inflexión donde se realizaron cambios con mayor significancia.

Iteración 1: Respuestas conservadoras, casi sin alucinaciones

En esta primera pasada, el sistema priorizó evitar alucinaciones aun a costa de no responder. El efecto se ve en las métricas: hallucination ≈ 0.03 (muy baja), pero context_precision ≈ 0.23 y context_recall ≈ 0.27 . El patrón cualitativo fue claro: la gran mayoría de las salidas fueron “no tengo información”, incluso cuando había pistas útiles en el contexto. En los ejemplos, ante consultas de trasplante o sustratos, el modelo se replegó sin arriesgar una síntesis.

Conclusión: excelente fidelidad al contexto, poca utilidad práctica por omisión de respuesta.

Fuente: elaboración propia con Opik (dashboard de evaluaciones).

Feedback score	physical_cod_8689
context_precision_metric	0.228571428
context_recall_metric	0.271428571
hallucination_metric	0.028571428

Ilustración 62: Métricas de iteración 1

Iteración 2: Se fuerza “decir algo” y se degrada la calidad

Probamos el enfoque opuesto: incentivar que el modelo complete una respuesta aun con contexto parcial. El resultado empeoró: se observa hallucination ≈ 0.92 (muy alta) con precision = 0 y recall ≈ 0.03 . El modelo empezó a generalizar sin respaldo o a mezclar especies/criterios no presentes en el contexto recuperado.

Decisión: revertir el sesgo y volver al enfoque conservador.

Fuente: elaboración propia con Opik (dashboard de evaluaciones).

Feedback score	boiling_swamp_9361
context_precision_metric	0
context_recall_metric	0.033333333
hallucination_metric	0.916666667

Ilustración 63: Métricas de iteración 2

Iteración 3: Enfoque por tipos de pregunta 1 y 2 ya resueltos

Estas pruebas se ejecutaron una vez que las preguntas de tipo 1 y 2 estaban considerablemente resueltas. En este caso, las métricas mejoraron notablemente: precision ≈ 0.44 , recall ≈ 0.51 con hallucination ≈ 0.006 . Se mantuvo una baja alucinación y el sistema extrajo y sintetizó evidencia relevante del contexto.

Sin embargo, para el resto de los tipos de pregunta el modelo continuó respondiendo “no tengo información”, lo que contuvo la alucinación pero también limitó precisión y recall a nivel global del conjunto.

Conclusión: al acotar el dominio y mejorar la curaduría/recuperación, suben precisión y recall sin sacrificar fidelidad.

Fuente: elaboración propia con Opik (dashboard de evaluaciones).

Feedback score	strategic_veranda_968
context_precision_metric	0.44
context_recall_metric	0.5125
hallucination_metric	0.00625

Ilustración 64: Métricas de iteración 3

Iteración 4: Se incorporan tipos de pregunta 3 y 4 y sigue la mejora

Sumamos tipos de pregunta 3 y 4, ya con mejores patrones de respuesta y controles de evidencia. En este caso se observa precisión ≈ 0.66 , recall ≈ 0.77 y hallucination ≈ 0.056 . El sistema cubre más información necesaria (recall) y selecciona mejor lo pertinente (precisión), manteniendo un nivel de alucinación.

Conclusión: la extensión gradual por tipo de pregunta escala calidad sin disparar alucinaciones.

Fuente: elaboración propia con Opik (dashboard de evaluaciones).

Feedback score	northern_ridge_9219
context_precision_metric	0.66
context_recall_metric	0.772222222
hallucination_metric	0.055555556

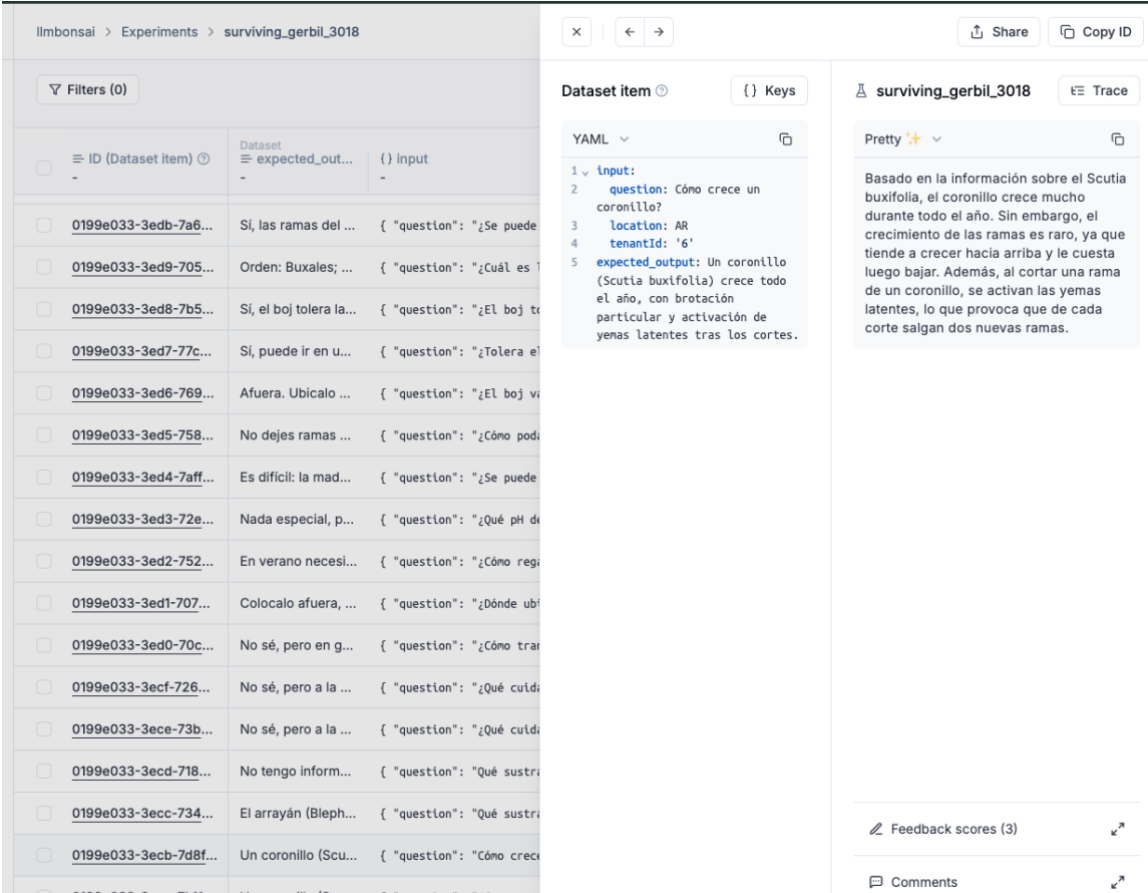
Ilustración 65: Métricas de iteración 4.

13.7.2. Alcance y configuración

Este anexo documenta el material de soporte de la evaluación automática: conjunto de pruebas, configuración de scoring y resultados consolidados de la última iteración principal. Incluimos capturas del panel de Opik, una síntesis de métricas agregadas y un ejemplo representativo del dataset en formato estructurado.

Resultados globales de la última evaluación

La Ilustración 66 muestra la visión general por experimento en Opik (Context Precision, Context Recall y Hallucination).



The screenshot displays the Opik interface for an experiment named 'surviving_gerbil_3018'. On the left, a table lists dataset items with columns for ID, Dataset (expected_output), and input. The right panel shows a detailed view of a dataset item, including its YAML structure and a rendered output.

ID (Dataset item)	Dataset (expected_output)	input
0199e033-3edb-7a6...	Sí, las ramas del ...	{ "question": "¿Se puede...
0199e033-3ed9-705...	Orden: Buxales; ...	{ "question": "¿Cuál es ...
0199e033-3ed8-7b5...	Sí, el boj tolera la...	{ "question": "¿El boj to...
0199e033-3ed7-77c...	Sí, puede ir en u...	{ "question": "¿Tolera el...
0199e033-3ed6-769...	Afuera. Ubicalo ...	{ "question": "¿El boj va...
0199e033-3ed5-758...	No dejes ramas ...	{ "question": "¿Cómo pod...
0199e033-3ed4-7aff...	Es difícil: la mad...	{ "question": "¿Se puede...
0199e033-3ed3-72e...	Nada especial, p...	{ "question": "¿Qué ph de...
0199e033-3ed2-752...	En verano necesi...	{ "question": "¿Cómo reg...
0199e033-3ed1-707...	Colocalo afuera, ...	{ "question": "¿Dónde ubi...
0199e033-3ed0-70c...	No sé, pero en g...	{ "question": "¿Cómo tra...
0199e033-3ecf-726...	No sé, pero a la ...	{ "question": "¿Qué cuid...
0199e033-3ece-73b...	No sé, pero a la ...	{ "question": "¿Qué cuid...
0199e033-3ecd-718...	No tengo inform...	{ "question": "Qué sustra...
0199e033-3ecc-734...	El arrayán (Bleph...	{ "question": "Qué sustra...
0199e033-3ecb-7d8f...	Un coronillo (Scu...	{ "question": "¿Cómo crece...

```
YAML
1 input:
2   question: ¿Cómo crece un
3   coronillo?
4   location: AR
5   tenantId: '6'
6   expected_output: Un coronillo
7   (Scutia buxifolia) crece todo
8   el año, con brotación
9   particular y activación de
10  yemas latentes tras los cortes.
```

Basado en la información sobre el Scutia buxifolia, el coronillo crece mucho durante todo el año. Sin embargo, el crecimiento de las ramas es raro, ya que tiende a crecer hacia arriba y le cuesta luego bajar. Además, al cortar una rama de un coronillo, se activan las yemas latentes, lo que provoca que de cada corte salgan dos nuevas ramas.

Feedback scores (3)

Comments

Ilustración 66: Visión general del último experimento realizado.

Casos de prueba: estructura y ejemplo

El dataset de evaluación se compone de ítems con: *question*, *metadatos* (como, *location* y *tenant_id*), y *expected_output*. A continuación se muestra un ejemplo (representativo) del formato utilizado:

```

Test case

{
  "input": {
    "question": "Cómo crece un coronilla?",
    "location": "UY",
    "tenantId": "6"
  },
  "expected_output": "Un coronilla (Scutia buxifolia)
crece todo el año, con brotación particular y activación de
yemas latentes tras los cortes."
},
},

```

Ilustración 67: Ejemplo representativo del formato de prueba empleado en Opik.

Para no sobrecargar el documento, se incluye solo un par de ejemplos; el dataset completo queda referenciado en el repositorio del proyecto.

