Universidad ORT Uruguay

Facultad de Ingeniería

Diplovalid: Registro y validación de calificaciones académicas

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

Bruno Gallo - 194647

Marcos Irisarri - 141354

Tutor: Juan Pablo Russo

2020

Declaración de autoría

Nosotros, Bruno Gallo y Marcos Irisarri, 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 el proyecto de grado de la carrera de Ingeniería en Sistemas en la Universidad ORT Uruguay;
- 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 esas 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.

Bruno Gallo

5 de marzo de 2020

Marcos Irisarri

5 de marzo de 2020

Agradecimientos

Queremos comenzar agradeciendo a nuestras familias que nos han acompañado durante el transcurso de toda la carrera y el proyecto. Sin su apoyo incondicional hubiera sido imposible llegar a este punto.

Agradecemos especialmente a nuestro tutor, Juan Pablo Russo, por guiarnos durante todo este proceso con su buena voluntad y disposición hacia el mejor resultado posible.

A Juan Martín Gallo y el resto del equipo de Effectus, por todo el apoyo, comprensión y dedicación brindados durante todo el proyecto. También agradecemos a Alejandro Narancio y Pablo San Nicolás por su asistencia desde el lado técnico y del negocio respectivamente.

Finalmente, agradecemos a la Universidad ORT en nombre de sus profesores, asistentes y todo el personal que de alguna forma u otra acompañó este trabajo.

A todos los mencionados, muchas gracias.

Abstract

El presente documento narra el proceso que el equipo transitó para la definición, ejecución y finalización del proyecto de fin de carrera de Ingeniería en Sistemas, que tuvo como producto final el sistema Diplovalid.

El proyecto consta en la creación de una aplicación web que permita el registro y validación de títulos académicos de forma segura, asegurando la integridad de la información registrada. Se creará una herramienta que registre la información de diplomas y títulos en una Blockchain. No siendo posible alterar registros viejos, se asegura que una vez que un diploma fue validado por la institución emisora este no podrá ser cambiado ni falsificado. De esta forma cualquier persona puede compartir sus logros y títulos en linkedin, curriculums, o con potenciales empleadores mediante un simple vínculo a la vez que certifica su validez.

Para la ejecución del proyecto se utilizó un proceso llamado *dual track agile*, dividiendo el mismo es dos etapas bien definidas; primero una etapa de *discovery* para el relevamiento de requerimientos, y luego una etapa de *delivery* para el desarrollo de las funcionalidades. Durante la etapa de desarrollo se utilizó *Scrum* como marco de trabajo, y los lenguajes de programación usados para la construcción de la solución fueron NodeJs, React, y Solidity.

Como resultado se obtiene un producto mínimo viable (MVP), con el cual se logra validar en un proyecto real las ventajas del uso de blockchain como base de un mecanismo de verificación de integridad.

Se describe aquí en detalle el problema detectado mediante distintas técnicas de relevamiento de requerimientos, la solución propuesta, el proceso de ingeniería llevado a cabo, el marco metodológico elegido, y la arquitectura creada para la solución. Se incluyen también capítulos referentes a toda la gestión del proyecto, de riesgos, y de calidad, con el fin de presentar el camino recorrido durante la realización de este proyecto

Concluye el documento con las conclusiones obtenidas tras la finalización del proyecto y las lecciones aprendidas en cada una de las áreas de trabajo.

Palabras clave

Diplomas, blockchain, smart contract, contrato inteligente, RSK, ethereum, registro, validación, integridad, seguridad, universidad, empleo, rrhh, calificación

Glosario

A

Acoplamiento: grado de interdependencia entre los módulos de software; una medida de cuán estrechamente conectados están dos rutinas o módulos; la fortaleza de las relaciones entre los módulos.

API: interfaz de programación de aplicaciones. Es una interfaz o protocolo de comunicación entre diferentes partes de un programa informático destinado a simplificar la implementación y el mantenimiento del software. Una API puede ser para un sistema basado en la web, sistema operativo, sistema de base de datos, hardware de computadora o biblioteca de software.

B

***Back-end*:** se refiere a la separación de intereses entre la capa de presentación y la capa de acceso a datos en la Ingeniería de Software. El *back-end* corresponde a la capa de acceso a datos, que se encuentra del lado del servidor y que procesa la entrada desde el frontend.

Bitcoin: tipo de moneda digital en el que se mantiene un registro de transacciones y se generan nuevas unidades de moneda mediante la solución computacional de problemas matemáticos, y que opera independientemente de un banco central.

Blockchain: estructura de datos en la que la información contenida se agrupa en conjuntos (bloques) a los que se les añade meta-informaciones relativas a otro bloque de la cadena anterior en una línea temporal, de manera que gracias a técnicas criptográficas, la información contenida en un bloque solo puede ser repudiada o editada modificando todos los bloques posteriores. [1]

***Brainstorming*:** discusión grupal para producir ideas o resolver problemas.

C

CEO: director ejecutivo. Es la persona de más alto rango en una empresa u otra institución, responsable en última instancia de tomar decisiones de gestión.

***Cloud computing*:** significa que el hardware y software es proporcionado como un servicio de otra empresa a través de Internet, por lo general de una manera completamente transparente

Contrato inteligente: un contrato inteligente es un programa informático que facilita, asegura, hace cumplir y ejecuta acuerdos registrados entre dos o más partes (por ejemplo personas u organizaciones). Son programas que viven en un sistema no controlado por ninguna de las partes, o sus agentes, y que ejecutan un contrato automático. Antes de la aparición de la *blockchains*, no existía ninguna plataforma que pudiera hacer realidad los contratos inteligentes, por lo que solo estaban definidos conceptualmente. En este caso, el contrato inteligente será ejecutado en cada nodo de la *blockchain* y tendrá todos los beneficios de la tecnología Blockchain.

Código fuente: un conjunto de líneas de texto con los pasos que debe seguir la computadora para ejecutar un programa informático. El código fuente de un programa está escrito por un programador en algún lenguaje de programación, pero en este primer estado no es directamente ejecutable por la computadora, sino que debe ser traducido a otro lenguaje o código binario.

CSV: tipo de documento en formato abierto sencillo para representar datos en forma de tabla, en las que las columnas se separan por comas.

D

DApp: es una abreviación de “aplicación descentralizada”. Una DApp tiene su código de *back-end* ejecutándose en una red descentralizada *peer-to-peer*, a diferencia de las aplicaciones centralizadas donde el código de *back-end* se ejecuta en servidores centralizados.

***Debug*:** ejecución controlada de un programa o código para seguir cada instrucción ejecutada y localizar así errores.

***Design thinking*:** es un proceso para la resolución creativa de problemas. El *design thinking* tiene un núcleo centrado en el ser humano. Alienta a las organizaciones a centrarse en las personas para las que están creando, lo que conduce a mejores productos, servicios y procesos internos.

Despliegue: son todas las actividades que hacen que un sistema de software esté disponible para su uso. El proceso de despliegue general consta de varias actividades interrelacionadas con posibles transiciones entre ellas. Estas actividades pueden ocurrir en el lado del productor, en el lado del consumidor, o en ambos.

E

Encriptación: proceso de transformar información o datos en un código, especialmente para evitar el acceso no autorizado.

***End point*:** un *endpoint* es un extremo de un canal de comunicación. Cuando una API interactúa con otro sistema, los puntos de contacto de esta comunicación se consideran endpoints.

Ethereum: plataforma de cómputo distribuido de código abierto, pública, basada en blockchain. También puede ser considerado un sistema operativo que ofrece funcionalidad de contrato inteligente. Admite una versión modificada del consenso de Nakamoto a través de transiciones de estado basadas en transacciones

F

***Feedback*:** información sobre las reacciones a un producto, generalmente por parte de clientes o usuarios, que se utiliza como base para la mejora.

***Front-end*:** refiere a la separación de intereses entre una capa de presentación y una capa de acceso a datos en la Ingeniería de Software. El *front-end* corresponde

a la capa de presentación. Esta se encuentra del lado del cliente e interactúa con los usuarios.

Función Hash (criptográfica): algoritmo matemático que mapea data de un tamaño arbitrario (generalmente llamado “mensaje”) a un *bit string* de un tamaño fijo (llamado *hash*). Son funciones de tipo *one-way*, es decir, que es una función prácticamente irreversible.

Una función criptográfica de hash ideal tiene las siguientes propiedades:

- Es determinística: el mismo mensaje siempre resulta en el mismo *hash*.
- Computa el *hash* de cualquier mensaje rápidamente.
- No es feasible generar un mensaje correspondiente a un *hash*.
- No es feasible encontrar dos mensajes diferentes con el mismo *hash*.
- Un pequeño cambio en el mensaje debe ocasionar grandes cambios en el *hash* resultante.

***Function-as-a-service*:** es una categoría de servicios de computación en la nube que proporciona una plataforma que permite a los clientes desarrollar, ejecutar y administrar funcionalidades de aplicaciones sin la complejidad de construir y mantener la infraestructura típicamente asociada con el desarrollo y el lanzamiento de una aplicación.

G

Git: software utilizado para el control de versiones de aplicaciones. Su función es llevar un historial de los cambios realizados a archivos y coordinar el trabajo de personas realizado sobre archivos compartidos.

Gitflow: es un modelo de flujo de trabajo de Git que nos permite un mejor control de los proyectos.