13.8. Cronogramas del proyecto

13.8.1. Cronograma versión 1

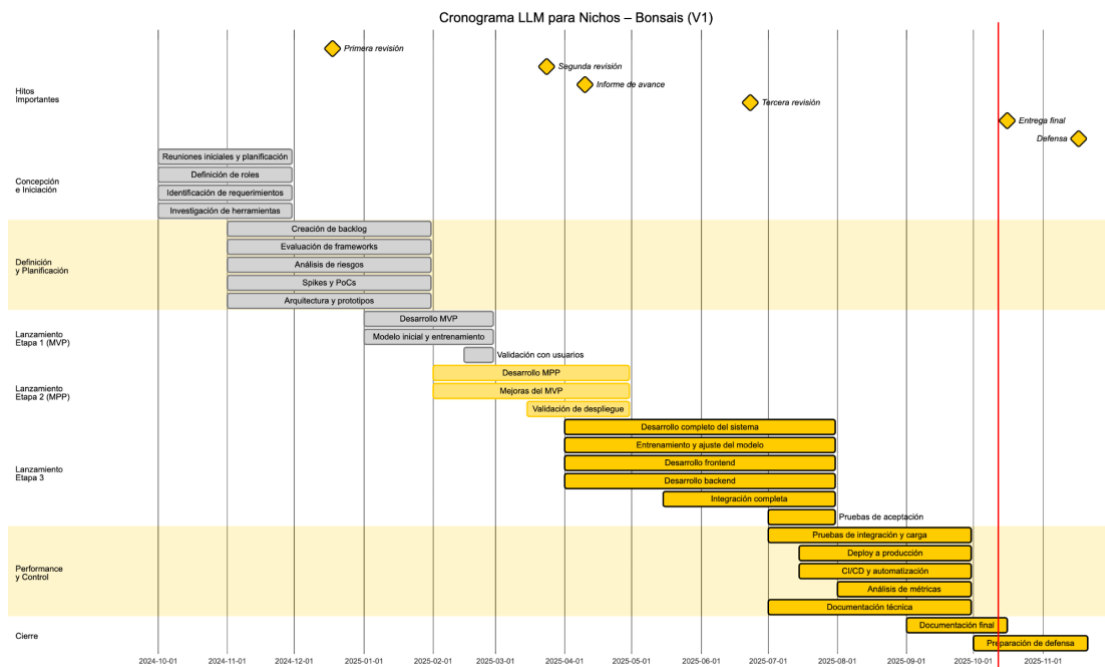


Ilustración 68: Cronograma del proyecto versión 1

13.8.2. Cronograma versión 2

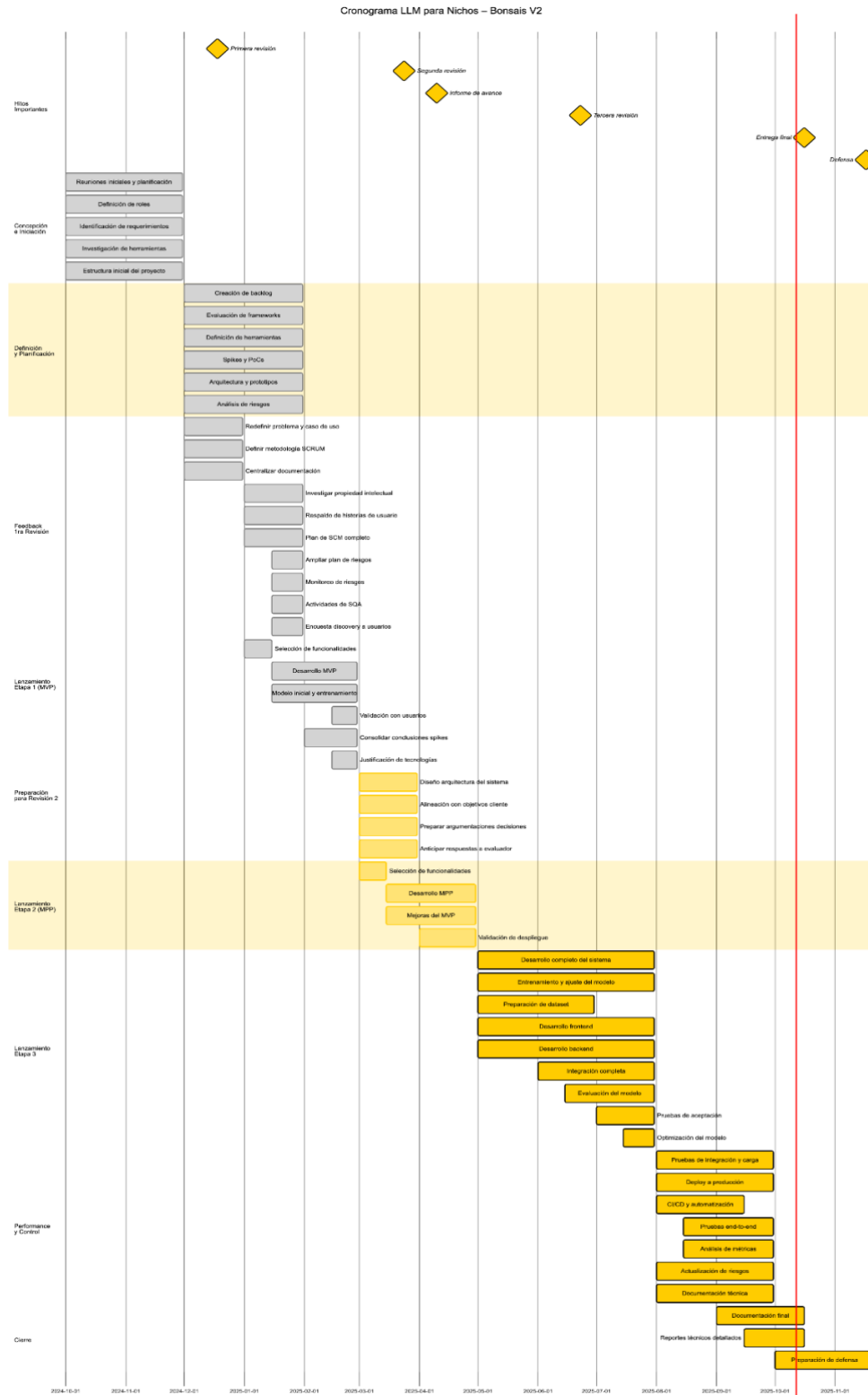


Ilustración 69: Cronograma del proyecto versión 2

13.8.3. Cronograma versión 3

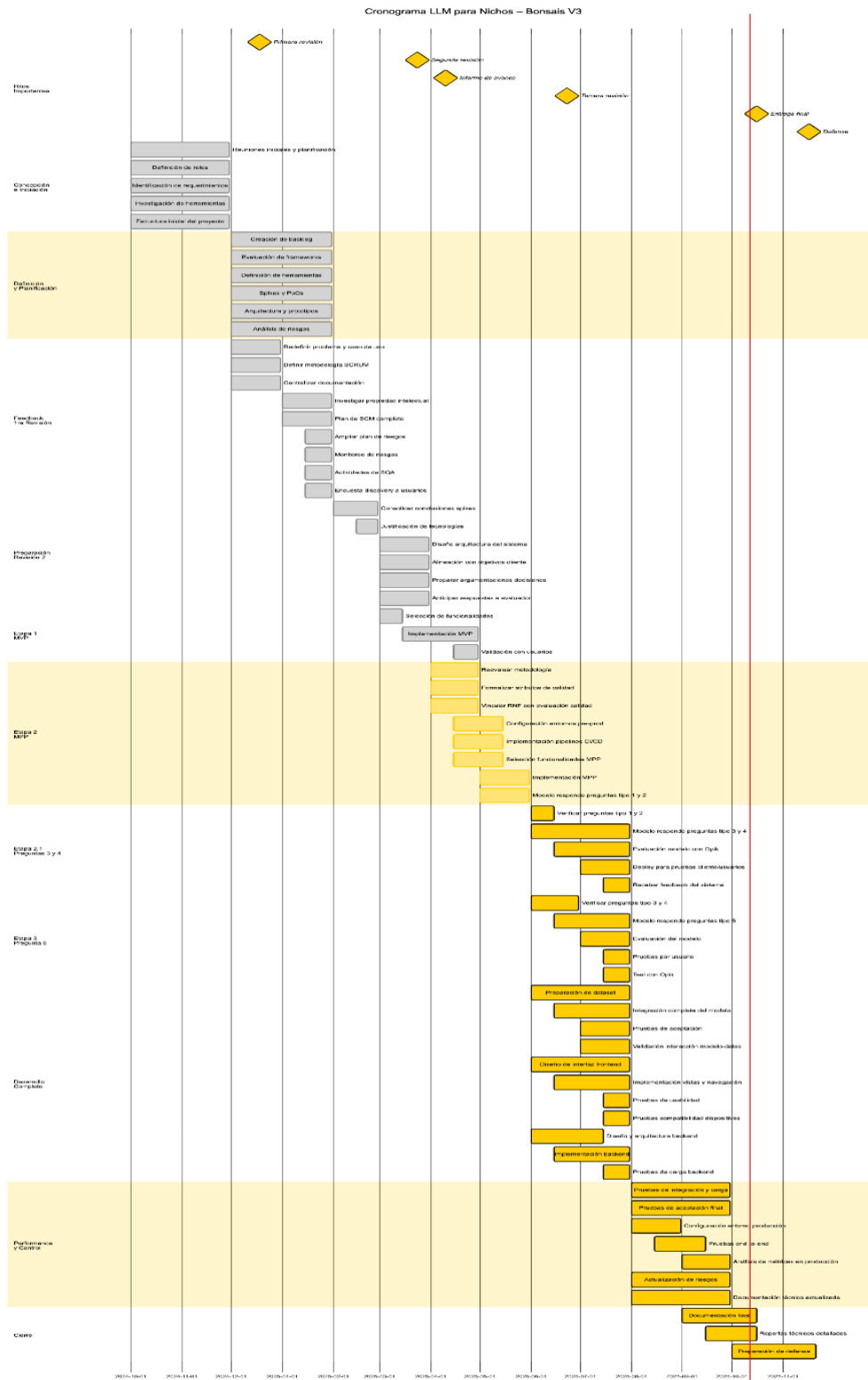


Ilustración 70: Cronograma del proyecto versión 3

13.9. Gestión de riesgos: notas extraídas de reuniones de análisis

Este documento detalla los riesgos potenciales identificados para el proyecto de LLM para Nichos, así como su evaluación, priorización y las estrategias para mitigarlos o gestionarlos de forma proactiva en caso de que ocurran.

Estrategia de gestión de riesgos

Con el fin de garantizar un enfoque proactivo en la gestión de riesgos, se decidió llevar a cabo una revisión periódica de los mismos al finalizar cada sprint, como parte de las reuniones de retrospectiva.

Durante estas reuniones, el equipo dará seguimiento analizará los riesgos identificados previamente, evaluará su estado, su ocurrencia durante el sprint y determinará si es necesario modificar las acciones de mitigación o incorporar nuevos riesgos detectados. Así se podrá asegurar una gestión continua que permitirá minimizar las incertidumbres y maximizar la eficiencia del desarrollo.

Riesgos Identificados

Consideramos que un riesgo es un evento o condición incierta que, de ocurrir, puede producir un efecto positivo o negativo en alguno de los objetivos del proyecto, alcance, costo, cronograma o calidad. Estos riesgos están relacionados con la incertidumbre inherente a los proyectos de desarrollo.

A continuación, se listan los riesgos identificados que serán gestionados, organizados por categorías.

Riesgos Tecnológicos: 1.1 Falta de conocimiento en LLM , 1.2 Problemas de compatibilidad , 1.3 Escalabilidad de la solución , 1.4 Errores en el despliegue

Riesgos de Gestión: 2.1 Retrasos en las entregas, 2.1 Desacuerdos con el Product Owner: Cambios frecuentes en los requerimientos o falta de alineación. 2.2. Falta de comunicación. 2.3.Falta de claridad en los roles

Riesgos del Equipo: 3.1 Disponibilidad del equipo, 3.2 Desbalance en la carga de trabajo.

Riesgos Externos: 4.1 Cambios en las tecnologías usadas, 4.2 Interrupciones en servicios externos, 4.3 Costos de infraestructura, 4.4 Legislación y regulaciones de datos.

Eliminación de un riesgo

Como forma de gestión de riesgos, durante las reuniones de revisión de los mismos se podrá descartar un riesgo por considerarse como no aplicable, significando su remoción en el seguimiento activo de los mismos.

Para ello, se establece el siguiente criterio para su eliminación.

- En caso de que se tomen acciones correctivas o preventivas que eliminen completamente la posibilidad de que ocurra el riesgo, el mismo será descartado por la eliminación de la causa raíz.
- Por cambios en el contexto del proyecto, ya sea por factores externos o internos generan cambios de tal manera en que el riesgo ya no es relevante.
- Si se implementan medidas de mitigación que logran reducir el impacto del riesgo a un nivel en el que ya no presenta una amenaza significativa para el proyecto se procederá a su eliminación, ya que se logra reducir el impacto a un nivel aceptable.
- Si el riesgo estaba asociado a una etapa específica del proyecto, el mismo podrá ser eliminado luego de que dicha etapa haya sido finalizada sin que el riesgo haya ocurrido.
- En caso de que el equipo considere en consenso de que un cierto riesgo ya no es relevante o su probabilidad de ocurrencia es insignificante, el mismo podrá ser eliminado con una debida justificación.

Evaluación de Riesgos - Febrero

Cada riesgo será evaluado según:

- Probabilidad: Baja (1), Media (2), Alta (3)
- Impacto: Bajo (1), Medio (2), Alto (3)

La severidad se calculará como: **Severidad = Probabilidad x Impacto**

Riesgo	Probabilidad	Impacto	Severidad
1.1. Falta de conocimiento en LLM	3	3	9
1.2. Problemas de compatibilidad	2	3	6
1.3. Escalabilidad de la solución	1	3	3
1.4. Errores en el despliegue	2	3	6
2.1. Retrasos en entregas	1	2	2
2.2. Desacuerdos con el PO	1	2	2
2.3. Falta de comunicación	1	2	2
2.4. Falta de claridad en los roles	1	2	2
3.1. Disponibilidad del equipo	1	3	3
3.2. Desbalance en la carga de trabajo	2	2	4
4.1. Cambios en tecnologías usadas	1	3	3
4.2. Interrupciones en servicios externos	1	3	3
4.3. Costos de infraestructura	2	3	6
4.4. Legislación y regulaciones de datos	2	2	4

Tabla 22: Riesgos

Matriz de riesgos

A continuación se detalla la matriz de riesgos. Los colores denotan la severidad de los mismos, aquellos de color rojo corresponden a una severidad alta, los amarillos a una severidad media y finalmente las secciones verdes denotan una severidad baja.

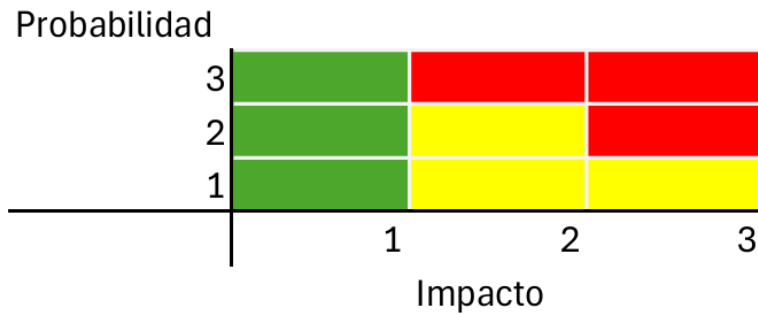


Ilustración 71: Matriz de riesgos

Visualización de riesgos

Con los riesgos identificados y los valores de probabilidad, impacto y severidad planteados para esta primera etapa, se generó la siguiente visualización de los mismos.

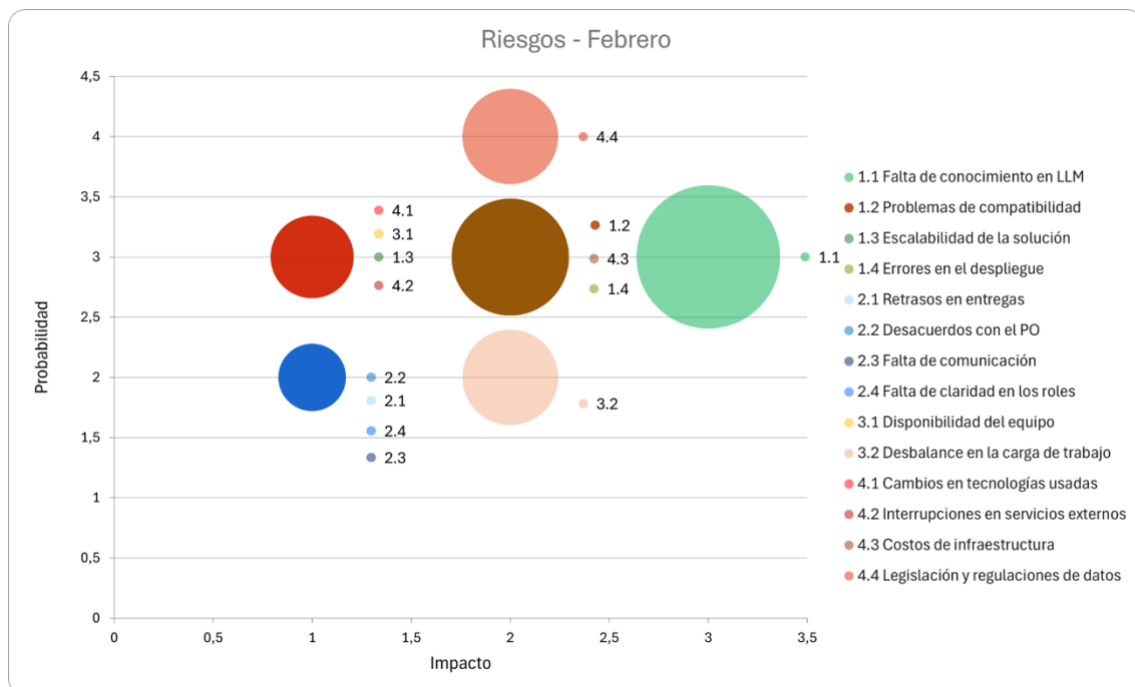


Ilustración 72: Visualización de riesgos.

Como mejora para la próxima revisión de los riesgos, consideramos que sería beneficioso modificar la valoración de probabilidad para poder lograr una mejor visualización de los mismos. Utilizar valores más amplios permitirá visualizarlos de mejor manera dentro de la gráfica.

Plan de Mitigación y Acción

Riesgos críticos (prioridad > 6)

1.1. Falta de conocimiento en LLM

- Mitigación
 - Establecer un backlog específico para el aprendizaje del LLM: tareas específicas que cubran investigación y experimentos.
 - Realizar spikes para investigar tecnologías relevantes y reducir la incertidumbre técnica.
 - Probar el LLM como un MVP para poder identificar limitaciones antes de entrar en el desarrollo completo.
 - Establecer métricas de evaluación claras
- Plan de acción
 - Consultar a expertos externos o comunidades técnicas si surgen problemas graves.

1.2. Problemas de compatibilidad

- Mitigación
 - Probar componentes en entornos aislados.
 - Validar integraciones tempranas.
 - Uso de estándares abiertos: Usar formatos de comunicación y almacenamiento estándar, para reducir fricciones en la integración.
 - Contar con un entorno de CI/CD: Configurar pipelines de integración continua para ejecutar pruebas automáticas en cada cambio.
- Plan de acción
 - Consultar documentación de las tecnologías involucradas para entender posibles incompatibilidades conocidas.
 - Probar los componentes de forma aislada para determinar si el problema es con la integración o con un componente individual.

- Rollback: si el problema es crítico y no se puede resolver rápidamente, revertir al estado funcional anterior utilizando el control de versiones.
- Aislar el componente afectado, y revisar los logs de error para obtener información detallada sobre el conflicto.

1.4. Errores en el despliegue

- Mitigación
 - Automatización del despliegue: Configurar un pipeline de CI/CD para minimizar errores humanos y garantizar que el proceso de despliegue sea repetible y consistente.
 - Usar contenedores (como Docker) para empaquetar tanto el *frontend* como el *backend*, asegurando que el entorno de desarrollo sea idéntico al de producción.
 - Usar sistemas de control de versiones para gestionar los cambios en el código.
 - Documentación y procedimientos claros.
 - Monitoreo y alertas sobre problemas en el despliegue.
- Plan de acción
 - Analizar los logs de despliegue para identificar el error específico.
 - Rollback a una versión estable, si presenta problemas críticos.
 - Revisar y corregir la configuración.
 - Revisión de dependencias.
 - Revisión de permisos y configuraciones de infraestructura.
 - Revisión post-mortem y ajustes en el proceso de infraestructura, configuración y pruebas.

4.3. Costos de infraestructura

- Mitigación
 - Configurar alertas de costos con proveedores de servicios en la nube.
 - Revisar periódicamente los recursos utilizados para identificar posibles sobras o recursos infrautilizados que se puedan ajustar o eliminar.
 - Implementar auto-scaling en los servicios de *backend* y en la infraestructura.
 - Optimizar consultas a bases de datos y almacenamiento en la nube.
 - Revisión periódica de la infraestructura.

- Plan de acción
 - Identificar el servicio o recurso que está causando el aumento inesperado de costos.
 - Auditar los recursos consumidos para verificar si hay recursos infrautilizados.
 - Verificar y ajustar las configuraciones de auto-scaling para evitar que los sistemas escalen innecesariamente.
 - Identificar recursos inactivos o infrautilizados y eliminarlos o reducir su tamaño.
 - Análisis post-mortem y ajustes a largo plazo.

Riesgos moderados (prioridad 4 - 6)

3.2. Desbalance en la carga de trabajo

- Mitigación
 - Establecer claramente roles y responsabilidades desde el inicio, con revisiones periódicas de la distribución de tareas.
 - Dividir las tareas del proyecto, y asignarlas de acuerdo a las habilidades de cada miembro del equipo.
 - Utilizar una herramienta de gestión de proyectos para asignar y priorizar tareas de forma clara.
 - Evaluar semanalmente la carga de trabajo de cada integrante del equipo en las reuniones recurrentes.
 - Trabajar en sprints, lo que permite realizar ajustes rápidos en la asignación de tareas.
 - Utilizar un Product Owner interno para gestionar y redistribuir tareas de forma eficiente.
- Plan de acción
 - Identificar rápidamente la sobrecarga, y redistribuir tareas a otros miembros del equipo de acuerdo con sus habilidades y disponibilidad.
 - Priorizar las tareas más críticas del proyecto, asegurando que las áreas claves sean atendidas primero.

4.4. Legislación y regulaciones de datos

- Mitigación
 - Identificar las regulaciones aplicables y asegurarse de que el proyecto esté alineado con las mismas.
 - Implementar medidas de seguridad como el cifrado de datos para proteger la información sensible.
 - Asegurar que solo personal autorizado tenga acceso a los datos sensibles.
 - Implementar una política de privacidad clara para los usuarios.
 - Recopilar la información mínima necesaria para el funcionamiento del sistema.
 - Implementar un proceso de consentimiento explícito de los usuarios antes de recopilar cualquier dato personal o sensible, especificando el uso de los datos en términos claros.
- Plan de acción
 - Si se detecta una violación de datos, notificar de inmediato a las autoridades regulatorias y, en caso de que sea necesario, a los usuarios afectados, según las leyes aplicables.
 - Evaluar el impacto de la violación para determinar la magnitud y las posibles consecuencias para los usuarios y la organización.
 - Tomar medidas correctivas para resolver la violación, como corregir cualquier vulnerabilidad en el sistema, actualizar protocolos de seguridad o revisar el acceso a los datos personales.
 - Informar a los usuarios afectados por la violación sobre qué datos fueron comprometidos, cómo están siendo protegidos, y qué medidas pueden tomar para protegerse (como cambiar contraseñas).
 - Proporcionar a las autoridades reguladoras un informe detallado sobre la violación, las medidas correctivas tomadas y las acciones que se implementarán para prevenir futuros incidentes.

Riesgos bajos (prioridad < 4)

1.3. Escalabilidad de la solución

- Mitigación
 - Asegurarse de que la arquitectura del sistema sea modular y flexible, permitiendo una expansión futura sin necesidad de una reestructuración

completa. Esto incluye separar los componentes críticos, como la base de datos, el *backend* y el LLM, para que puedan escalar de manera independiente si es necesario.

- Asegurarse de que las consultas a la base de datos estén bien optimizadas y que se utilicen índices para mejorar el rendimiento con un gran número de registros.
 - Realizar algunas pruebas de carga con una cantidad controlada de usuarios simulados, para detectar cuellos de botella.
 - Configurar servicios con auto-scaling.
 - Implementar herramientas de monitoreo para controlar el rendimiento del sistema en tiempo real.
- Plan de acción
 - Monitoreo proactivo para identificar rápidamente qué partes del sistema están experimentando sobrecarga.
 - Revisar los logs para identificar cuellos de botella.
 - Añadir nuevos servidores o instancias para distribuir la carga de trabajo.
 - Realizar una evaluación del sistema para ajustar la arquitectura y procesos según las necesidades de crecimiento.
 - Continuar realizando pruebas de carga y realizar ajustes basados en la demanda real.

2.1. Retrasos en entregas

- Mitigación
 - Utilizar metodologías ágiles para dividir el trabajo en tareas pequeñas y bien definidas.
 - Establecer metas claras y alcanzables para cada sprint y asignar responsabilidades específicas a cada miembro del equipo.
 - Realizar reuniones de planificación regulares.
 - Definir un cronograma con hitos y etapas.
- Plan de acción
 - En cuanto se identifiquen posibles retrasos, analizar la causa raíz para poder tomar decisiones y ajustar el plan.
 - Si hay retrasos importantes, reajustar las prioridades.

- Si los retrasos afectan las entregas o plazos, se debe comunicar de inmediato la situación, proporcionando una nueva estimación de tiempo y explicando las razones del retraso.
- Hacer una revisión de las metas del sprint y ajustarlas para que sean más realistas, sin comprometer la calidad.
- Realizar un plan de contingencia que contemple un margen de tiempo adicional en los sprints posteriores.

2.2. Desacuerdos con el PO

- Mitigación
 - Definición clara de los requerimientos del proyecto.
 - Establecer un proceso formal para el cambio de requerimientos: cualquier cambio en los requisitos debe ser documentado y comunicado claramente.
 - Usar herramientas de gestión de proyectos para registrar y hacer seguimiento de los cambios en los requerimientos.
 - Realizar reuniones regulares con el PO para revisar y ajustar las prioridades del proyecto.
 - Mantener una comunicación constante y efectiva con el PO, asegurándose de que las preguntas y comentarios se resuelvan rápidamente.
- Plan de acción
 - Organizar una reunión con el PO para revisar los cambios y asegurarse de que las prioridades sean claras.
 - Aclarar los objetivos a corto y largo plazo, y cómo los cambios pueden afectar el cronograma, alcance y recursos del proyecto.
 - Documentar los acuerdos alcanzados en cuanto a los cambios de los requerimientos.

2.3. Falta de comunicación

- Mitigación
 - Establecer canales de comunicación centralizados.
 - Documentar las decisiones tomadas y discutir las en las reuniones de sprint.
 - Fomentar la comunicación abierta y las preguntas.
 - Utilizar herramientas de gestión de proyectos

- Plan de acción
 - Si ocurre un incidente debido a falta de comunicación, organizar una reunión con las personas involucradas para poder obtener toda la información y rectificar el error.
 - Utilizar correctamente los canales de comunicación preestablecidos.
 - Designar a un facilitador o responsable que asegure la fluidez de la comunicación en temas críticos.

2.4. Falta de claridad en los roles

- Mitigación
 - Documentación centralizada con descripciones de tareas y roles claras.
 - Reuniones de alineación.
- Plan de acción
 - Ante dudas estar abiertos a revisiones y dar aclaraciones en equipo de las tareas a completar y roles que cumplir.
 - Redefinir / actualizar los criterios para que estén al día con las necesidades.
 - Incorporar reuniones de feedback periódicas para identificar confusiones relacionadas con roles y mejorar la definición de responsabilidades.

3.1. Disponibilidad del equipo

- Mitigación
 - Establecer horarios de trabajo que sean visibles para todo el equipo.
 - Realizar reuniones de sincronización periódicas
 - Avisar inasistencias.
- Plan de acción
 - Si un miembro del equipo está ausente por un tiempo prolongado (vacaciones, enfermedad, etc.), asignar tareas críticas a otros miembros del equipo de manera provisional hasta su regreso
 - Realizar una revisión periódica de la carga de trabajo de cada miembro del equipo para identificar posibles sobrecargas y redistribuir tareas en caso necesario.
 - Fomentar una cultura de transparencia, en la que los miembros del equipo se sientan cómodos al comunicar su disponibilidad

4.1. Cambios en tecnologías usadas

- Mitigación
 - Mantenerse al tanto de las actualizaciones, parches y nuevas versiones de las tecnologías que utilizamos.
 - Uso de versiones específicas: usar herramientas como Docker para gestionar entornos controlados.
 - Usar entornos virtuales para trabajar, de modo que se puedan controlar las dependencias y versiones sin afectar a otros componentes del sistema.
 - Mantener una documentación detallada sobre las versiones y configuraciones actuales del LLM y las herramientas asociadas.
- Plan de acción
 - Informarse inmediatamente sobre los detalles de la actualización y analizar su impacto potencial en el proyecto.
 - Determinar si la actualización es crítica y necesita ser implementada inmediatamente, o si se puede esperar a una futura versión más estable.
 - Si hay incompatibilidades, desarrollar un parche que permita que el sistema siga funcionando correctamente.

4.2. Interrupciones en servicios externos

- Mitigación
 - Realizar un análisis exhaustivo de los proveedores de servicios externos, asegurando que sean confiables, con un historial de alta disponibilidad y que ofrezcan acuerdos de nivel de servicios claro que especifiquen tiempos de inactividad aceptables.
 - Tener configuraciones de alta disponibilidad, en los servidores dónde se realice el despliegue.
 - Configurar alertas y monitoreo en tiempo real para los servicios externos críticos. Nos permitirá identificar caídas o problemas en servicios de manera temprana y actuar antes de que afecten al usuario.
- Plan de acción
 - Definir un procedimiento detallado para gestionar interrupciones en los servicios. Esto debe incluir un listado de pasos a seguir, desde la identificación del fallo hasta la resolución y la comunicación con los proveedores.

- Si un servicio externo se interrumpe, tener una alternativa funcional, como un servicio de reemplazo temporal o un mecanismo local de cacheo para mitigar el impacto de la caída del servicio.
- Configurar los servidores y servicios en la nube con escalabilidad automática. Esto permite que el sistema se ajuste a los picos de demanda y pueda sobrellevar interrupciones o caídas de servicios sin afectar la disponibilidad del proyecto.
- Después de una interrupción, realizar una revisión exhaustiva del incidente para entender las causas, evaluar la efectividad de las medidas tomadas y determinar cómo mejorar el sistema para prevenir futuras caídas.
- Hacer simulaciones de incidentes para asegurar que todos los miembros del equipo sepan cómo actuar durante una interrupción de servicio.

Evaluación de Riesgos - Marzo

Durante esta revisión, se retomaron los riesgos identificados anteriormente y se analizó la forma en que estaban siendo evaluados. Se observó que las escalas originales, de rango limitado (1 a 3), no nos permitían ilustrar con suficiente precisión los niveles de probabilidad e impacto en la matriz de riesgos. Esto generaba dificultades para poder ilustrar los riesgos tanto en la visualización en la gráfica, como en la priorización de los mismos.

Con el objetivo de mejorar el análisis y generar un mayor margen para la asignación de valores, se decidió modificar las escalas, extendiéndolas a un rango de 1 a 5. Esta actualización permite distinguir con mayor detalle entre niveles bajos, medios e intermedios, así como reflejar mejor las diferencias entre situaciones de riesgo similares pero no idénticas.

A continuación se detallan las nuevas escalas:

- Probabilidad: Rara (1), Improbable (2), Posible (3), Probable (4), Frecuente (5)
- Impacto: Insignificante (1), Menor (2), Moderado (3), Significativo (4), Crítico (5)

La severidad se continúa calculando de la siguiente manera: Severidad = Probabilidad x Impacto.

Riesgo	Probabilidad	Impacto	Severidad
1.1. Falta de conocimiento en LLM	2	3	6
1.2. Problemas de compatibilidad	0	2	0
1.3. Escalabilidad de la solución	3	2	6
1.4. Errores en el despliegue	4	3	12
2.1. Retrasos en entregas	2	3	6
2.2. Desacuerdos con el PO	4	2	8
2.3. Falta de comunicación	1	2	2
2.4. Falta de claridad en los roles	2	2	4
3.1. Disponibilidad del equipo	2	3	6
3.2. Desbalance en la carga de trabajo	1	2	2
4.1. Cambios en tecnologías usadas	1	3	3
4.2. Interrupciones en servicios externos	2	4	8
4.3. Costos de infraestructura	1	4	4
4.4. Legislación y regulaciones de datos	4	4	16

Tabla 23: Riesgos

Matriz de riesgos

A continuación se detalla la matriz de riesgos correspondientes. Los colores denotan la severidad de los mismos, aquellos de color rojo corresponden a una severidad alta, los amarillos a una severidad media y finalmente las secciones verdes denotan una severidad

baja. Esta matriz también debió modificarse debido al cambio en el rango de valores asignables para ilustrar la probabilidad y el impacto de los riesgos.