***Google cloud*:** ofrecido por Google, es un conjunto de servicios de computación en la nube que se ejecuta en la misma infraestructura que Google usa internamente para sus productos de usuario final, como Google Search y YouTube

H

Hash: es la salida resultado de aplicar la función *hash* a una entrada.

***Hosting*: s**ervicio en línea que te permite publicar un sitio o aplicación web en Internet.

I

Interfaz de usuario (UI): los medios por los cuales el usuario y un sistema informático interactúan, en particular el uso de dispositivos de entrada y software.

***Internet of things* (IoT):** es un sistema de dispositivos informáticos interrelacionados, en el cual las máquinas mecánicas y digitales están provistas de identificadores únicos (UID) y la capacidad de transferir datos a través de una red sin requerir interacción de persona a persona o de persona a computadora .

J

JSON: es un formato de archivo estándar abierto, y un formato de intercambio de datos, que utiliza texto legible por humanos para almacenar y transmitir objetos de datos que consisten en pares de atributos-valores y tipos de datos de *array* (o cualquier otro valor serializable).

K

Kanban: es un método usado para gestionar la creación de productos con énfasis en la entrega continua sin sobrecargar al equipo de desarrollo. Al igual que Scrum, Kanban es un proceso diseñado para ayudar a los equipos a trabajar juntos de manera más efectiva.

L

Linkedin: LinkedIn es un sitio web de redes sociales diseñado para profesionales de negocios. Permite compartir información relacionada con el trabajo con otros

usuarios y mantener una lista en línea de contactos profesionales. Al igual que Facebook y MySpace, LinkedIn permite crear un perfil personalizado.

Log: grabación secuencial en un archivo o en una base de datos de todos los acontecimientos (eventos o acciones) que afectan a un proceso particular (aplicación, actividad de una red informática, etc.).

M

Mock: en la programación orientada a objetos, los *mocks* son objetos simulados que imitan el comportamiento de los objetos reales de manera controlada, generalmente como parte de una iniciativa de prueba de software.

Modificabilidad: es un atributo de calidad del software que se relaciona con el costo de introducir cambios y la facilidad con la cual un sistema los incorpora.

Machine learning: es el estudio científico de algoritmos y modelos estadísticos que utilizan los sistemas informáticos para realizar una tarea específica sin usar instrucciones explícitas, sino que se basan en patrones e inferencia. Es visto como un subconjunto de la inteligencia artificial.

MEC: Ministerio de Educación y Cultura de Uruguay, es el responsable de la coordinación de la educación nacional, de la promoción del desarrollo cultural del país, de la preservación del patrimonio artístico, histórico y cultural de la nación, así como de la innovación, la ciencia y la tecnología y de la promoción y fortalecimiento de la vigencia de los derechos humanos.

M.V.P. (Producto mínimo viable): es una versión de un producto con suficientes características para satisfacer a los clientes y usuarios iniciales, y proporcionar retroalimentación para el desarrollo futuro.

N

NPM: sistema de gestión de paquetes por defecto para Node.js.

P

***Peer-to-peer*:** es una arquitectura de aplicación distribuida que reparte tareas o cargas de trabajo entre pares (*peers*). Estos son igualmente privilegiados, participantes equipotentes en la aplicación. Se dice que forman una red de nodos *peer-to-peer*.

Performance: es un indicador de qué tan bien un sistema o componente de software cumple con sus requisitos de "puntualidad". La puntualidad se mide en términos de tiempo de respuesta o rendimiento. El tiempo de respuesta es el tiempo requerido para responder a una solicitud. El rendimiento de un sistema es el número de solicitudes que se pueden procesar en un intervalo de tiempo especificado.

Placeholder: cadena de caracteres que toma temporalmente el lugar de un dato final**.**

***Product backlog*:** es una lista de las nuevas características, cambios en las características existentes, correcciones de errores, cambios de infraestructura u otras actividades que un equipo puede realizar para lograr un resultado específico.

***Product owner*:** es el miembro del equipo ágil responsable de definir historias y priorizar el *product backlog* del equipo para agilizar la ejecución de las prioridades del programa mientras se mantiene la integridad conceptual y técnica de las características o componentes para el equipo.

Prueba de concepto: experimento o proyecto piloto que tiene como fin generar evidencia para demostrar que un concepto de diseño, propuesta comercial, etc., es factible.

R

Repositorio: en el desarrollo de software, un repositorio es una ubicación central de almacenamiento de archivos. Es utilizado por los sistemas de control de versiones para almacenar múltiples versiones de archivos. Si bien un repositorio se puede configurar en una máquina local para un solo usuario, a menudo se almacena en un servidor, al que pueden acceder varios usuarios.

***Release*:** es el proceso de lanzamiento de un nuevo producto para un mercado específico o una base de usuarios. En el desarrollo de software, el lanzamiento de un producto a veces se realiza con una versión beta para que los desarrolladores / usuarios principales puedan ayudar con retroalimentación antes del lanzamiento del software real.

***Roadmap*:** cronograma que determina las actividades a realizar de un programa largo o complejo.

RSK: es una plataforma de contratos inteligentes que está conectada a la cadena de bloques de Bitcoin a través de la tecnología de *sidechain*. Rootstock es una versión avanzada de QixCoin, una criptomoneda turing-completa creada en 2013 por el mismo equipo de desarrollo.

S

Scrum: un conjunto de prácticas utilizadas en la gestión ágil de proyectos que enfatizan la comunicación diaria y la reevaluación flexible de los planes que se llevan a cabo en fases cortas e iterativas de trabajo.

***Scrum master*:** es el facilitador de un equipo de desarrollo ágil. Scrum es una metodología que permite que un equipo se auto-organice y realice cambios rápidamente, de acuerdo con principios ágiles. El scrum master gestiona el proceso de cómo se intercambia la información.

SDK: conjunto de herramientas de software necesarias para desarrollar programas que interactúen con otro software mediante una API.

Seguridad: medida de la habilidad de un sistema de proteger su data e información de accesos no autorizados mientras le provee acceso a personas y sistemas que sí están autorizados.

***Sidechain*:** son mecanismos emergentes que permiten que los tokens y otros activos digitales de una *blockchain* se utilicen de forma segura en una *blockchain*

separada y luego se vuelvan a mover a la cadena de bloques original si es necesario.

Software as a Service (SaaS)**:** modelo de distribución de software en el que tanto el software como los datos manejados son centralizados y alojados en un único servidor externo a la empresa.

Solidity: lenguaje de alto nivel orientado a contratos inteligentes. Su sintaxis es similar a la de JavaScript y está enfocado específicamente a la máquina virtual de Ethereum (EVM).

Sprint**:** es cada una de las iteraciones que se encuentran dentro de un proyecto Scrum. Al final de cada *sprint* se cuenta con un incremento del producto que aporta valor al cliente.

Sprint backlog**:** es el conjunto de elementos del *product backlog* seleccionados para el sprint.

T

Template: Combinación de archivos que componen la parte visual de un sitio web.

Time box**:** es un período de tiempo fijo que se le asigna a una actividad para que se lleve a cabo.

Transacción: una transacción en *blockchain* se puede definir como una pequeña unidad de tarea que se almacena en registros públicos. Estos registros también se conocen como bloques. Estos bloques se ejecutan, implementan y almacenan en la *blockchain* solo después de la validación por todas las personas involucradas en la red *blockchain*.

Turing-completo: un lenguaje es Turing completo si teóricamente puede resolver cualquier problema siempre que tenga suficiente tiempo y recursos.

U

Usabilidad: está relacionada con cuán fácil es para un usuario lograr una tarea deseada y el soporte que el sistema le provee al usuario.

V

***Value proposition canvas*:** es una herramienta que puede ayudar a garantizar que un producto o servicio se posicione en torno a lo que el cliente valora y necesita. El *value proposition canvas* se puede utilizar cuando sea necesario refinar una oferta de producto o servicio existente o cuando se está desarrollando una nueva oferta desde cero.

W

***Work in progress* (WIP):** es el número de elementos de tarea en los que un equipo está trabajando actualmente. Limitar el trabajo en progreso es una de las propiedades principales de Kanban.

Índice

1. Introducción

El propósito del presente documento es describir el proyecto final de grado “Diplovalid: Registro y validación de calificaciones académicas”, realizado como requisito para la obtención del título de Ingeniero en Sistemas de la Universidad ORT. El proyecto fue realizado entre marzo de 2019 y marzo de 2020.

1.1. Presentación del equipo

El equipo está integrado por Marcos Irisarri y Bruno Gallo, ambos estudiantes de la carrera Ingeniería de Sistemas en Universidad ORT, quienes comenzaron a cursar la carrera en el primer semestre del año 2014. A lo largo de los años han compartido varias cursos y realizado múltiples trabajos obligatorios en equipo. También han compartido muchas horas de estudio durante la preparación de parciales y/o exámenes.

El hecho de haber compartido años de estudio previamente es relevante al momento de decidir formar equipo para la realización de la tesis final de la carrera. Se parte de una sinergia de trabajo y estudio establecida, además de haber desarrollado una relación de amistad con el tiempo. Debido a estos factores se decide realizar la tesis en un grupo de dos integrantes, prefiriendo esta opción frente a la posibilidad de incorporar un tercero con el que no se tiene relación previa.