		Probabilidad				
		1 (Rara)	2 (Improbable)	3 (Posible)	4 (Probable)	5 (Frecuente)
Impacto	1 (Insignificante)	1	2	3	4	5
	2 (Menor)	2	4	6	8	10
	3 (Moderado)	3	6	9	12	15
	4 (Significativo)	4	8	12	16	20
	5 (Crítico)	5	10	15	20	25

Ilustración 73: Matriz de riesgos

Plan de Mitigación y Acción

En esta revisión, se evaluaron las acciones de mitigación y los planes de acción definidos para cada uno de los riesgos identificados. Concluimos que dichas acciones siguen siendo adecuadas y proporcionales al nivel del riesgo asociado, abarcando de manera integral las posibles contingencias. Por este motivo, no se consideraron necesarias modificaciones, y se decidió mantener las mismas estrategias.

Riesgos críticos (severidad > 16)

Clasificamos un riesgo como crítico en caso de que su severidad se encuentre entre 16-25.

Estos riesgos requieren atención inmediata, un control activo frecuente para verificar su ocurrencia y acciones correctivas urgentes en caso de que ocurran.

Dentro de esta clasificación, actualmente se poseen los siguientes riesgos:

4.4. Legislación y regulaciones de datos

- Mitigación
 - Identificar las regulaciones aplicables y asegurarse de que el proyecto esté alineado con las mismas.
 - Implementar medidas de seguridad como el cifrado de datos para proteger la información sensible.
 - Asegurar que solo personal autorizado tenga acceso a los datos sensibles.
 - Implementar una política de privacidad clara para los usuarios.

- Recopilar la información mínima necesaria para el funcionamiento del sistema.
- Implementar un proceso de consentimiento explícito de los usuarios antes de recopilar cualquier dato personal o sensible, especificando el uso de los datos en términos claros.
- Plan de acción
 - Si se detecta una violación de datos, notificar de inmediato a las autoridades regulatorias y, en caso de que sea necesario, a los usuarios afectados, según las leyes aplicables.
 - Evaluar el impacto de la violación para determinar la magnitud y las posibles consecuencias para los usuarios y la organización.
 - Tomar medidas correctivas para resolver la violación, como corregir cualquier vulnerabilidad en el sistema, actualizar protocolos de seguridad o revisar el acceso a los datos personales.
 - Informar a los usuarios afectados por la violación sobre qué datos fueron comprometidos, cómo están siendo protegidos, y qué medidas pueden tomar para protegerse (como cambiar contraseñas).
 - Proporcionar a las autoridades reguladoras un informe detallado sobre la violación, las medidas correctivas tomadas y las acciones que se implementarán para prevenir futuros incidentes.

Riesgos moderados (6 < severidad < 15)

Clasificamos un riesgo como moderado en caso de que su severidad se encuentre entre 6-15.

Estos riesgos son considerados como gestionables y requieren evaluación recurrente y posibles medidas de mitigación, en caso de que ocurran.

Dentro de esta clasificación, actualmente se poseen los siguientes riesgos:

1.1. Falta de conocimiento en LLM

- Mitigación
 - Establecer un backlog específico para el aprendizaje del LLM: tareas específicas que cubran investigación y experimentos.
 - Realizar spikes para investigar tecnologías relevantes y reducir la incertidumbre técnica.
 - Probar el LLM como un MVP para poder identificar limitaciones antes de entrar en el desarrollo completo.
 - Establecer métricas de evaluación claras
- Plan de acción
 - Consultar a expertos externos o comunidades técnicas si surgen problemas graves.

Este riesgo bajó su severidad ya que la probabilidad de ocurrencia disminuyó ya que realizamos las actividades de investigación pertinentes y el equipo logró adquirir un conocimiento sobre la temática. Esto se debió a las actividades de investigación y spikes realizados sobre dicha temática.

1.3. Escalabilidad de la solución

- Mitigación
 - Asegurarse de que la arquitectura del sistema sea modular y flexible, permitiendo una expansión futura sin necesidad de una reestructuración completa. Esto incluye separar los componentes críticos, como la base de datos, el backend y el LLM, para que puedan escalar de manera independiente si es necesario.

- Asegurarse de que las consultas a la base de datos estén bien optimizadas y que se utilicen índices para mejorar el rendimiento con un gran número de registros.
 - Realizar algunas pruebas de carga con una cantidad controlada de usuarios simulados, para detectar cuellos de botella.
 - Configurar servicios con auto-scaling.
 - Implementar herramientas de monitoreo para controlar el rendimiento del sistema en tiempo real.
- Plan de acción
 - Monitoreo proactivo para identificar rápidamente qué partes del sistema están experimentando sobrecarga.
 - Revisar los logs para identificar cuellos de botella.
 - Añadir nuevos servidores o instancias para distribuir la carga de trabajo.
 - Realizar una evaluación del sistema para ajustar la arquitectura y procesos según las necesidades de crecimiento.
 - Continuar realizando pruebas de carga y realizar ajustes basados en la demanda real.

Este riesgo aumentó significativamente su probabilidad de ocurrencia ya que en esta etapa del proyecto ya se comenzó a planificar la liberación de un MVP, para ser testeado tanto por el cliente, como por posibles usuarios finales.

1.4. Errores en el despliegue

- Mitigación
 - Automatización del despliegue: Configurar un pipeline de CI/CD para minimizar errores humanos y garantizar que el proceso de despliegue sea repetible y consistente.
 - Usar contenedores (como Docker) para empaquetar tanto el *frontend* como el *backend*, asegurando que el entorno de desarrollo sea idéntico al de producción.
 - Usar sistemas de control de versiones para gestionar los cambios en el código.
 - Documentación y procedimientos claros.

- Monitoreo y alertas sobre problemas en el despliegue.
- Plan de acción
 - Analizar los logs de despliegue para identificar el error específico.
 - Rollback a una versión estable, si presenta problemas críticos.
 - Revisar y corregir la configuración.
 - Revisión de dependencias.
 - Revisión de permisos y configuraciones de infraestructura.
 - Revisión post-mortem y ajustes en el proceso de infraestructura, configuración y pruebas.

2.1. Retrasos en entregas

- Mitigación
 - Utilizar metodologías ágiles para dividir el trabajo en tareas pequeñas y bien definidas.
 - Establecer metas claras y alcanzables para cada sprint y asignar responsabilidades específicas a cada miembro del equipo.
 - Realizar reuniones de planificación regulares.
 - Definir un cronograma con hitos y etapas.
- Plan de acción
 - En cuanto se identifiquen posibles retrasos, analizar la causa raíz para poder tomar decisiones y ajustar el plan.
 - Si hay retrasos importantes, reajustar las prioridades.
 - Si los retrasos afectan las entregas o plazos, se debe comunicar de inmediato la situación, proporcionando una nueva estimación de tiempo y explicando las razones del retraso.
 - Hacer una revisión de las metas del sprint y ajustarlas para que sean más realistas, sin comprometer la calidad.
 - Realizar un plan de contingencia que contemple un margen de tiempo adicional en los sprints posteriores.

2.2. Desacuerdos con el PO

- Mitigación
 - Definición clara de los requerimientos del proyecto.
 - Establecer un proceso formal para el cambio de requerimientos: cualquier cambio en los requisitos debe ser documentado y comunicado claramente.

- Usar herramientas de gestión de proyectos para registrar y hacer seguimiento de los cambios en los requerimientos.
- Realizar reuniones regulares con el PO para revisar y ajustar las prioridades del proyecto.
- Mantener una comunicación constante y efectiva con el PO, asegurándose de que las preguntas y comentarios se resuelvan rápidamente.
- Plan de acción
 - Organizar una reunión con el PO para revisar los cambios y asegurarse de que las prioridades sean claras.
 - Aclarar los objetivos a corto y largo plazo, y cómo los cambios pueden afectar el cronograma, alcance y recursos del proyecto.
 - Documentar los acuerdos alcanzados en cuanto a los cambios de los requerimientos.

3.1. Disponibilidad del equipo

- Mitigación
 - Establecer horarios de trabajo que sean visibles para todo el equipo.
 - Realizar reuniones de sincronización periódicas
 - Avisar inasistencias.
- Plan de acción
 - Si un miembro del equipo está ausente por un tiempo prolongado (vacaciones, enfermedad, etc.), asignar tareas críticas a otros miembros del equipo de manera provisional hasta su regreso
 - Realizar una revisión periódica de la carga de trabajo de cada miembro del equipo para identificar posibles sobrecargas y redistribuir tareas en caso necesario.
 - Fomentar una cultura de transparencia, en la que los miembros del equipo se sientan cómodos al comunicar su disponibilidad

4.2. Interrupciones en servicios externos

- Mitigación
 - Realizar un análisis exhaustivo de los proveedores de servicios externos, asegurando que sean confiables, con un historial de alta disponibilidad y que ofrezcan acuerdos de nivel de servicios claro que especifiquen tiempos de inactividad aceptables.

- Tener configuraciones de alta disponibilidad, en los servidores dónde se realice el despliegue.
- Configurar alertas y monitoreo en tiempo real para los servicios externos críticos. Nos permitirá identificar caídas o problemas en servicios de manera temprana y actuar antes de que afecten al usuario.
- Plan de acción
 - Definir un procedimiento detallado para gestionar interrupciones en los servicios. Esto debe incluir un listado de pasos a seguir, desde la identificación del fallo hasta la resolución y la comunicación con los proveedores.
 - Si un servicio externo se interrumpe, tener una alternativa funcional, como un servicio de reemplazo temporal o un mecanismo local de cacheo para mitigar el impacto de la caída del servicio.
 - Configurar los servidores y servicios en la nube con escalabilidad automática. Esto permite que el sistema se ajuste a los picos de demanda y pueda sobrellevar interrupciones o caídas de servicios sin afectar la disponibilidad del proyecto.
 - Después de una interrupción, realizar una revisión exhaustiva del incidente para entender las causas, evaluar la efectividad de las medidas tomadas y determinar cómo mejorar el sistema para prevenir futuras caídas.
 - Hacer simulaciones de incidentes para asegurar que todos los miembros del equipo sepan cómo actuar durante una interrupción de servicio.

Riesgos bajos (severidad < 5)

Clasificamos un riesgo como bajo en caso de que su severidad se encuentre entre 0-5.

Estos riesgos son considerados como riesgos que deben ser monitoreados y poseen medidas de mitigación leves.

Dentro de esta clasificación, actualmente se poseen los siguientes riesgos:

2.3. Falta de comunicación

- Mitigación
 - Establecer canales de comunicación centralizados.
 - Documentar las decisiones tomadas y discutir las en las reuniones de sprint.
 - Fomentar la comunicación abierta y las preguntas.
 - Utilizar herramientas de gestión de proyectos
- Plan de acción
 - Si ocurre un incidente debido a falta de comunicación, organizar una reunión con las personas involucradas para poder obtener toda la información y rectificar el error.
 - Utilizar correctamente los canales de comunicación preestablecidos.
 - Designar a un facilitador o responsable que asegure la fluidez de la comunicación en temas críticos.

Este riesgo disminuyó su grado de severidad ya que el equipo logra mantener una comunicación fluida entre los integrantes, mediante diversos canales de comunicación, así como reuniones recurrentes entre los miembros. Se establecieron nuevos canales de comunicación, como Slack, para lograr mantener hilos de conversación específicos para los diversos temas que puedan surgir. Esta herramienta permitió una mejor organización de la comunicación entre los miembros.

2.4. Falta de claridad en los roles

- Mitigación
 - Documentación centralizada con descripciones de tareas y roles claras.
 - Reuniones de alineación.
- Plan de acción
 - Ante dudas estar abiertos a revisiones y dar aclaraciones en equipo de las tareas a completar y roles que cumplir.
 - Redefinir / actualizar los criterios para que estén al día con las necesidades.
 - Incorporar reuniones de feedback periódicas para identificar confusiones relacionadas con roles y mejorar la definición de responsabilidades.

3.2. Desbalance en la carga de trabajo

- Mitigación
 - Establecer claramente roles y responsabilidades desde el inicio, con revisiones periódicas de la distribución de tareas.
 - Dividir las tareas del proyecto, y asignarlas de acuerdo a las habilidades de cada miembro del equipo.
 - Utilizar una herramienta de gestión de proyectos para asignar y priorizar tareas de forma clara.
 - Evaluar semanalmente la carga de trabajo de cada integrante del equipo en las reuniones recurrentes.
 - Trabajar en sprints, lo que permite realizar ajustes rápidos en la asignación de tareas.
 - Utilizar un Product Owner interno para gestionar y redistribuir tareas de forma eficiente.
- Plan de acción
 - Identificar rápidamente la sobrecarga, y redistribuir tareas a otros miembros del equipo de acuerdo con sus habilidades y disponibilidad.
 - Priorizar las tareas más críticas del proyecto, asegurando que las áreas claves sean atendidas primero.

4.1. Cambios en tecnologías usadas

- Mitigación
 - Mantenerse al tanto de las actualizaciones, parches y nuevas versiones de las tecnologías que utilizamos.
 - Uso de versiones específicas: usar herramientas como Docker para gestionar entornos controlados.
 - Usar entornos virtuales para trabajar, de modo que se puedan controlar las dependencias y versiones sin afectar a otros componentes del sistema.
 - Mantener una documentación detallada sobre las versiones y configuraciones actuales del LLM y las herramientas asociadas.
- Plan de acción
 - Informarse inmediatamente sobre los detalles de la actualización y analizar su impacto potencial en el proyecto.
 - Determinar si la actualización es crítica y necesita ser implementada inmediatamente, o si se puede esperar a una futura versión más estable.
 - Si hay incompatibilidades, desarrollar un parche que permita que el sistema siga funcionando correctamente.

4.3. Costos de infraestructura

- Mitigación
 - Configurar alertas de costos con proveedores de servicios en la nube.
 - Revisar periódicamente los recursos utilizados para identificar posibles sobras o recursos infrautilizados que se puedan ajustar o eliminar.
 - Implementar auto-scaling en los servicios de *backend* y en la infraestructura.
 - Optimizar consultas a bases de datos y almacenamiento en la nube.
 - Revisión periódica de la infraestructura.
- Plan de acción
 - Identificar el servicio o recurso que está causando el aumento inesperado de costos.
 - Auditar los recursos consumidos para verificar si hay recursos infrautilizados.

- Verificar y ajustar las configuraciones de auto-scaling para evitar que los sistemas escalen innecesariamente.
- Identificar recursos inactivos o infrautilizados y eliminarlos o reducir su tamaño.
- Análisis post-mortem y ajustes a largo plazo.