El tutor del proyecto fue Juan Pablo Russo, docente de universidad ORT, quien fue de gran ayuda guiando al equipo durante todo el año para poder cumplir con los objetivos planteados.

1.2. Descripción del cliente

Effectus es una empresa start-up fundada en el año 2017 por Juan Pablo Mazza, Alejandro Monetti, Juan Martín Gallo, y Alan Tugentman, ex alumnos de ORT. Su principal rama de actividad es el desarrollo de software a medida para clientes, tanto aplicaciones web como móviles utilizando procesos ágiles para lograrlo. Desde su

fundación la empresa ha ido creciendo rápidamente, pasando de ser 4 personas trabajando inicialmente a superar la veintena actualmente.

En busca de expandirse aún más, es de interés de la empresa el desarrollo de distintas pruebas de concepto basadas en nuevas tecnologías. El objetivo es tener una base de conocimiento para poder ofrecer soluciones modernas y basadas en las últimas tecnologías, tanto a clientes nuevos como a los ya existentes.

1.3. Elección del proyecto

El proyecto surge con el objetivo de culminar un ciclo de aprendizaje y realizar la tesis de grado de Ingeniería en Sistemas en la universidad ORT Uruguay.

Durante el mes de febrero de 2019 el equipo tuvo que decidir sobre qué tipo de proyecto quería trabajar, si realizar un desarrollo para un cliente específico o realizar un emprendimiento propio. Debido al poco tiempo disponible para desarrollar una idea propia y presentarla al CIE para su validación, el equipo decide ir por la primera opción. Se solicita una reunión con los responsables de la empresa Effectus, software factory fundada por ex alumnos de ORT, con el fin de ponerse a disposición para desarrollar una prueba de concepto sobre alguna tecnología sobre la cual la empresa tuviera interés en incursionar.

Los responsable de la empresa plantean al equipo que tienen interés en comenzar a investigar nuevas tecnologías, con la finalidad de mantenerse actualizados y poder responder a las necesidades de sus clientes con las más modernas soluciones. Puntualmente les interesa profundizar en las áreas de *Machine Learning*, *Internet of Things*, y *Blockchain*.

Partiendo de un interés personal de parte de los miembros del equipo, se elige *Blockchain* como la tecnología a investigar. La empresa Effectus plantea que ellos consideran que la mejor forma de realizar esta investigación es mediante la creación de una prueba de concepto. Creen que el uso de esta tecnología en un proyecto real puede llevar a descubrir sus fortalezas, debilidades y posibles dominios de

implementación. Queda a cargo del equipo encontrar un problema que sea relevante resolver con esta tecnología.

El equipo se reúne con Alejandro Narancio, CEO de *Infuy*, una empresa uruguaya de desarrollo de software que destaca por su especialización en la construcción de soluciones basadas en tecnología blockchain. Alejandro sugiere el desarrollo de un sistema que permita el registro de algún tipo de documento en una blockchain, asegurando así la integridad y autenticidad de estos documentos. En el transcurso de la reunión surge la idea de realizar un sistema que permita el registro de diplomas académicos, siendo estos validados por la universidad emisora, y guardados en una blockchain. De esta forma un graduado podría presentar sus diplomas a potenciales empleadores asegurando su validez.

Imagen 1: Reunión con Alejandro Narancio, CEO de Infuy

Se presenta la idea a los responsables de Effectus, a quienes les parece que es un problema válido para resolver mediante el uso de blockchain, y de donde se podrán obtener valiosos aprendizajes sobre el uso de la tecnología.

Se acuerda que Effectus proveerá al equipo su respaldo para la realización del proyecto, quedando a disposición el uso de sus oficinas para trabajar, la creación de un repositorio privado donde almacenar el código fuente, y demás recursos que el equipo pueda necesitar. Se aclara que el objetivo del proyecto para la empresa es la

creación de una prueba de concepto sobre tecnología blockchain, quedando a cargo del equipo el relevamiento de requerimientos, identificación de interesados, creación y validación de prototipos, y demás tareas relacionadas a la ingeniería de requerimientos. El equipo se compromete, al finalizar el proyecto, a entregar el código fuente resultante y un documento que detalle lo aprendido. Este documento debe incluir las lecciones aprendidas, ventajas y desventajas del uso de blockchain como tecnología de almacenamiento de datos, comparación con otras alternativas, ejemplos de uso, cómo determinar si es una solución apropiada para el problema a resolver, y pasos a seguir para el uso de blockchain en un proyecto. El documento resultante al finalizar el proyecto puede verse en el anexo 12.1. Documento de transferencia de conocimiento a Effectus.

1.4. Objetivos

Antes de comenzar, se definieron los principales objetivos del proyecto: académicos, del equipo y del producto. De este modo, se aseguró que ambos integrantes del equipo trabajen hacia una meta en común. Brindaron además una forma de evaluar el éxito del proyecto de forma medible una vez finalizado el mismo.

Para definir los objetivos utilizamos el criterio S.M.A.R.T. [2], acrónimo que sugiere que los objetivos deben ser definidos de forma específica, medible, alcanzable, relevante, y estar acotados por un cierto tiempo.

1.4.1. Objetivos académicos

Objetivo	Criterio de éxito establecido
Aplicar los conocimientos técnicos y de gestión de proyectos incorporados durante el transcurso de la carrera	Puntaje obtenido mayor a 85, lo que implica una nota de excelencia demostrando que se aplicaron de forma correcta los conocimientos aprendidos en la carrera
Aprender tecnologías nuevas y emergentes	Al finalizar el proyecto cada integrante deberá tener sólidos conocimientos de al menos una nueva tecnología con la que no estuviera familiarizado al comienzo del mismo

Gestionar un proyecto de software real, aplicando todo el proceso de ingeniería de software fuera del ambiente académico	Se llega a la fecha de entrega con los requerimientos prioritarios implementados, el proceso bien documentado según feedback recibido por parte del tutor y correctores en la última revisión, y todos los actores involucrados satisfechos con el trabajo realizado de acuerdo a encuestas de satisfacción realizadas.

Tabla 1: Objetivos académicos

1.4.2. Objetivos del equipo

Objetivo	Criterio de éxito establecido
Mejorar y afianzar la relación entre compañeros de equipo	Obtener un promedio de 4 para cada integrante en la encuesta “Clima de Trabajo en el equipo” al finalizar el proyecto
Mejorar la comunicación y el trabajo en equipo	Se respeta lo acordado en el documento “Reglamento del equipo” definido antes de comenzar el proyecto

Tabla 2: Objetivos del equipo

El reglamento del equipo puede ser visto en detalle en el anexo 12.2. Reglamento del equipo.

1.4.3. Objetivos del producto

Objetivo	Criterio de éxito establecido
Lograr la satisfacción del cliente y los usuarios desarrollando un producto útil y de calidad	Obtener un resultado mayor al 80% en las encuestas finales de satisfacción del cliente y de los usuarios.
Implementar la mayor parte de los requerimientos acordados con el cliente al comienzo del proyecto	Implementación del 100% de los requerimientos definidos como prioritarios entre el cliente y el equipo

Transferir a Effectus el conocimiento y la experiencia obtenidos durante el desarrollo de un proyecto que utiliza tecnología blockchain.	Generación de un documento donde se detallen las lecciones aprendidas, ventajas y desventajas de la tecnología, ejemplos de uso, cómo determinar si es una solución apropiada para el problema a resolver, y pasos a seguir para el uso de blockchain en un proyecto. Instalación de la aplicación desarrollada en un ambiente perteneciente a Effectus, y entrega del código fuente.

Tabla 3: Objetivos del producto

1.5. Estructura del documento

Se brinda a continuación una breve explicación de las diferentes secciones que componen el documento.

Introducción

Este capítulo da una visión general sobre algunos de los aspectos fundamentales del proyecto: cómo surge, quién es el cliente y cuáles fueron los principales objetivos del equipo.

Problema y solución

Se presenta el problema planteado por la empresa y la validación del mismo. Se describe la solución propuesta y se especifica quiénes son los principales actores que participan en ella.

Marco metodológico

En esta sección se definen los roles del equipo y las diferentes fases del proyecto. Se describen las metodologías y marcos de trabajo utilizados en cada una de ellas, mencionando cómo fueron adaptados para lograr el mejor desempeño posible durante el desarrollo del trabajo.

Ingeniería de requerimientos

Se detalla el proceso que se utilizó para analizar y especificar los requerimientos funcionales y no funcionales del proyecto.

Arquitectura y desarrollo

Se describen las decisiones de diseño y arquitectura tomadas para la construcción del producto. Se analiza la solución para determinar los atributos de calidad relacionados con la arquitectura elegida. Además, se detallan algunos de los desafíos más relevantes que surgieron durante el desarrollo del producto y las tecnologías utilizadas.

Gestión del proyecto

Se determina el proceso de gestión llevado a cabo, se introduce un cronograma del proyecto, con los hitos más destacados del mismo. Se describe la aplicación del marco de trabajo, especificando la planificación de sprints y su ejecución. Más adelante se detalla la gestión de estos sprints, la gestión de alcance y las métricas de gestión obtenidas.

Gestión de riesgos

Se detallan los principales riesgos detectados al comenzar el proyecto y cómo fue su evolución a lo largo del tiempo. Se especifican planes de mitigación y contingencia para cada uno de ellos, así como el impacto que tendrían y la probabilidad de ocurrencia. Se incluyen gráficas, métricas y conclusiones al finalizar el proyecto.

Gestión de la calidad

Se describen los objetivos en términos de calidad, tanto del producto como del proceso. Se detallan las actividades que se llevaron a cabo para asegurar la calidad a lo largo del proyecto y las métricas utilizadas.

Gestión de la configuración

Se detalla la estrategia que el equipo definió para llevar adelante la gestión de la configuración del software. Se especifican los elementos de configuración de software (ECS) identificados, las herramientas utilizadas para gestionarlos, la estrategia para el manejo de versiones, y la forma en la que se controlan los cambios en el proyecto.

Conclusiones

Se presentan las conclusiones obtenidas y las lecciones aprendidas. Se explica cómo finalizó el trabajo y cuáles son los planes a futuro.

2. Problema y solución

Este capítulo busca contextualizar al lector en torno al problema existente que motivó al grupo a realizar un producto que le diera solución. También se presentan los principales interesados y la situación actual de los mismos.

2.1. Contexto del problema

Se comienza analizando el proceso completo que ocurre entre que una persona completa todos los cursos y exámenes de una determinada carrera, obtiene su diploma de graduado, y finalmente lo presenta en una empresa para solicitar trabajo o en otra universidad para seguir con sus estudios de máster o postgrado.

Luego de una reunión con Adriana Fernandez y Victoria González, integrantes del departamento de coordinación de graduados de universidad ORT, se llega al siguiente diagrama de flujo. En el mismo se puede ver cada paso del proceso desde la creación de la carrera hasta la validación del diploma de un graduado, junto con el rol del actor involucrado en cada etapa de este proceso.

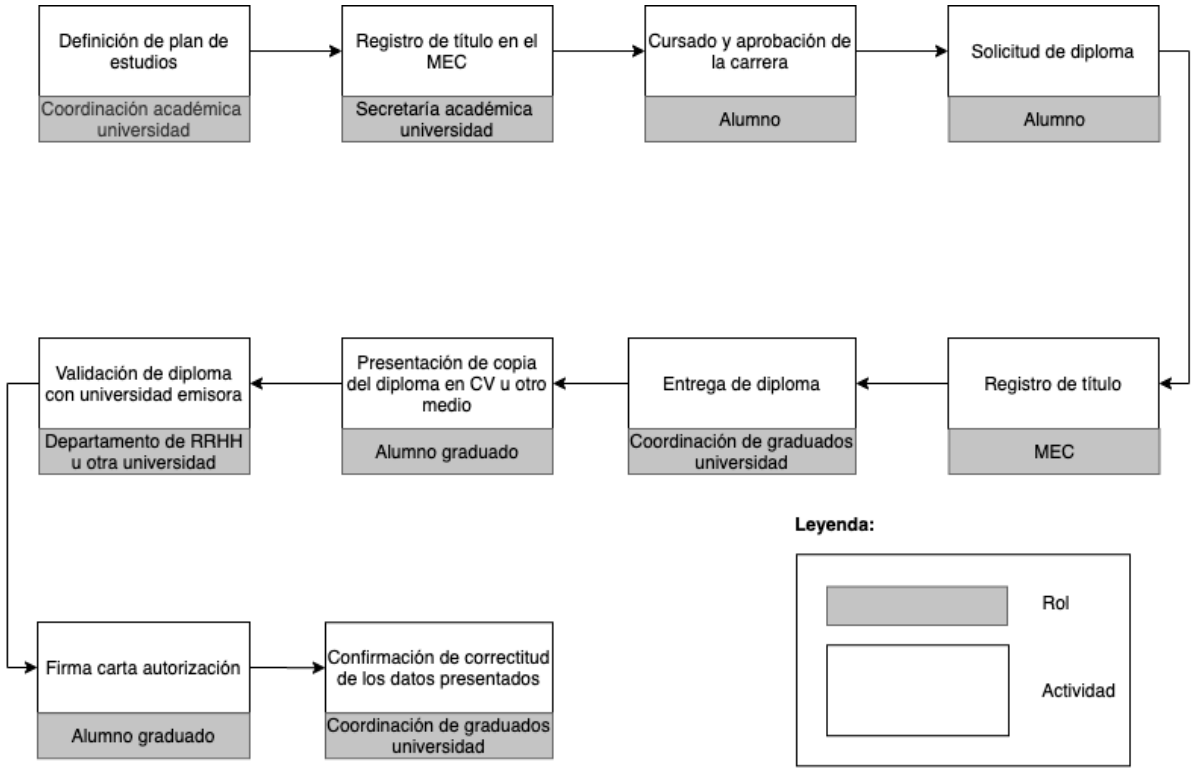


Imagen 2: Diagrama de flujo del proceso de emisión y validación de diplomas

Aclaración: El análisis realizado aplica para la emisión y validación de diplomas en el territorio nacional, en caso de presentarse en el exterior es un proceso más complejo que involucra entre otros al Ministerio de Relaciones Exteriores. Este último caso queda fuera del alcance de este proyecto.

2.2. Interesados y sus necesidades

Se incluye a continuación una descripción de cada uno de los involucrados en el proceso de emisión y presentación de diplomas, detallando el rol que cumplen y cual es su interés en el proyecto.

Rol	Descripción	Interés
Graduado	Persona que una vez completada una carrera académica recibe un diploma como evidencia del título obtenido.	Demostrar la validez de su diploma de forma certera frente a potenciales empleadores u otras instituciones académicas.
Coordinación de graduados de una universidad	Oficina que tiene por objetivo mantener el contacto con los graduados una vez que finalizan sus estudios. Tiene como tarea apoyar a los estudiantes y graduados en su inserción y mejora laboral. Asimismo, oficia de nexo permanente con las principales consultoras, empresas, organismos públicos y organizaciones internacionales de nuestro medio. [3]	Facilitar a los graduados la validación de sus diplomas frente a terceros para permitirles acceder a diversas oportunidades laborales o de especialización académica.
Departamento de RRHH o reclutamiento	Oficina encargada de contactar, entrevistar, y contratar personas para un trabajo. Puede ser parte de la empresa que solicita al personal o tratarse de una consultora independiente.	Certificar que el diploma presentado por un candidato a cubrir una determinada posición en un trabajo es válido. Confirmar que la información en cuanto al título del candidato es correcta y coincide con lo emitido por la universidad correspondiente.

Ministerio de Educación y Cultura (MEC)	El Ministerio de Educación y Cultura de Uruguay es el responsable de la coordinación de la educación nacional y de la promoción del desarrollo cultural del país.	Al no haber sido posible concretar una reunión con autoridades del ministerio, no es posible definir sus necesidades concretas. Más adelante se detallan las causas por las cuales no fue posible concretar esta entrevista.

Tabla 4: Detalle de interesados y sus necesidades

2.3. Justificación del problema

Para terminar de entender el problema que se intenta solucionar, se busca tener el punto de vista de los distintos interesados mediante encuestas y entrevistas. Finalmente se presenta un detalle de la problemática que enfrenta cada uno de los interesados actualmente en el proceso.

2.3.1. Entrevista a secretario de gestión académica

Por recomendación de Adriana y Victoria, del departamento de graduados, se agenda una reunión en Universidad ORT con Pablo San Nicolás, secretario de gestión académica, y Gabriela Sánchez, coordinadora de registro y documentación. Se busca de esta forma profundizar en el proceso de emisión y entrega de diplomas.

Imagen 3: Reunión con Pablo San Nicolás y Gabriela Sánchez

De esta entrevista surgen los siguientes pasos a seguir para la emisión de un diploma por parte de la universidad:

1. Diariamente se corre un proceso automático en el sistema de la universidad que detecta a aquellos estudiantes que han completado todos los créditos de la carrera y están en condiciones de solicitar el diploma. Para estos estudiantes se habilita una opción en el sistema de gestión que les permite realizar la solicitud del diploma, lo cual da comienzo al proceso de emisión del mismo.
    a. Aclaración: debe ser el graduado quien solicite la emisión de su diploma, no se emiten diplomas en forma automática.
2. El graduado accede al sistema de gestión y solicita la emisión de su diploma. La cédula de identidad registrada en el sistema debe estar vigente o no se le permite realizar la solicitud hasta que no suba una copia escaneada de una cédula vigente.
3. Una vez recibida la solicitud, la universidad recopila toda la información necesaria del graduado:
    a. Escolaridad
    b. Pase de secundaria que debió presentar al momento de inscribirse a la carrera
    c. Copia de la cédula de identidad
    d. En caso que el alumno provenga de otra universidad y tenga reválidas, se debe adjuntar también la resolución en que se acepta esa reválida y el certificado de la universidad de origen.

    Se verifica que la documentación sea correcta y se manda a imprimir el diploma en una cartulina ya preimpresa que tiene la universidad. Estas cartulinas tienen ciertas características para dificultar su falsificación, como el logo de la universidad, un relieve, una imagen de fondo. Además cada cartulina tiene un número secuencial utilizado para llevar un control de cada diploma impreso, similar a lo que ocurre con las facturas en los comercios.