Evaluación de Riesgos - Abril

En la revisión de riesgos correspondiente al mes de abril, profundizamos nuestro análisis incorporando tanto los riesgos previamente identificados como algunos nuevos que, en ciertos casos, ocurrieron durante este periodo de desarrollo. Entre las situaciones detectadas destaca la pérdida de acceso a tecnologías clave, que nos dejó sin acceso al correo electrónico asociado a todo el proyecto, incluyendo el modelo y otras herramientas utilizadas en este proyecto. Además, ocurrió que realizamos muchas consultas con la misma clave por lo que el modelo utilizado, en este caso Chat GPT, bloqueó la misma. Esto imposibilitó al equipo de realizar más consultas con la misma, por lo tanto se consideró como un riesgo a tener en cuenta.

Durante la revisión, también volvimos a examinar la escala de evaluación seleccionada en la revisión pasada. Esta nueva escala demostró ser muy útil y, en consecuencia, la hemos mantenido para esta revisión.

A continuación se detallan los nuevos riesgos incorporados en abril, su valoración según la escala ampliada y las acciones de mitigación que implementaremos de manera proactiva. Los riesgos ya existentes conservan las estrategias y planes de acción previamente establecidos.

Nuevos Riesgos Identificados

En esta revisión, detectamos tres riesgos tecnológicos que no habían sido evaluados previamente y que, de materializarse, pueden impactar de forma significativa la continuidad y calidad del proyecto. En el caso del riesgo 1.5, él mismo ocurrió en este último período.

1.4.1 Falta de conocimiento sobre las herramientas de AWS: Riesgo de incurrir en errores operativos o de configuración, así como retrasos en los plazos de entrega por curvas de aprendizaje elevados en las herramientas.

1.5 Pérdida de acceso a tecnologías: Riesgo de quedar sin acceso a herramientas, cuentas o servicios críticos (correo electrónico, repositorios) que soportan el desarrollo de este proyecto.

1.5.1. Control de acceso a modelo de IA: Riesgo de no disponer de un mecanismo seguro y autónomo para gestionar credenciales o claves de API que permitan operar el modelo de IA sin depender únicamente de una sola API Key para poder realizar consultas.

Planes de mitigación y acción frente a estos nuevos riesgos

Siguiendo la matriz de riesgos establecida en la revisión pasada, los riesgos 1.4.1 y 1.5 fueron clasificados como riesgos con severidad media, mientras que el riesgo 1.5.1 es considerado con severidad baja. Teniendo esto en cuenta se generaron los siguientes planes de mitigación y acción para cada uno de ellos.

1.4.1. Falta de conocimiento sobre herramientas de AWS

- Mitigación
 - Organizar sesiones de capacitación en el equipo focalizadas en los servicios de AWS que serán clave para este proyecto.
- Plan de acción
 - Generar una guía de despliegue en dónde se especifique los pasos a seguir para desplegar los componentes en las herramientas de AWS.
 - Generar una guía en dónde se detallen las herramientas de AWS que se utilizan en este proyecto y qué funcionalidad cumplen para el mismo.
 - Incorporar revisiones mensuales de configuraciones de AWS para corroborar que todo esté funcionando correctamente.

1.5 Pérdida de acceso a tecnologías

- Mitigación
 - Configurar cuentas secundarias y de respaldo en todos los servicios críticos.
 - Centralizar credenciales en un gestor de contraseñas.

- Plan de acción
 - Ingresar recurrentemente a la cuenta de correo electrónico, para que se generen logs dentro de dicha cuenta y no sea considerada dudosa por parte de Google.

1.5.1 Control de acceso a modelo de IA

- Mitigación
 - Evaluar uso de mecanismos de rotación automática de credenciales o autenticación federales.
 - Investigar sobre flujo de autenticación alternativo que no dependa de una sola API Key estática
- Plan de acción
 - Investigar y seleccionar una solución de gestión de secretos.
 - Implementar un script automatizado que rote la clave de la API y actualice la configuración del cliente

Con estos nuevos riesgos identificados, se procedió a revisar los valores establecidos de probabilidad de ocurrencia, impacto y severidad de cada uno de los riesgos identificados para este proyecto.

Riesgos	Probabilidad	Impacto	Severidad
1.1 Falta de conocimiento en LLM	3	4	12
1.2 Problemas de compatibilidad	0	2	0
1.3 Escalabilidad de la solución	4	5	20
1.4 Errores en el despliegue	3	3	9
1.4.1 Falta de conocimiento sobre herramientas de AWS	3	3	9
1.5 Pérdida de acceso a tecnologías	2	5	10
1.5.1 Control de acceso a modelo de IA	1	5	5

2.1 Retrasos en entregas	2	3	6
2.2 Desacuerdos con el PO	3	4	12
2.3 Falta de comunicación	3	2	6
2.4 Falta de claridad en los roles	1	2	2
3.1 Disponibilidad del equipo	2	3	6
3.2 Desbalance en la carga de trabajo	1	2	2
4.1 Cambios en tecnologías usadas	1	3	3
4.2 Interrupciones en servicios externos	2	5	10
4.3 Costos de infraestructura	3	4	12
4.4 Legislación y regulaciones de datos	4	4	16

Tabla 24: Riesgos

Evolución de la severidad de los riesgos

Hasta la revisión pasada, estuvimos graficando los riesgos en un único punto en el tiempo debido a las iteraciones en las revisiones de los mismos realizadas. A partir de esta revisión, contamos ya con datos de varios meses que nos permiten observar tendencias y comportamientos dinámicos de cada riesgo. Por primera vez en este reporte, incorporamos el siguiente gráfico de líneas que ilustra la severidad (Probabilidad * Impacto) de cada riesgo identificado desde febrero hasta abril.

En el gráfico, cada línea representa un riesgo identificado y especificado por su número identificador que se ha mantenido desde un principio. Dicho gráfico es acumulativo y servirá como herramienta para monitorear de forma más sencilla aquellos riesgos con mayor severidad y que podrían afectar en mayor medida este proyecto.

A continuación, el resultado visual generado:

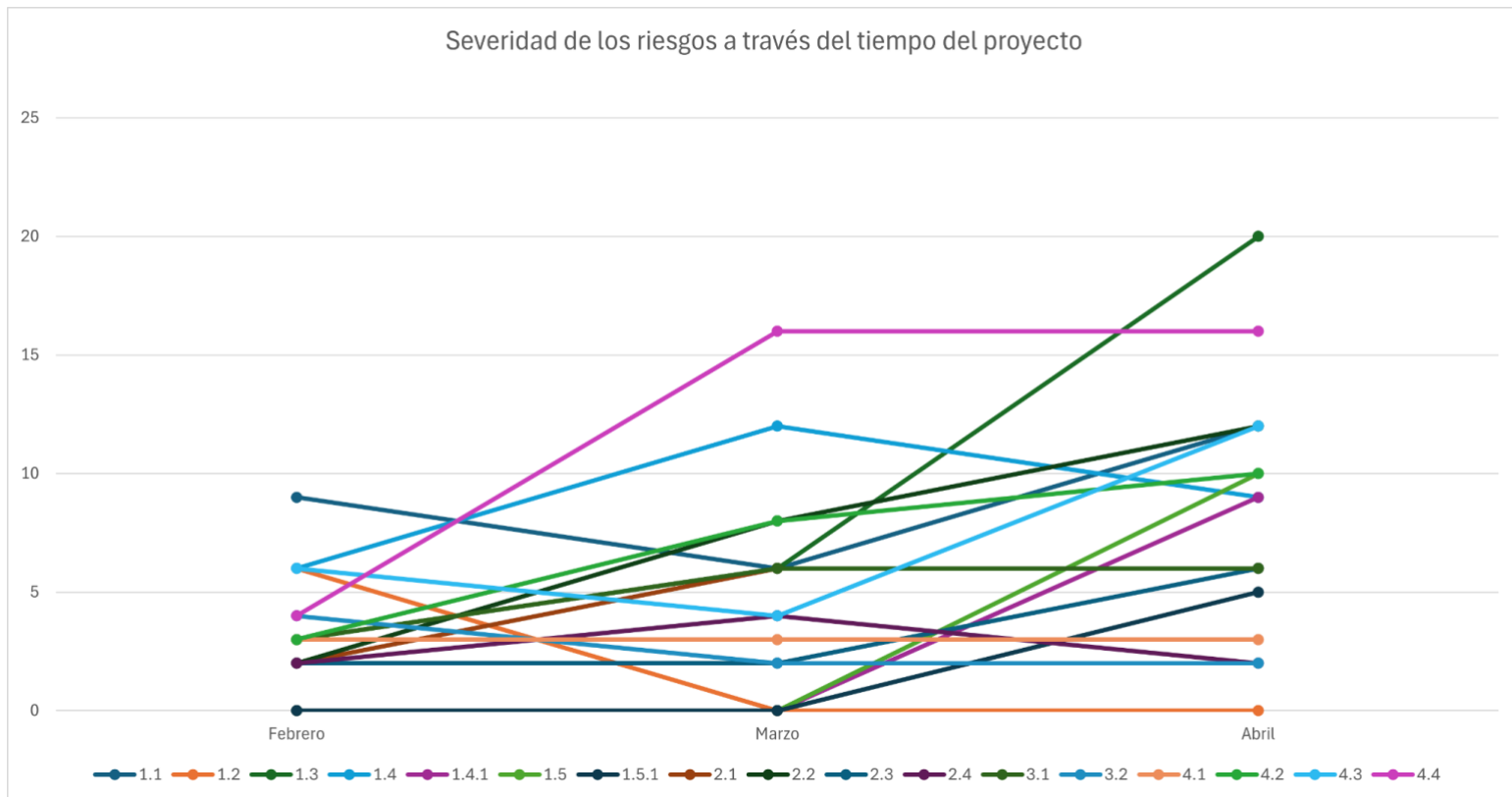


Ilustración 74: Severidad de los riesgos a través del tiempo del proyecto.

Evolución de riesgos - Mayo

En esta oportunidad, se realizó el relevamiento de los riesgos identificados y se consideró que la valoración de la probabilidad, impacto y severidad de los mismos no se vió modificado con las actividades que se realizaron durante dicho mes. Por lo tanto se mantuvieron los valores de los mismos.

Evolución de riesgos - Junio

En la revisión de riesgos correspondiente al mes de junio, se profundizó el análisis de los riesgos previamente identificados, modificando la probabilidad e impacto de alguno de ellos. Durante este mes, no se identificaron nuevos riesgos.

Modificaciones en las valoraciones de los riesgos

Se destaca en algunos casos la modificación en los valores de probabilidad, impacto y severidad para ciertos riesgos identificados. En el caso del riesgo identificado como 1.1 Falta de conocimiento en LLM, se disminuyó considerablemente la probabilidad de que

el mismo ocurra ya que el equipo logró aumentar considerablemente su conocimiento sobre el tema.

En el caso del riesgo 1.3 Escalabilidad de la solución, se disminuyó su valor de probabilidad ya que durante este período se desarrolló concretamente el plan de infraestructura para el proyecto para lograr el despliegue del mismo. El mismo se ideó para que cada uno de los servicios utilizados en el despliegue pueda escalar automáticamente dependiendo de la cantidad de flujo que reciben. Similarmente, el riesgo 1.4 Errores en el despliegue aumentó su probabilidad en comparación con los análisis anteriores ya que él mismo ocurrió y se aplicaron las medidas de mitigación especificadas en un principio para el mismo. Se automatizó el proceso para que estos errores no ocurran nuevamente y se puso énfasis en aquellos errores identificados. Con respecto al riesgo 1.4.1 Falta de conocimiento sobre herramientas de AWS, se disminuyó la probabilidad de ocurrencia ya que dos miembros del equipo cursaron una materia dentro de la facultad en dónde se profundizó sobre esta temática, allí pusieron en práctica los conocimientos adquiridos en los obligatorios de dicha materia. Esto permitió que dichos miembros compartieran el conocimiento adquirido con los demás miembros del equipo y replicaron dichos conocimientos para desarrollar el plan de despliegue e infraestructura para este proyecto.

Con respecto al riesgo 1.5.1 Control de acceso a modelo de IA, se aumentó su probabilidad de ocurrencia ya que el error reportado sobre el uso de las API Key del modelo consultado volvieron a ocurrir sobre este período de análisis. Se decidió poner mayor énfasis en la búsqueda de un flujo de autenticación alternativo que no dependa de una sola API Key estática.

En el caso del riesgo 2.4 Falta de claridad en los roles, se disminuyó su probabilidad a cero ya que debido a la etapa del proyecto en la que se encontraba el equipo se tomó la decisión de abandonar los roles estructurados definidos al comienzo del proyecto y adoptar una conducta de igualdad entre los mismos. Esto se debió a que el equipo consideró que este enfoque podría ser beneficioso en esta etapa intermedia del proyecto, dónde el foco principal fue la implementación de funcionalidades así como la evolución del modelo para mejorar la precisión de las respuestas generadas.

Lo mismo ocurrió con el riesgo 4.4 Legislación y regulación de los datos. Su probabilidad de ocurrencia disminuyó.

A continuación se detallan los valores de probabilidad, impacto y severidad identificados para cada uno de los riesgos:

Riesgos	Probabilidad	Impacto	Severidad
1.1 Falta de conocimiento en LLM	1	4	4
1.2 Problemas de compatibilidad	0	2	0
1.3 Escalabilidad de la solución	2	5	10
1.4 Errores en el despliegue	4	3	12
1.4.1 Falta de conocimiento sobre herramientas de AWS	2	4	8
1.5 Pérdida de acceso a tecnologías	1	5	5
1.5.1 Control de acceso a modelo de IA	2	5	10
2.1 Retrasos en entregas	3	3	9
2.2 Desacuerdos con el PO	2	4	8
2.3 Falta de comunicación	2	3	6
2.4 Falta de claridad en los roles	0	2	0
3.1 Disponibilidad del equipo	2	3	12
3.2 Desbalance en la carga de trabajo	1	2	2
4.1 Cambios en tecnologías usadas	1	3	3

4.2 Interrupciones en servicios externos	3	5	15
4.3 Costos de infraestructura	3	4	12
4.4 Legislación y regulaciones de datos	2	4	8

Tabla 25: Riesgos

Evolución de la severidad de los riesgos

Se destaca la evolución de los riesgos, el incremento en la severidad de algunos de ellos.