4. El diploma impreso pasa por diferentes personas que verifican que la información y escolaridad sean correctas. A continuación se le agregan las firmas correspondientes según el tipo de carrera, en caso de carreras reconocidas por el MEC firma el secretario docente, el decano y finalmente el rector.
5. Se registra el diploma internamente en un libro de registro escrito a mano. Esto es exigencia del ministerio. Este libro queda en poder de la institución emisora del diploma.
6. En determinado momento del mes, cuando el ministerio notifica que puede recibir los diplomas, se le manda un lote con todos los diplomas que están en condiciones de enviarse para su certificación.
7. El ministerio verifica que la información del diploma sea correcta, esto es que la escolaridad coincida con el plan de estudios registrado en el ministerio, que las firmas del decano y el rector coincidan con las registradas, que no existan faltas de ortografía ni otros errores. Si está todo correcto se ingresan los datos del diploma a la base de datos del ministerio y a continuación se envía el diploma al coordinador del área de educación superior para que lo firme. Finalmente los escribanos del ministerio registran los datos del diploma en un libro interno [4]. A partir de ese momento el título pasa a tener validez oficialmente. La fecha en que se registra en los libros del ministerio es la fecha que se toma como graduación oficialmente. Todo este proceso de parte del ministerio lleva en promedio unos 3 meses. Al completarse el proceso el ministerio conserva la escolaridad y una copia del título, la restante documentación, incluido el diploma firmado, es devuelta a la universidad.
8. Una vez recibido el diploma firmado por parte del ministerio, la universidad realiza una última verificación de los datos incluidos, y adjunta una vía de la escolaridad dirigida al graduado y una carta de felicitación de parte del rector. Se ensobra todo y se le entrega a la coordinación de graduados para que contacte al alumno y le notifique que puede pasar a retirar su diploma.

Cuando el graduado quiere presentarse a un trabajo u otra universidad para seguir sus estudios de máster o postgrado, debe presentar una copia de su diploma o

incluso la versión original en algunos casos. La empresa o institución que recibe el diploma debe entonces contactactarse con la institución emisora del mismo para certificar su validez y la de la información contenida. De conformidad con la Ley N° 18.331 de "Protección de Datos Personales y Acción de Habeas Data" [5] es necesario que el graduado presente su consentimiento por escrito autorizando a la universidad emisora a confirmar o rechazar la información proporcionada por él. Desde la oficina de Pablo San Nicolás se reciben constantemente solicitudes de validación de diplomas.

Actualmente la universidad ORT dispone de una funcionalidad en su sitio web donde el graduado puede generar un código que permita a alguien externo ver una copia su diploma accediendo a la web de la universidad con este código. Por seguridad, la validez del código generado es de 30 días.

En el caso del Ministerio de Educación y Cultura, no está contemplado disponer de recursos para validar si una cierta persona se graduó de cierta carrera o no. Sólo en caso que exista una orden judicial se brinda esta información.

2.3.2. Encuesta a graduados

Para conocer el punto de vista de los graduados y su experiencia al momento de tener que certificar su preparación académica, se crea una encuesta online que se envía a más de 35 egresados académicos. Se procura enviar la encuesta a personas de distintas edades, carreras, universidades, y ciudades, buscando la mayor heterogeneidad posible entre quienes responden.

Lo primero que surge es que a la mayoría de los graduados les parece poco eficiente el proceso que tienen que llevar a cabo para obtener su diploma una vez completados los requerimientos académicos.

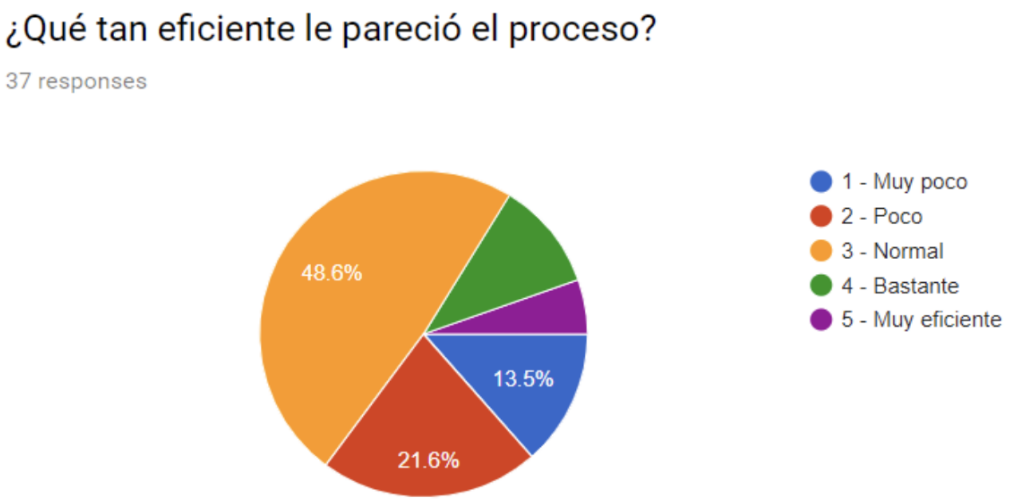


Imagen 4: Resultado de encuesta a graduados

Consultados sobre si toman alguna medida para demostrar la autenticidad de sus diplomas, la gran mayoría responde que no. Pero se muestran interesados en el uso de una herramienta que les permita a sus potenciales empleadores certificar la validez del documento presentado.

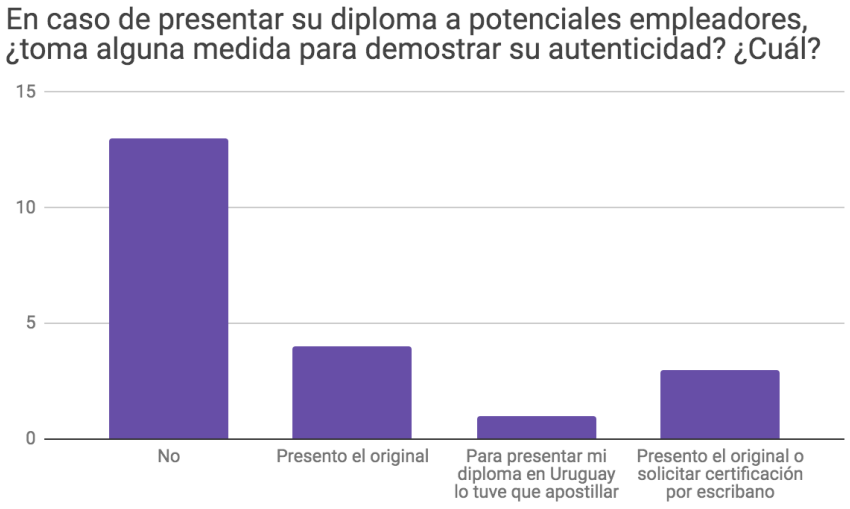


Imagen 5: Resultado de encuesta a graduados

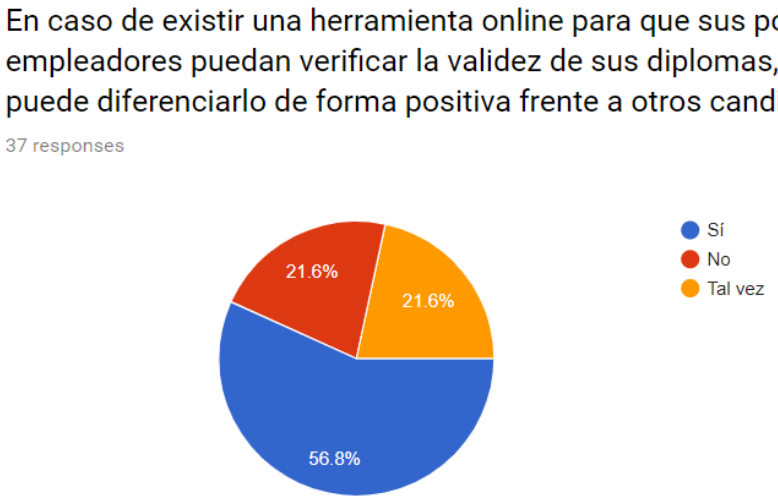


Imagen 6: Resultado de encuesta a graduados

2.3.3. Entrevista a responsables de RRHH

Para conocer la experiencia de quienes reciben los diplomas por parte de graduados, se realizan entrevistas telefónicas con personas responsables del reclutamiento de personal en diferentes empresas.

De estas entrevistas surge que los empleadores en general no verifican la validez de los títulos que los aplicantes dicen tener. Sólo en algunos casos solicitan una copia o foto del mismo. Muchos se guían por referencias laborales o personales. Asimismo, los empleadores consideran que una herramienta útil a la hora de verificar calificaciones de los aplicantes agregaría valor a sus tareas de contratación de personal.

2.3.4. Dificultad para agendar entrevista con el MEC

Si hicieron varios contactos vía mail y teléfono con el Ministerio de Educación y Cultura (MEC) con la finalidad de lograr una entrevista y entender su punto de vista en el proceso. Ante la falta de respuesta se determina que el ministerio quede fuera del alcance de este proyecto, concentrando los esfuerzos en la creación de una solución que brinde valor a los demás grupos de usuarios interesados ya entrevistados.

La solución propuesta hace foco en el proceso de validación de diplomas ya emitidos y certificados por el MEC, cómo se ve en la sección 2.4. Solución propuesta. De esta forma la participación del ministerio queda por fuera del alcance de esta solución, esto permite que la solución planteada brinde valor independientemente del involucramiento o no del ministerio. Se hace foco en la validación de diplomas en posesión de un usuario por parte de la institución académica emisora, independientemente de quien registra el diploma oficialmente.

En la sección 7.3.1 Difícil acceso a la información se analiza esta situación como un riesgo materializado en el proyecto.

2.3.5. Problemáticas detectadas

A continuación se describen las problemáticas detectadas según el tipo de rol que participa en el proceso. En la sección 4.1.2.2. Propuesta de valor, se analiza en más detalle la propuesta de valor, dolores y ganancias de cada rol.

2.3.5.1. Graduados

- **Dificultad para demostrar la validez de un título:** Un graduado que presenta su título en una aplicación para un trabajo no tiene forma de demostrar fehacientemente la validez de su diploma, más allá de incluir una copia del mismo u ofrecerse a presentar el original. Depende de la voluntad de la institución emisora en destinar recursos para estar atendiendo llamadas y correos electrónicos con pedidos de verificación.

2.3.5.2. Universidades

- Gasto de recursos y tiempo respondiendo consultas sobre la validez de diplomas presentados por alumnos en otros lugares.
- Necesidad de solicitar carta de consentimiento firmado el alumno graduado para poder divulgar la información solicitada.

2.3.5.3. Empresas

- **Dificultad para comprobar la validez de un título:** Hoy en día quien quiera verificar la validez de un título presentado por alguien, debe en primera instancia ponerse en contacto con la universidad emisora y solicitar esta información. Sin una carta de consentimiento firmado por el titular del diploma, la universidad sólo puede responder si esa persona se graduó o no, no puede responder respecto a ningún otro dato tal como la fecha de graduación, promedio de calificaciones, etc. A veces es necesario recurrir a consultoras que realicen este trabajo, lo que implica un costo para las empresas e instituciones.
- **Facilidad de falsificar diplomas:** Son decenas las universidades fantasma que venden títulos falsos a través de internet [6], una rápida búsqueda en internet revela además múltiples casos de fraude ocurridos en todas partes del mundo con títulos apócrifos.

2.4. Solución propuesta

2.4.1. Descripción general de la solución

Una vez analizado el proceso completo de emisión y validación de diplomas junto con las problemáticas detectadas para cada rol participante, y los actores a los que se tuvo acceso, se procede a determinar en qué parte de este proceso existe la posibilidad de crear una solución que brinde valor y solucione estos problemas.

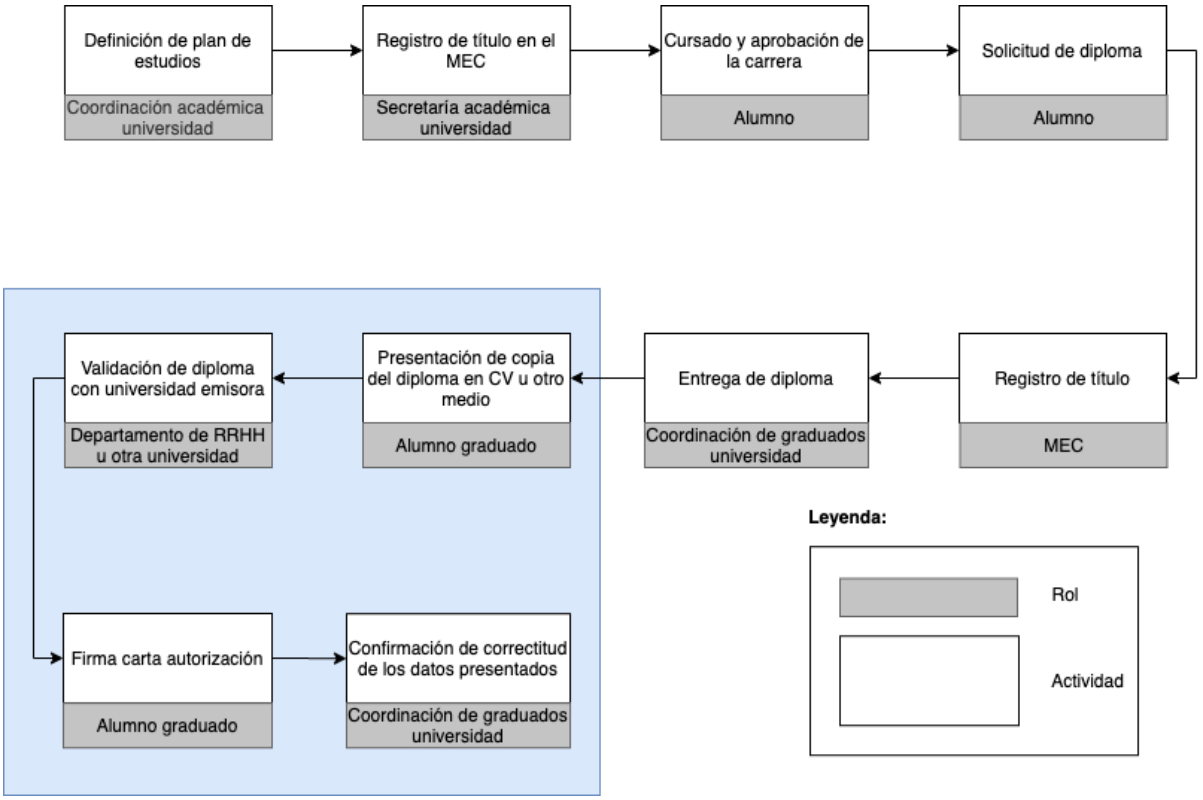


Imagen 7: Diagrama de flujo con la parte del proceso a trabajar resaltada

Queda claro que un elemento fundamental para poder brindar una solución a la problemática hallada es la seguridad de los datos. Se plantea entonces la creación de una solución que asegure la confidencialidad, integridad y autenticidad de la información almacenada para todos los involucrados.

La solución propuesta consta en la creación de una herramienta que registre la información de diplomas y títulos en una blockchain, tecnología de base de datos que permite a los usuarios insertar información la cual se asegura inmutable en el tiempo. Mediante el uso de esta tecnología además se brinda fácilmente la posibilidad de que cualquier tercero pueda consultar la validez de un diploma a través de un simple vínculo. No siendo posible alterar registros viejos se asegura que una vez que un diploma fue validado por una institución este no podrá ser cambiado ni falsificado.

De esta forma cualquier persona puede compartir sus logros y títulos en linkedin, curriculums, o con potenciales empleadores mediante un simple link.

A modo de ejemplo podrían registrarse:

- Títulos universitarios, o académicos de cualquier tipo
- Certificados de idiomas
- Certificados de logros deportivos
- Participaciones en talleres, charlas o eventos

2.4.2. Objetivo y visión del producto

Objetivos: simplificar y acelerar el proceso de verificación de calificaciones a las distintas partes involucradas, a través de un sistema que pueda ser utilizado para ingresar, verificar, y consultar calificaciones de forma segura y confiable.

Visión: introducir una forma innovadora de realizar el proceso de consulta y verificación de calificaciones que pueda ser adoptada por instituciones académicas de todo el país, con la posibilidad de ampliarse a nivel internacional, y que se adecue a las necesidades de las distintas instituciones, utilizando tecnologías de vanguardia para asegurar la confidencialidad, integridad, y disponibilidad de los datos del sistema.

2.4.3. Flujo de la aplicación

A continuación se resume el flujo de la aplicación creada como solución al problema, con el objetivo de clarificar el funcionamiento de la misma.

Existen 3 roles dentro de la aplicación: Administrador, Graduado, e Institución académica.

Administrador

El administrador de la aplicación es el encargado de crear el perfil de la institución académica en el sistema. Además de los datos básicos (nombre, dirección, teléfono, correo electrónico), se carga una lista de carreras dictadas en dicha institución. Teniendo en cuenta que algunas instituciones dictan más de 200 carreras, se ofrece la opción de cargar esta información mediante un archivo CSV.

Adicionalmente se cargan los requisitos que debe completar un graduado para solicitar la validación de su diploma. Los requisitos tienen como objetivo que el graduado verifique su identidad y autorice a la institución a verificar calificaciones a través del sistema. Por cada requisitos se indica el nombre y el tipo de dato. Estos requisitos pueden ser tanto de tipo campo (texto) como de tipo archivo, para por ejemplo solicitar la carga de un archivo PDF. De esta forma, cada institución académica puede personalizar los datos que requiere a los graduados para poder validar los diplomas. Un ejemplo de un requisito de tipo “campo” puede ser un número de estudiante, o un número de documento. Un ejemplo de un requisito de tipo archivo (imagen o PDF) puede ser el consentimiento del egresado, que es una carta firmada por el mismo dándole autorización a la institución a verificar las calificaciones ingresadas.

Una vez completada la carga de información, se le brinda a la institución académica un usuario y contraseña para acceder al sistema y comenzar a aprobar o rechazar solicitudes de validación de diplomas.

Graduado

Los graduados pueden registrarse libremente en el sistema tan sólo cargando sus datos personales.

Una vez confirmado el registro vía mail, el graduado puede acceder al sistema ingresando sus credenciales y un token que le llegará vía mail en cada intento de acceso. Este token actúa como método de autenticación de 2 factores.