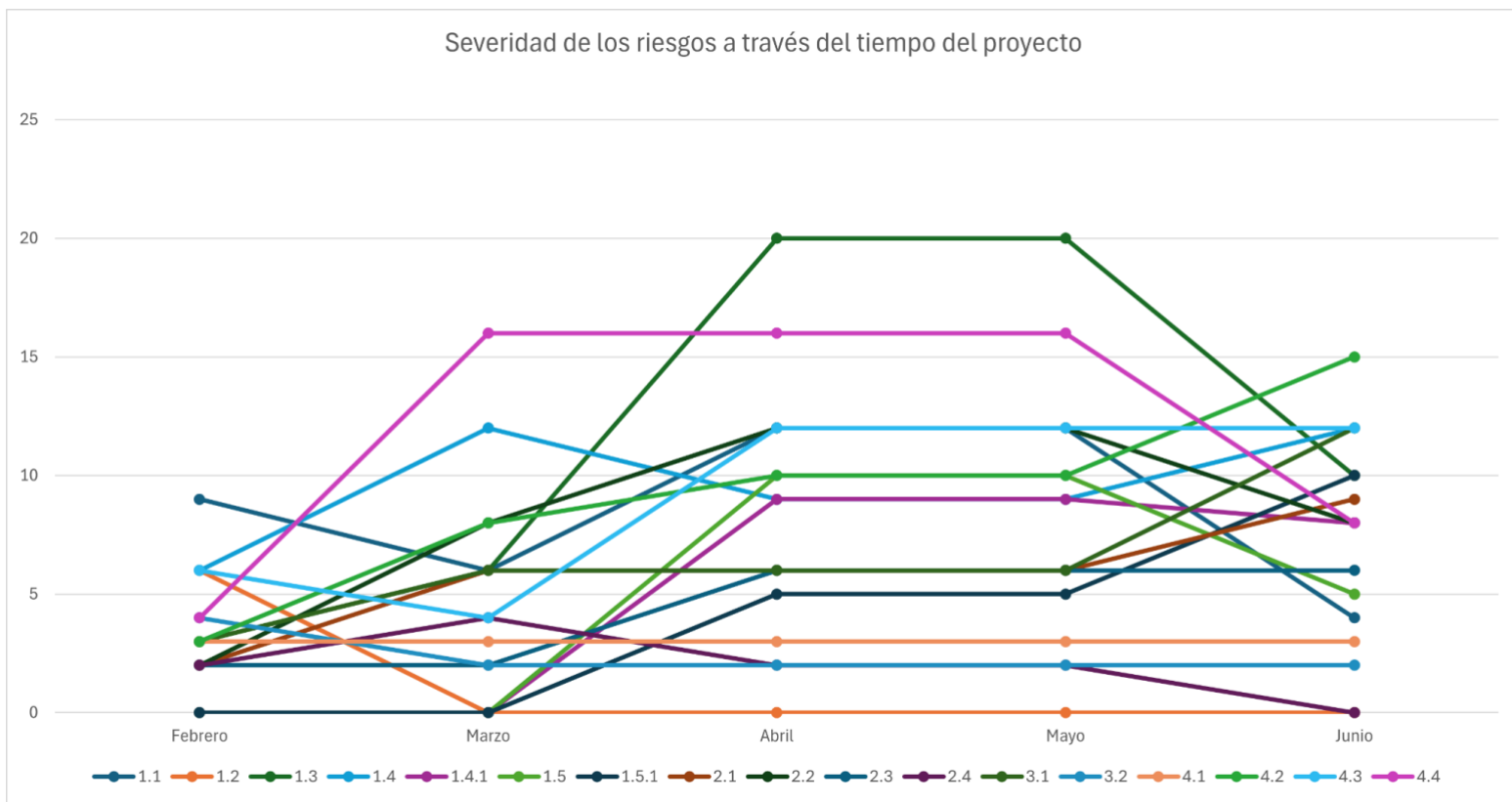


Ilustración 75: Severidad de los riesgos a través del tiempo

Evolución de riesgos - Agosto

Durante esta revisión, identificamos un nuevo riesgo ocurrido durante dicho período de evaluación, la desalineación con los objetivos del cliente. El problema surgió cuando, siguiendo la recomendación del tutor, modificamos la interfaz de usuario. Sin embargo, en la revisión con el cliente, se hizo evidente que su prioridad principal no era la UI, sino la funcionalidad del modelo LLM y las configuraciones probadas.

Este nuevo riesgo resalta la importancia de validar las prioridades con el cliente para asegurar que el desarrollo se centre en lo que realmente aporta valor al proyecto.

Planes de mitigación y acción frente a este nuevo riesgo

Siguiendo la matriz de riesgos establecida previamente, el riesgo 2.5 Desalineación con los objetivos del cliente fue calificado con una severidad alta. Teniendo esto en cuenta se generó el siguiente plan de mitigación y acción para el mismo.

2.5. Desalineación con los objetivos del cliente

- Mitigación
- Plan de acción
 - Planificar muestras al tutor con mayor frecuencia de las interfaces gráficas generadas para validar que las mismas cumplan con los estándares de producto esperados para este tipo de proyectos por parte de la universidad, teniendo en cuenta que el cliente no tiene un foto específico en ello.

13.10. Gestión de calidad

13.10.1. Plan de calidad

Nombre fase	Actividad	Producto Resultado	Producto/s Consumido/s	Rol Responsable	Roles Participantes	Documentos de referencia	Mecanismos
Investigación	Estudio de las tecnologías	Documento de investigación tecnológica	Necesidades del cliente, Información de sitios web	Equipo	Equipo	Estándar de documentos internos	
	Estudio de tecnologías de LLM	Documento de investigación y POCs	Información de sitios webs	Equipo	Equipo	Estándar de documentos internos	POCs
	Planificación	Documentación Iniciales, Listados de requerimientos iniciales, Entendimiento del problema		Equipo	Equipo		Reuniones con cliente y equipo
Ingeniería de requerimientos	Validación de Requerimientos	Documento de especificación de requerimientos	Listado de requerimientos iniciales	Equipo	Equipo, Usuarios finales		Encuestas y Entrevistas con usuarios
	Especificación de requerimientos	Requerimientos validados por cliente. Product Backlog	Requerimientos, actores, necesidades del cliente, necesidades de usuarios finales	Equipo	Cliente, Equipo		Reuniones con cliente y equipo
	Creación de prototipo	Prototipo en Figma	Documento de especificación de requerimientos	Equipo	Equipo		Herramientas de prototipación
	Validación del prototipo	Feedback sobre prototipo	Prototipo en Figma	Equipo	Equipo, Usuarios finales		Entrevistas con usuarios
	Definición de criterios de aceptación	Criterios de aceptación definidos	Prototipo, Especificación de los requerimientos	Equipo	Equipo	Estándar de criterios de aceptación	

Diseño	Análisis de Costos	Documento de costos de arquitectura	Requerimientos de cliente	Responsable Infraestructura			
	Diseño de la arquitectura	Documento de arquitectura	Especificación de requerimientos	Equipo			
	Revisión de la arquitectura	Documento de arquitectura revisado	Documento de arquitectura	Equipo			
Desarrollo	Planificación	Backlog refinado	Product Backlog	Equipo	Equipo		Priorización, Planning Poker
	Codificación	Funcionalidad desarrollada	Product Backlog	Equipo	Equipo	Estándares de código (Prettier, Eslint)	Husky
	Code Review	Código revisado y aprobado	Funcionalidad Desarrollada	Equipo	Equipo	Estándares de código	Pull requests con 2 aprobaciones
Testing	Diseño de casos de prueba	Planilla de casos de prueba	Especificación de requerimientos	Equipo	Equipo	Plantilla de diseño de casos de prueba	Plantilla de diseño de casos de prueba
	Implementación de CI para automatizar pruebas	Ejecución automática de pruebas	Casos de pruebas, Código desarrollado	Responsable Infraestructura			GitHub Actions/CI Pipeline
	Pruebas Unitarias	Cobertura de pruebas Mayor a 90%	Código desarrollado	Equipo Producto			Jest
	Pruebas de Integración	Validación de módulos integrados	Módulos desarrollados	Equipo Producto			Jest
	Pruebas para LLM	Validación de precisión y relevancia del LLM	Modelo LLM Integrado	Equipo RAG			Evaluación manual y métricas automatizadas
	Pruebas de rendimiento	Informe de rendimiento bajo carga	Sistema integrado	Equipo Producto			Load Testing, K6
	Pruebas de Usabilidad	Validación de interfaz de usuario	Prototipo, Sistema deployado, Tareas para validación con prototipo	Equipo	Equipo, Usuarios finales		Pruebas con usuarios, Encuestas
Despliegue	Planificación	Plan de		Responsab			

		despliegue, README.md		le Infraestructura			
	Configuración del entorno AWS	Código Terraform y configuración del entorno de producción	Plan de despliegue, Documentación AWS y Vercel	Responsable Infraestructura			Código en Terraform
	Configuración y Mantenimiento del Pipeline CI/CD	Pipeline CI/CD configurado y funcional	Plan de despliegue	Responsable Infraestructura		Documentación de AWS, Guías de CI/CD	GitHub Actions, AWS CodePipeline
	Monitoreo Continuo	Sistema de monitoreo y alertas	Sistema en producción	Responsable Infraestructura	Equipo	Logs del pipeline	Herramientas de CI/CD, Notificaciones automáticas, CloudWatch, Budget Alerts
Procesos de Gestión (Independientes de la fase)	Planificación de actividades	Cronograma de actividades	Plan de la calidad	Equipo			
	Gestión de riesgos	Plan de riesgos	Especificación de requerimientos, supuestos, Documento de investigación tecnológica	Equipo		Bibliografía de manejo de riesgos	Matriz de riesgos
	Diseño del Plan de la calidad	Plan de la calidad	RNFs, Requerimientos	Equipo		Bibliografía de calidad de software	Definición de métricas y umbrales

Tabla 26: Plan de calidad

13.10.2. Casos de prueba

#	Descripción	Requerimiento Asociado	Resultado Esperado
1	Crear comunidad con datos válidos	RFT1	Creación exitosa con espacio aislado
2	Crear comunidad sin datos requeridos	RFT1	Mensaje de error
3	Cambiar modo de operación de comunidad a moderado	RFT2	Cambio exitoso, respuestas pendientes de aprobación
4	Cambiar modo de operación de comunidad a libre	RFT2	Cambio exitoso, respuestas publicadas directamente
5	Habilitar selección de mejor respuesta en comunidad moderada	RFT3	Configuración exitosa
6	Deshabilitar selección de mejor respuesta	RFT3	Configuración exitosa
7	Cargar archivo Markdown con datos válidos	RFA1	Carga exitosa, datos integrados a la base de conocimiento
8	Cargar archivo Excel con datos válidos (nomenclátor)	RFA1	Carga exitosa, datos integrados a la base de conocimiento
9	Cargar archivo con formato inválido	RFA1	Mensaje de error
10	Cargar archivo que excede tamaño máximo permitido	RFA1	Mensaje de error
11	Invitar miembro con email válido	RFA2	Invitación enviada exitosamente
12	Invitar miembro con email inválido	RFA2	Mensaje de error
13	Invitar miembro con email ya registrado en la comunidad	RFA2	Mensaje de error
14	Listar todos los miembros de la comunidad	RFA3	Listado completo de miembros
15	Desactivar cuenta de miembro existente	RFA3	Desactivación exitosa
16	Desactivar cuenta de miembro inexistente	RFA3	Mensaje de error
17	Listar preguntas sin filtros	RFA4	Listado completo de preguntas
18	Filtrar preguntas por estado "PENDIENTE"	RFA4	Listado de preguntas pendientes
19	Filtrar preguntas por etiqueta específica	RFA4	Listado de preguntas con esa etiqueta
20	Filtrar preguntas con criterios que no devuelven resultados	RFA4	Listado vacío
21	Dejar comentario en respuesta de IA con datos válidos	RFA5	Comentario registrado exitosamente
22	Dejar comentario vacío en respuesta de IA	RFA5	Mensaje de error
23	Rechazar respuesta de IA en modo moderado	RFA6	Respuesta rechazada, pregunta marcada como RECHAZADA
24	Intentar rechazar respuesta de IA en modo libre	RFA6	Mensaje de error o acción no permitida
25	Aceptar respuesta de IA en modo moderado	RFA6	Respuesta aceptada y visible

26	Realizar búsqueda semántica con texto válido	RFA7	Resultados ordenados por relevancia
27	Realizar búsqueda semántica con texto sin resultados	RFA7	Listado vacío
28	Crear etiqueta nueva con datos válidos	RFA8	Alta exitosa
29	Crear etiqueta con nombre duplicado	RFA8	Mensaje de error
30	Editar etiqueta existente con datos válidos	RFA8	Edición exitosa
31	Eliminar etiqueta no utilizada	RFA8	Eliminación exitosa
32	Eliminar etiqueta en uso	RFA8	Mensaje de advertencia o error
33	Visualizar dashboard de estadísticas	RFA9	Dashboard con estadísticas actualizadas
34	Configurar comunidad con parámetros válidos	RFA10	Configuración actualizada exitosamente
35	Configurar comunidad con parámetros inválidos	RFA10	Mensaje de error
36	Hacer pregunta con texto válido en modo libre	RFU1	Pregunta creada, respuesta de IA generada
37	Hacer pregunta con texto válido en modo moderado	RFU1	Pregunta creada, respuesta pendiente de aprobación
38	Hacer pregunta sin texto	RFU1	Mensaje de error
39	Hacer pregunta que excede límite de caracteres	RFU1	Mensaje de error
40	Visualizar respuesta de pregunta existente	RFU2	Contenido de respuesta mostrado correctamente
41	Visualizar respuesta de pregunta inexistente	RFU2	Mensaje de error
42	Visualizar feed de preguntas recientes	RFU2	Listado de preguntas recientes
43	Visualizar feed cuando no hay preguntas	RFU2	Mensaje indicando que no hay preguntas
44	Filtrar preguntas por etiqueta en feed	RFU3	Listado filtrado de preguntas
45	Filtrar preguntas por estado "RECHAZADA"	RFU3	Listado de preguntas rechazadas
46	Filtrar preguntas con criterios múltiples	RFU3	Listado que cumple todos los criterios
47	Agregar etiqueta válida a pregunta propia	RFU3	Etiqueta agregada exitosamente
48	Agregar etiqueta a pregunta de otro usuario sin permisos	RFU3	Mensaje de error
49	Agregar respuesta de usuario con texto válido	RFU5	Respuesta agregada exitosamente
50	Agregar respuesta de usuario sin texto	RFU5	Mensaje de error
51	Agregar respuesta de usuario que excede límite de caracteres	RFU5	Mensaje de error
52	Seleccionar mejor respuesta en modo libre	RFU6	Respuesta marcada como mejor