Una vez dentro, el usuario tiene la posibilidad de modificar sus datos, cargar una imagen de perfil, y crear una solicitud de validación de diploma.

Para cargar una solicitud de validación de su diploma, el usuario debe primero seleccionar la institución académica donde cursó sus estudios.

Una vez seleccionada la institución, se muestran los requisitos que el graduado debe completar para realizar una solicitud de validación de diploma en dicha institución.

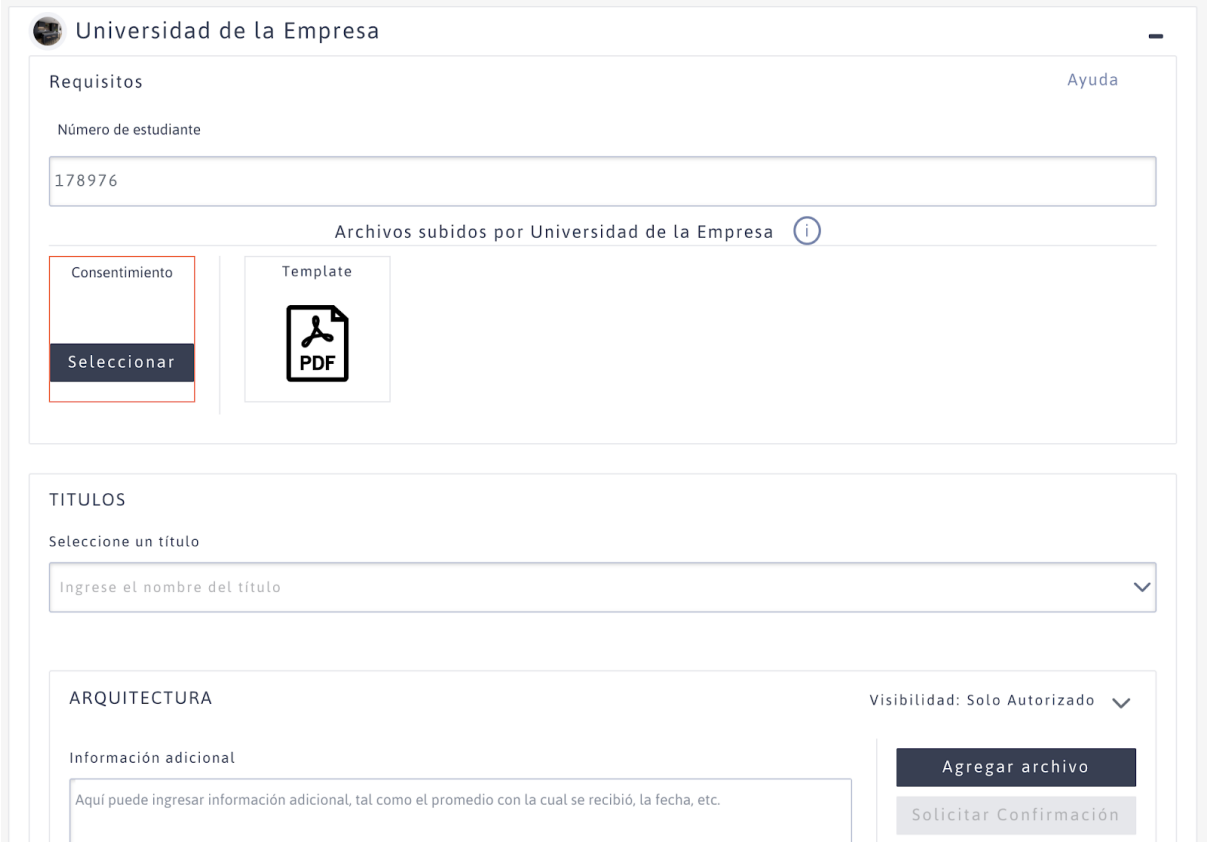


Imagen 8: Carga de campos requeridos por la institución

Finalmente tiene la posibilidad de agregar alguna información adicional que considere apropiada (por ejemplo, promedio de calificaciones). Al terminar pulsa el botón "Solicitar Confirmación" y la solicitud es enviada a la institución académica.

A partir de este momento la solicitud figura como pendiente, a la espera de la revisión por parte de la institución académica.

Cuando la institución académica revise la solicitud, le llegará un mail al graduado indicando el resultado de la revisión. En caso de haber sido aprobada, se habilita una opción en su perfil llamada "Compartir perfil".

De esta forma se genera un link que el graduado puede compartir con cualquier persona para que vean su diploma validado en la aplicación, sin necesidad de crearse un usuario en la aplicación.

Imagen 9: Perfil público del graduado

Institución académica

Al igual que los graduados, el responsable de una institución académica puede acceder al sistema usando sus credenciales, en este caso provistas por el administrador del sistema. También para el ingreso de una institución académica se aplica la autenticación de 2 factores por lo que recibirán por mail un token en cada intento de ingreso.

Una vez dentro del sistema, se despliega el perfil de la institución previamente cargado por el administrador. El responsable de la institución académica es libre de realizar las modificaciones que crea pertinentes, tanto en cuanto a los datos de la institución como de las carreras que se dictan o los requisitos que se requiere que completen los graduados.

En el panel de la izquierda puede accederse al listado de solicitudes pendientes de revisar.

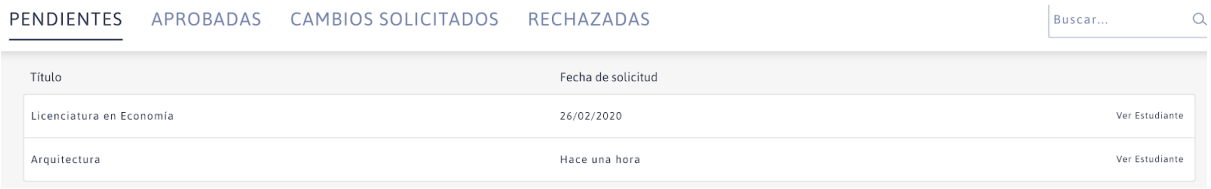


Imagen 10: Listado de solicitudes pendientes para una institución

Una vez revisada la solicitud, existen 3 posibilidades: aprobar la solicitud, solicitar un cambio, o rechazar la solicitud.

En caso de estar todos los datos correctos, coincidentes con la información registrada en la base de datos de la institución, entonces se selecciona la opción "Aprobar solicitud" y el diploma del graduado queda validado.

En caso de existir algún error o imprecisión en los datos (nombre mal escrito, calificación incorrecta, etc) es posible seleccionar la opción "Solicitar un cambio" donde da la opción de incluir un mensaje para indicar al graduado cual es el cambio que debe hacer para que su solicitud sea aprobada.

Finalmente, en caso de que el diploma presentado sea claramente falsificado u otras sospechas de fraude, entonces se debe seleccionar la opción "Rechazar solicitud" agregando un mensaje explicando la causa del rechazo. En este caso se bloquea el usuario de la persona que emitió la solicitud, y se envía una alerta al administrador para que se investigue un posible intento de fraude en la plataforma.

Diagrama de flujo

Se incluye a continuación un diagrama mostrando los distintos flujos de la aplicación involucrando a los distintos tipos de usuario.