53	Seleccionar mejor respuesta aceptada en modo moderado	RFU6	Respuesta marcada como mejor
54	Intentar seleccionar respuesta pendiente como mejor en modo moderado	RFU6	Mensaje de error
55	Cambiar selección de mejor respuesta	RFU6	Nueva respuesta marcada como mejor
56	Votar respuesta positivamente (upvote)	RFU7	Voto registrado correctamente
57	Votar respuesta negativamente (downvote)	RFU7	Voto registrado correctamente
58	Cambiar voto de positivo a negativo	RFU7	Voto actualizado correctamente
59	Votar múltiples veces la misma respuesta	RFU7	Mensaje de error o voto único mantenido
60	Aceptar invitación con link válido	RFU8	Registro completado exitosamente
61	Aceptar invitación con link vencido	RFU8	Mensaje de error
62	Aceptar invitación con link inválido	RFU8	Mensaje de error
63	Iniciar sesión con email y contraseña válidos	RFU9	Log in exitoso
64	Iniciar sesión con email válido y contraseña incorrecta	RFU9	Mensaje de error
65	Iniciar sesión con email no registrado	RFU9	Mensaje de error
66	Iniciar sesión con cuenta desactivada	RFU9	Mensaje de error
67	Visualizar perfil propio	RFU10	Información del perfil mostrada
68	Editar perfil con datos válidos	RFU10	Edición exitosa
69	Editar perfil con datos inválidos	RFU10	Mensaje de error
70	Editar perfil con email ya existente	RFU10	Mensaje de error
71	Seleccionar comunidad de listado disponible	RFU11	Autenticación en comunidad exitosa
72	Intentar acceder a comunidad sin membresía	RFU11	Mensaje de error
73	Guardar pregunta de interés	RFB12	Pregunta guardada exitosamente
74	Guardar pregunta ya guardada previamente	RFB12	Mensaje indicando que ya está guardada
75	Remover pregunta de guardadas	RFB12	Pregunta removida exitosamente
76	Buscar preguntas con palabra clave válida	RFB13	Resultados de búsqueda relevantes

77	Buscar preguntas sin resultados	RFB13	Mensaje indicando que no hay resultados
78	Buscar preguntas con caracteres especiales	RFB13	Búsqueda procesada correctamente
79	Listar todas las preguntas del usuario	RFB14	Listado completo de preguntas propias
80	Listar preguntas de usuario sin preguntas realizadas	RFB14	Mensaje indicando que no hay preguntas

Tabla 27: Casos de prueba

13.10.3. Script de Tareas para validacion con usuarios

Fichas de tareas – Validación de Producto

Buenos días/tardes, mi nombre es [NOMBRE] y en primer lugar quiero agradecerle por haber aceptado nuestra invitación para participar en esta prueba de usabilidad.

Antes de comenzar quiero comentarle que esto no es un examen, ni una prueba de sus habilidades, sino solamente una evaluación de la facilidad de uso de nuestra plataforma. Nosotros sabemos que esta aplicación tiene aspectos a mejorar, por lo que es natural que usted al usarla se tope con ellos, lo que puede generarle dificultades en la navegación. Al contrario de lo que podría parecer, eso no solo no es inconveniente, sino que es exactamente lo que buscamos - encontrar los puntos débiles para poder corregirlos.

Con respecto a la prueba, es muy sencilla. Le vamos a proponer algunas tareas que usted realizará tal como haría normalmente, con una sola diferencia: le pediremos que nos cuente en voz alta qué está pensando y por qué hace cada cosa. Esto nos ayuda a entender su proceso de pensamiento.

La sesión durará aproximadamente 20-30 minutos y vamos a probar la versión real de la plataforma que estamos desarrollando.

¿Alguna pregunta antes de comenzar?

Preguntas previas:

Antes de comenzar con las tareas, me gustaría hacerle algunas preguntas breves:

¿Has usado alguna vez plataformas como StackOverflow, Reddit o foros de preguntas y respuestas?

¿Qué tipo de información sueles buscar sobre el cuidado de bonsáis? ¿Dónde la buscas normalmente?

¿Confiarías en respuestas generadas por una IA para temas relacionados con bonsáis?
¿Por qué?

¿Has participado alguna vez en comunidades online especializadas?

Tarea 0: Iniciar sesión

Objetivo: Observar si el usuario entiende el proceso de login y la navegación inicial.

Terminada si: El usuario logra iniciar sesión con las credenciales proporcionadas.

Instrucciones: "Imagina que ya tienes una cuenta en la plataforma. Aquí tienes tus credenciales [proporcionar email y contraseña]. Inicia sesión."

Tarea 1: Seleccionar comunidad/tenant

Objetivo: Evaluar si el usuario comprende el concepto de comunidades y puede seleccionar una.

Terminada si: Selecciona una comunidad y accede al feed principal.

Instrucciones: "Después de iniciar sesión, debes elegir una comunidad donde participar. Selecciona la que te parezca más interesante."

Tarea 2: Hacer una pregunta

Objetivo: Observar el flujo de creación de una pregunta y la claridad del etiquetado.

Terminada si: El usuario logra hacer la pregunta y la visualiza publicada (confirmación visible).

Instrucciones: Imagina que eres un usuario nuevo y tenés una duda sobre bonsáis (por ejemplo, "¿Cada cuánto debo regar un bonsái ficus en verano?"). Intenta publicarla en la plataforma.

Tarea 3: Ver mis preguntas

Objetivo: Observar si el usuario encuentra fácilmente dónde ver sus preguntas realizadas.

Terminada si: Encuentra y abre la pregunta publicada en la Tarea 1.

Instrucciones: Ahora intentá ver todas las preguntas que hiciste previamente.

Tarea 4: Modificar el perfil

Objetivo: Observar si el usuario entiende y completa correctamente el cambio de datos del perfil.

Terminada si: Cambia el nombre y guarda los cambios.

Instrucciones: Imagina que querés cambiar tu nombre y correo en la cuenta. Intentalo ahora.

Tarea 5: Explorar respuestas y comentarios de una pregunta

Objetivo: Evaluar la navegación en las respuestas, comprensión de etiquetas y sistema de votación.

Terminada si: Encuentra los comentarios y entiende la diferencia entre tipos de respuestas.

Instrucciones: Imagina que estás en la página de inicio, encontrás una pregunta interesante y querés leer las respuestas y comentarios. Navegá hasta encontrarlos.

Tarea 6: Votar respuestas y comentarios

Objetivo: Evaluar si el usuario comprende el sistema de votación y puede interactuar con las respuestas.

Terminada si: El usuario vota (positiva o negativamente) al menos una respuesta y explica por qué lo hizo.

Instrucciones: "Después de leer las respuestas de la pregunta anterior, imagina que alguna te resultó especialmente útil o no tan útil. Intenta votar para expresar tu opinión sobre esa respuesta."

Tarea 7: Buscar contenido

Objetivo: Observar si el usuario puede encontrar y usar la funcionalidad de búsqueda.

Terminada si: Encuentra la función de búsqueda y realiza una búsqueda sobre bonsáis.

Instrucciones: "Imagina que buscas información específica sobre 'riego de bonsáis'. Intenta encontrarla usando la plataforma."

Feedback final: ¿Qué fue lo que más te gustó y lo que menos te gustó de la experiencia? ¿Qué funcionalidad adicional te gustaría ver en la plataforma? En una escala de 1 a 5, ¿qué puntaje le das a la facilidad de uso general? ¿Por qué?

13.10.4. Resumen de feedback recibido en las entrevistas de validación

Aspectos Positivos

Facilidad de uso y navegación

Los usuarios destacaron consistentemente la facilidad de navegación y la intuitividad general de la plataforma. La mayoría logró completar las tareas en menos de 30 segundos, calificando la experiencia con puntajes máximos de facilidad de uso. El diseño fue descrito como "claro", "a la vista" y sin mayores dificultades para encontrar las funcionalidades principales. Los usuarios consideraron la plataforma "muy intuitiva", "sencilla" y "sin dificultades".

Sistema de votación y validación comunitaria

El sistema de votación fue particularmente bien recibido, con usuarios expresando que "le parece importante que se pueda votar" y valorando que "la opinión de otros usuarios pese". Los usuarios comprendieron rápidamente la diferencia entre votar positivo y negativo, encontrando los botones de votación de manera intuitiva. El concepto de "mejor respuesta" fue comprendido por la mayoría, con usuarios mostrándose contentos con esta funcionalidad para destacar respuestas de calidad.

Acceso al historial y búsqueda

La funcionalidad de "Mis Preguntas" fue elogiada, con usuarios destacando la capacidad de guardar preguntas y filtrar contenido. Los usuarios pudieron acceder a esta sección en menos de 15 segundos y entendieron cómo realizar búsquedas y filtrado por etiquetas sin dificultades. Uno de los entrevistados destacó específicamente que "le gusta que las respuestas puedan quedar guardadas y puedan revisarse luego".

Proceso de login y selección de comunidad

El proceso de inicio de sesión fue completado exitosamente por todos los usuarios en menos de 10 segundos. Los usuarios comprendieron el concepto de "comunidades" sin problema y pudieron seleccionar una antes de continuar navegando. El formulario de login resultó claro y no generó confusiones.

Configuración de perfil

Los usuarios destacaron la claridad del perfil, mencionando que "se pueden encontrar las cosas fácilmente" y apreciando funcionalidades como las notificaciones por mail con la posibilidad de habilitarlas o deshabilitarlas.

Aspectos Negativos y Oportunidades de Mejora

Problemas de responsive en móvil

Se identificó un problema significativo donde el botón de "Preguntar" no era visible hasta hacer scroll hacia abajo, lo cual generó frustración en al menos un usuario. El texto placeholder para hacer preguntas resultó demasiado largo cuando se visualiza en modo móvil, afectando la usabilidad en dispositivos pequeños.

Obligatoriedad de la descripción

Múltiples usuarios expresaron incomodidad con la obligatoriedad del campo de descripción al hacer preguntas. Consideraron que "no debería ser obligatorio, no todos quieren" agregar una descripción extensa, sugiriendo que debería ser opcional para preguntas breves o directas.

Confusión en la navegación de pestañas

Un usuario experimentó confusión al intentar identificar en qué pestaña se encontraba, intentando presionar repetidamente en "Comunidad" pese a estar ya dentro de esa sección. Esto sugiere que la indicación visual de la pestaña activa podría mejorarse.

Sistema de etiquetas

Aunque los usuarios pudieron utilizar el sistema de etiquetas, muchos no lo hicieron proactivamente, sugiriendo que no les llamó la atención o no encontraron valor inmediato en utilizarlo. Un usuario sugirió implementar "etiquetas automáticas" y otro solicitó una etiqueta específica de "técnicas de bonsái" que no estaba disponible, evidenciando limitaciones en las categorías predefinidas.

Sugerencias para futuras funcionalidades

Los usuarios sugirieron múltiples funcionalidades adicionales que consideran valiosas:

- Identificador de especies mediante foto, incluyendo detección de enfermedades y necesidades de cuidado
- Ficha de seguimiento con imágenes de plantas como bitácora de evolución
- Sección para que bonsaístas puedan mostrar la evolución de sus ejemplares
- Modo oscuro como opción en el perfil
- Botones de información popup que brinden contexto sobre cada sección
- Posibilidad de subir imágenes para realizar preguntas basadas en fotografías

Conclusiones generales

Las pruebas de validación confirmaron la madurez del prototipo y su adecuación al perfil de usuario objetivo. El sistema fue percibido como accesible, confiable y con una curva de aprendizaje prácticamente nula. Las observaciones obtenidas orientan los próximos pasos del desarrollo hacia la personalización contextual de las respuestas, el fortalecimiento de la comunidad como fuente de conocimiento colectivo, y la mejora continua de la experiencia de usuario. En conjunto, los resultados evidencian una validación positiva del producto, tanto a nivel funcional como conceptual, consolidando su potencial de adopción real por parte de la comunidad bonsaísta.

13.10.5. Resumen de las notas extraídas de devolución en revisiones

13.10.5.1. Revisión 1

Fecha: 18 de diciembre de 2024

Revisor: Amalia Álvarez

Fortalezas

- Trabajo en equipo cohesionado y eficiente con distribución equitativa de tareas y responsabilidades
- Sólida gestión del tiempo durante la presentación, ajustándose al cronograma planeado
- Cronograma bien estructurado con tiempos realistas para cada fase del proyecto
- Plan de Gestión de Configuración de Software correctamente desarrollado con procedimientos claros
- Buena capacidad para anticipar y gestionar posibles riesgos con un plan bien definido

Oportunidades de Mejora

- Definir y explicar de manera concisa la problemática que busca resolver el proyecto
- Implementar un proceso sólido de ingeniería de requerimientos con validación mediante entrevistas con usuarios reales
- Incluir en el SCM documentación para usuarios finales y registro de todos los entregables
- Aclarar el uso de una adaptación de *SCRUM* para el proceso de desarrollo
- Enfocar el Plan de SQA en prácticas preventivas que aseguren la calidad desde etapas tempranas
- Definir con claridad quiénes son los usuarios principales y cómo se llegará a ellos
- Justificar todas las decisiones tecnológicas desde un punto de vista técnico y de negocio
- Identificar riesgos asociados al uso del producto, como propiedad intelectual
- Establecer un mecanismo para monitorear y visualizar la evolución de los riesgos

- Contemplar todos los activos del producto final incluyendo entregables, documentación técnica y manuales

13.10.5.2. Revisión 2

Fecha: 10 de abril de 2025

Revisor: Rafael Bentancur

Fortalezas

- Enfoque centrado en el usuario real mediante encuestas y entrevistas que permitieron acceder a información relevante y contextual
- Estructura basada en design thinking, considerada adecuada al tipo de desafío abordado
- Transición metodológica vista como una adaptación adecuada ante la incertidumbre del proyecto
- Uso de prototipos para probar aspectos de la interfaz (colores, contraste, modos y disposición de elementos)
- Recolección de sugerencias y nuevas funcionalidades surgidas de la interacción con usuarios reales
- Uso de métricas como indicador de madurez en el desarrollo y seguimiento del proyecto
- Evaluación de la precisión del modelo con validación humana e incorporación de retroalimentación especializada

Oportunidades de Mejora

- Describir con mayor claridad cómo se aplicó el Design Thinking a lo largo del desarrollo
- Explicar el cambio de metodología, justificando la decisión en función del contexto del proyecto
- No limitarse a valores promedio en el análisis de métricas; incorporar gráficos que muestren tendencias, picos o caídas
- Explicar cómo las métricas influyeron en la toma de decisiones y en la evolución del producto

- Desarrollar con más detalle la relación entre atributos de calidad, requerimientos no funcionales y actividades para garantizarlos
- Reflejar formalmente todos los atributos de calidad en el informe
- Incluir diagramas que muestren cómo se tomaron decisiones de diseño alineadas a criterios como precisión, costos y rendimientos
- Agregar diagramas de interacción entre componentes, especialmente para procesos complejos
- Mostrar de forma clara la distribución temporal del trabajo con hitos relevantes y escala cronológica
- Definir claramente qué es un bonsái al iniciar la presentación
- Definir el problema sin incluir al LLM dentro de esa definición
- Mencionar el propósito del análisis de competidores y por qué las soluciones existentes no resuelven el problema

13.10.5.3. Revisión 3

Fecha: 26 de agosto de 2025

Revisor: Gastón Mousqués

Fortalezas

- Enfoque claro en las necesidades reales del usuario final evidenciado en relevamientos con encuestas y entrevistas
- Proceso metodológico con lógica coherente con design thinking, apropiado para la complejidad del desafío
- Claridad en la comunicación de aspectos técnicos y conceptuales: definición del problema, proceso de descubrimiento, explicación del uso de IA y descripción del funcionamiento del sistema
- Presentación de pruebas realizadas con el modelo, demostrando su funcionamiento y potencial aplicación en entorno real

Oportunidades de Mejora

- Explicar con mayor precisión las limitaciones impuestas por el cliente
- Presentar la solución después del problema, acompañando la demo con explicación simultánea que evidencie cómo cada funcionalidad responde a una necesidad específica
- Presentar los requerimientos de forma más accesible mediante imágenes, ejemplos sencillos o representaciones visuales
- Mejorar la notación del diagrama de arquitectura para que sea más clara y estandarizada
- Explicar de manera conjunta las decisiones tecnológicas adoptadas y sus fundamentos
- Detallar paso a paso el proceso metodológico seguido, describiendo ceremonias y mostrando el tablero de gestión
- Definir el esfuerzo como unidad de medida en lugar de contabilizar tickets
- En gráficas y métricas, explicar claramente qué representan, incluyendo títulos y ejes debidamente identificados
- Incorporar datos y métricas concretas de calidad: número de fallas, retrabajo, cobertura de pruebas
- Explicar previamente los requerimientos no funcionales y detallar las decisiones técnicas que derivaron de ellos
- Ampliar la descripción de pruebas realizadas, diferenciando entre usabilidad, unitarias e integrales
- Agrupar gestión de configuración con calidad, corrigiendo términos y describiendo cómo se gestionaron documentos, código, builds y versiones

13.10.6. Análisis de heurísticas de Nielsen

Visibilidad del estado del sistema

Se muestra al usuario en qué comunidad está interactuando así como que usuario está logueado.

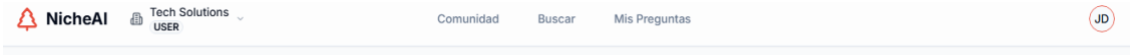


Ilustración 76: Visualización de la comunidad

Se implementaron mensajes de carga ("Cargando...", "Enviando...", "Preguntando...") que aparecen durante operaciones asíncronas, proporcionando feedback visual inmediato.

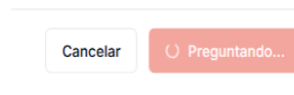


Ilustración 77: Visualización de mensajes de carga

El sistema de toasts notifica al usuario sobre acciones completadas exitosamente o errores ocurridos, con mensajes descriptivos que indican claramente qué sucedió.



Ilustración 78: Visualización de acciones completadas exitosamente

Relación entre el sistema y el mundo real

La plataforma utiliza un lenguaje natural y familiar para los usuarios hispanohablantes, evitando términos técnicos innecesarios. Los iconos utilizados son representativos y fácilmente reconocibles.



Ilustración 79: Visualización de iconos utilizados

Problema encontrado: En el diseño inicial se utilizó la palabra tenant para definir las comunidades, la cual se encontró que es un término demasiado técnico y los usuarios tenían problemas entendiendo a qué se refiere por lo cual se resolvió utilizar la palabra comunidad.

Control y libertad del usuario

Existe un botón "Volver" explícito en las páginas de detalle de preguntas que permite navegar hacia atrás. El sistema permite cambiar de comunidad en cualquier momento mediante el dropdown de selección de tenant.

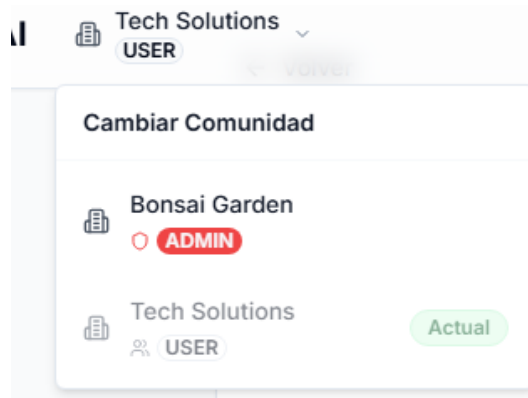


Ilustración 80: Cambio de comunidad

Problema encontrado: Los usuarios no pueden eliminar sus preguntas realizadas. Se podría solucionar agregando un icono de basurero a la derecha de cada pregunta en la sección "Mis preguntas".

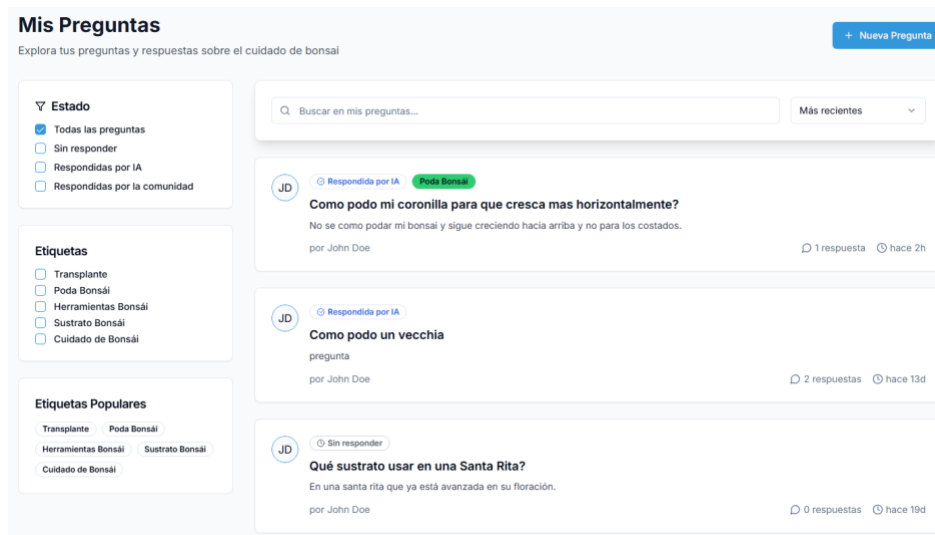


Ilustración 81: Visualización de "Mis Preguntas"

Consistencia y estándares

Configuración
Personaliza tu experiencia en NicheAI

Información Personal

Nombre: John Apellido: Doe

Nombre de Usuario: johndoe Correo Electrónico: user1@example.com

Fecha de Nacimiento: 01/30/2002 País: Uruguay
Este país se utilizará para identificar mejor las especies de árboles y bonáis en tus preguntas

Sitio Web: https://tu-sitio.com

Dirección: Si completas un campo, todos los campos de dirección son obligatorios
Calle: 123 Main St

Código Postal: 12345 Ciudad: New York

País de Residencia: USA

Ilustración 82: Configuración de información personal

Los patrones de interacción son uniformes: todos los botones primarios tienen el mismo estilo, los formularios siguen la misma estructura, y los mensajes de feedback utilizan el mismo sistema de toasts. La navegación sigue convenciones web estándar con header fijo, menú de navegación claro y breadcrumbs donde es apropiado. Los iconos son consistentes en toda la aplicación (Lucide React) y los colores tienen significados consistentes (verde para éxito/aceptado, rojo para error/rechazado, azul para información/IA).

Hacer una Pregunta ×

¡Descubre los secretos del bonsái! Haz tu pregunta y recibe respuestas expertas de IA alimentada por la sabiduría de toda tu comunidad.

Pregunta *

Descripción *

0 caracteres

Etiquetas

Agrega hasta 5 etiquetas para describir de qué trata tu pregunta

Etiquetas (mostrando 5 de 5)

Transplante Poda Bonsái Herramientas Bonsái Sustrato Bonsái Cuidado de Bonsái

Cancelar Preguntar

Ilustración 83: Visualización de "Hacer una pregunta".

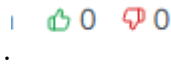


Ilustración 84: Iconos

Prevención de errores

Los campos de formulario incluyen validación en tiempo real que previene el envío de datos inválidos.

Descripción *

Mínimo 5 caracteres (4/5)

Ilustración 85: Validación en formularios.

Los botones se deshabilitan durante operaciones para prevenir múltiples envíos accidentales.

Reconocimiento antes que recuerdo

El selector de comunidades muestra todas las comunidades disponibles visualmente, sin requerir que el usuario recuerde nombres. Las respuestas anteriores del usuario son accesibles desde "Mis Preguntas" sin necesidad de recordar dónde preguntó.

Flexibilidad y eficiencia de uso

La interfaz es responsive y se adapta a diferentes dispositivos, ofreciendo accesos optimizados para móvil.

Problema encontrado: Algunas secciones no estaban respondiendo bien a ser responsive se agregó como un ticket de bug para poder mejorar esas secciones. Por ejemplo, el texto default para hacer preguntas es demasiado largo cuando se está en modo móvil.

Diseño estético y minimalista

Los iconos son simples y representativos, sin decoración excesiva. La tipografía es clara y legible con jerarquías bien definidas. Los elementos interactivos tienen estados visuales claros (hover, active, disabled) sin ser intrusivos.

Ayudar a los usuarios a reconocer, diagnosticar y recuperarse de errores

Los mensajes de error son descriptivos y orientados a la solución. El sistema de toasts muestra errores con variante "destructiva" visualmente distintiva. Los mensajes de error incluyen descripción del problema ("Error al iniciar sesión", "No se pudo guardar la pregunta").

Errores encontrados: Se encontró un caso donde el mensaje de error no es descriptivo. Se trata durante la activación de los usuarios, donde una vez vencida la invitación el error que se muestra es el siguiente, el cual debería ser corregido por “Invitación vencida, por favor comuníquese para reenviar la invitación”

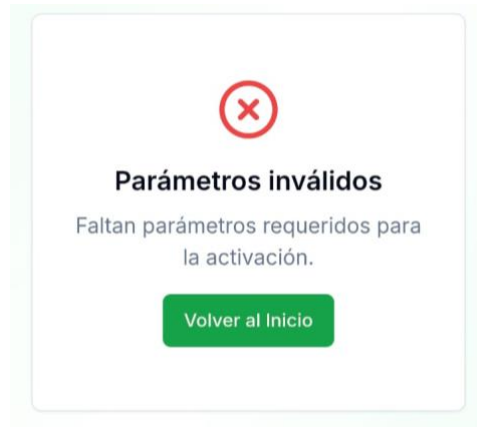


Ilustración 86: Visualización de error.

Ayuda y documentación

Errores encontrados: No hay una sección de FAQ o guía de usuario visible. Tampoco hay tutoriales o onboarding para nuevos usuarios. Se podría solucionar implementando una sección de FAQs o añadiendo tooltips según el contexto del usuario para ayudarlo en las interacciones que causen más dificultades entre los usuarios.

13.10.7. Encuesta de satisfacción del cliente

¿En qué grado la plataforma cumple con las expectativas en función del alcance acordado?: Cumple completamente

Nivel de satisfacción con la plataforma: Muy satisfecho

Funcionalidades/características más valoradas de la plataforma: “Las respuestas vienen con un tono que transmite confianza. No suenan a imposiciones que daría un super gurú, lo cual está bien porque deja lugar a que la AI se equivoque. El hecho de poder agregar respuestas a mano y que quién hizo la pregunta indique cuál le parece mejor también se valora. Finalmente, creo que las opciones de agregar datos al modelo a través de archivos o de forma manual es algo destacable (y fácil de utilizar por usuarios no técnicos)”

Sugerencias de mejora: “El modelo siempre se puede mejorar, teniendo en cuenta nuevas casuísticas, pero hoy hace lo que se pidió. Mejoraría un poquito el proceso de feedback de la UI al momento de cargar la información en la sección de "Gestión de Datos de Conocimiento””.

¿Profesionalismo del equipo para abordar el proyecto?: Excelente

¿Cómo valoras la comunicación con el equipo?: Muy fácil

Aspectos más valorados del equipo: “Se sintió que separaron tareas y tenían perfiles bien definidos. Eso hizo fácil la comunicación dado que cuando un integrante del equipo te chateaba/hablaba sabías de qué temática iba la charla. En todo momento se manejaron muy profesionalmente: cuando necesitaron algo, lo pidieron; siempre fueron super concretos en las reuniones, teniendo una agenda clara de lo que querían hablar.”

Oportunidades de mejora del equipo: “Como cliente no encuentro aspectos a mejorar del equipo. Seguramente ustedes de su lado tengan oportunidades de mejora, pero de mi lado funcionó todo muy bien.”

Evaluación integral del proyecto: Excelente

Aspectos más valorados del proyecto: "Creo que el resultado es compacto, está alineado con lo que se pidió y más importante, con aquellos requerimientos implícitos por el contexto del proyecto (UI, comportamientos esperables de ciertas features, etc.). Hubo una gestión y de mi lado sentí que tenían el control del proyecto."

Oportunidades de mejora del proyecto: "Creo que si tuviera que hacerlo de nuevo, habría tratado de "empujar" las tareas de validación del modelo más para el comienzo del plan. Creo que hoy estamos en un momento en el cual está empezando a llegar feedback, pero la mayoría del feedback con oportunidades de mejora que recibí giran en torno al modelo ("no encontré datos de esto", "pregunte por esto y me devolvió esto otro demasiado genérico", etc.). Mejoraría eso: salir antes con una versión para usuarios finales que permitiera solo probar el modelo (pregunta-respuesta... sin usuarios, simplemente recibiendo la info de contexto ingresada a mano."

¿Hay algo más que te gustaría contarnos o sugerirnos para seguir mejorando?: "Hay muchas cosas a mejorar que no tiene sentido comentarlas en este momento porque se mejoran con experiencia (llegar al cerebro a través de los oídos no funciona). Más que mejorar, deberían mantener la forma de trabajo una vez que salgan del modo "proyecto de grado". Lo que hicieron en este proyecto es como se debe trabajar en un proyecto de desarrollo de software. ¿Esto quiere decir que tengo que pelearme a muerte si no se hace de esta forma? NO! Muchas veces van a tener que trabajar de otra forma, pero sean críticos con sus procesos. No me gustaría que en 1-2 años me cruce con alguno de ustedes en algún proyecto y los vea "haciendo desastres" y diciendo "naaaa... eso lo hacía prolijo porque era el proyecto para recibirme" (nota: me ha pasado)."