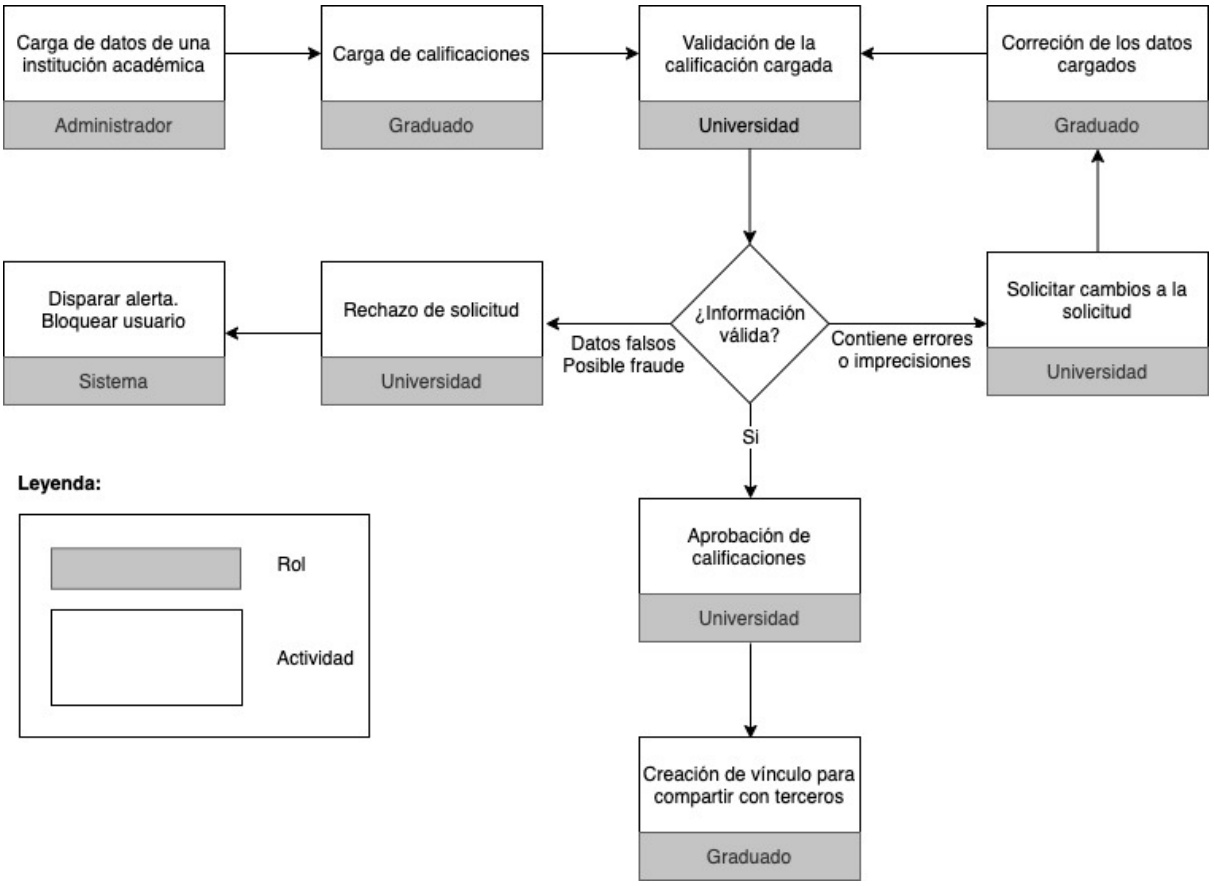


Imagen 11: Diagrama de flujo de la aplicación

3. Marco metodológico

En el presente capítulo se detalla la metodología que el equipo seleccionó para la realización del proyecto, así como también los marcos de trabajo utilizados y algunas de las principales decisiones que fueron tomadas.

3.1. Características del proyecto y contexto inicial del equipo

Antes de seleccionar el ciclo de vida del proyecto, fue necesario reflexionar acerca de las características del proyecto y el equipo.

El proyecto es una prueba de concepto

El proyecto es una prueba de concepto realizada para la empresa Effectus, con el objetivo de aprender y validar el uso de blockchain, y comprender de mejor manera cuáles son los beneficios que conlleva en cuanto a la seguridad de un sistema, y cuáles son las desventajas a la hora de aplicarla. En particular, se escogió el dominio de registro, validación, y consulta de diplomas académicos para evaluar su aplicabilidad. Esto presentó un desafío adicional, ya que además de tener el objetivo de desarrollar un sistema útil y novedoso, se debió aprender y validar una nueva tecnología. Además, al ser una prueba de concepto, no se contó con un cliente que tenga conocimiento en la tecnología a utilizar, por lo que debió existir una fuerte presencia de la investigación a lo largo del proyecto. El uso de una tecnología desconocida también agregó incertidumbre al proyecto.

El equipo está conformado por dos integrantes

A diferencia de la mayoría de equipos de proyectos, el equipo está compuesto únicamente por dos integrantes. Esto se debió tomar en cuenta en varios aspectos del proyecto, como por ejemplo al momento de asignar roles, ya que estos estarían distribuidos solamente en dos miembros del equipo. Además, en caso de presentarse la indisponibilidad de un miembro del equipo, el restante debía asumir todas las responsabilidades asociadas al proyecto. Tener un equipo de sólo dos

integrantes también significó que la coordinación y comunicación dentro del equipo se llevó a cabo con mayor facilidad.

El equipo no posee *expertise* en el dominio

Ningún miembro del equipo poseía conocimiento real acerca del dominio del problema. Esto implicó que sea necesario una etapa inicial en la cual se deba realizar un esfuerzo por comprender el dominio del problema, las soluciones existentes, y los distintos interesados para poder diseñar un producto que se ajuste a sus necesidades.

El equipo no contó con un conjunto inicial de requerimientos

De la mano del punto anterior, el equipo no recibió un conjunto de requerimientos por parte del cliente. Esto tuvo como consecuencia la necesidad no sólo de interiorizarse con el dominio en etapas iniciales, sino también de realizar un relevamiento completo de los requerimientos de la solución, entrevistando a potenciales usuarios de todos los grupos, ya sea estudiantes, empleadores, o integrantes de instituciones educativas. Esto también implicó que no sea posible comenzar con el desarrollo del sistema inmediatamente al inicio del proyecto, sino que se pudo comenzar con el desarrollo recién cuando se contó con un conjunto de requerimientos lo suficientemente definidos para guiarlo. Finalmente, al no poseer conocimiento del dominio, era claro que una vez que se cuente con requerimientos, estos serían propensos al cambio.

El equipo se puso como objetivo desarrollar un producto de calidad

Como fue mencionado en la sección 1.4 Objetivos, el equipo estableció como objetivo el desarrollo de un producto de calidad que sea de utilidad a los usuarios. Para ello, fue necesario utilizar marcos de trabajo que se adapten a las necesidades del equipo de aprender acerca del dominio y que también permitan el desarrollo a través de un proceso medible, eficaz, y ordenado.

3.2. Metodología y ciclo de vida

Teniendo en cuenta las características del proyecto, se prosiguió a seleccionar la metodología y ciclo de vida del mismo. Se ponderaron tanto los modelos tradicionales como los ágiles. En primer lugar, se descartaron los modelos tradicionales, como el cascada, debido a que entre las principales restricciones de estos modelos se encuentra la de contar con requerimientos estables, algo de lo cual este proyecto carece. Esto llevó al equipo a inclinarse más hacia los ciclos de vida de desarrollo ágiles. Los ciclos de vida ágiles no requieren requerimientos estables o entregables definidos para iniciar, y se espera que la mejor solución se presente durante el proyecto, como resultado del feedback de usuarios y cliente.

Una vez seleccionado el tipo de ciclo de vida del proyecto, se debió considerar las necesidades del proyecto para elegir el marco de trabajo a utilizar. Debido al desconocimiento del dominio y las tecnologías a utilizar, no era posible iniciar con el desarrollo del sistema desde las primeras etapas del proyecto, sino que era necesaria una fase inicial en la cual primara la investigación. El objetivo de esta fase sería comprender el dominio de mejor manera con el fin de obtener un conjunto de requerimientos lo suficientemente detallados como para dar paso al desarrollo del sistema. En ese momento fue claro que el proyecto contaría con dos etapas: una etapa inicial en la cual sea más marcado el proceso de relevamiento de requerimientos, y una etapa posterior de desarrollo.

3.2.1. Dual Track Agile

Dentro de los marcos de trabajo ágiles, se decidió utilizar el Dual Track Agile. Este define dos ciclos principales para obtener *feedback*:

- El ***discovery cycle*:** valida el valor y la factibilidad de las ideas, creando prototipos en el camino.
- El ***delivery cycle***: entrega el valor de una idea ya validada.

Es posible encontrar una descripción más detallada del Dual Track Agile en el anexo 12.3. Descripción del Dual Track Agile.

Este marco de trabajo fue elegido debido a que se adapta adecuadamente a las necesidades del proyecto. Como fue mencionado anteriormente, el proyecto se dividió en dos fases. En la primer fase se relevaron los requerimientos a través del *discovery cycle*, utilizando prototipos para obtener feedback de forma veloz. En la segunda fase se utilizó el *delivery cycle*, construyendo sobre las ideas ya validadas. Se definieron los siguientes objetivos para cada una de los ciclos:

- ***Discovery cycle:*** relevar y validar tanto requerimientos como arquitectura del sistema en cuestión, con el objetivo de obtener un conjunto de historias de usuario lo suficientemente detallada como para guiar el equipo hacia el desarrollo de un M.V.P (producto mínimo viable), es decir, un producto con suficientes características para satisfacer a los clientes iniciales, y proporcionar retroalimentación para el desarrollo futuro.

- ***Delivery cycle:*** desarrollar las ideas ya validadas, entregando valor a cliente y usuarios obteniendo feedback valioso, así como también validar las funcionalidades desarrolladas.

3.2.2. Adaptación

A continuación se detalla cómo fue realizada la adaptación del proceso de Dual Track Agile para el proyecto en cuestión.

3.2.2.1. Etapa inicial

Como fue explicado anteriormente, el *discovery cycle* definido en el *dual track agile* permite obtener feedback de forma rápida, sin necesidad de desarrollar funcionalidades. Esto resultó particularmente útil en la etapa inicial del proyecto, ya que brindó la posibilidad de comprender de mejor manera tanto los requerimientos como los potenciales usuarios y clientes, y evitó la inversión de tiempo y recursos en desarrollar funcionalidades de poco valor o no deseadas. Por esta razón, se definió

una etapa inicial del proyecto, en la cual se hizo hincapié en el *discovery cycle*, y se asignó menos tiempo para desarrollo.

Esta etapa concluyó con la primera revisión. Para esa instancia, se contó con un conjunto de historias de usuario que fueran suficientemente concretas como para comenzar a guiar al equipo hacia el desarrollo del M.V.P.

Dicho entregable tomó la forma de un ***user story map***. *User story mapping* es una técnica que consiste en organizar el *product backlog* en dos dimensiones con el objetivo de construir un *roadmap*. Este mapa se compone de dos ejes, en uno de ellos están los *releases* (vertical) y en el otro las funcionalidades (horizontal).

3.2.2.1.1. Marco de trabajo a utilizar: Kanban

En esta etapa se utilizó un ciclo de vida evolutivo, ya que se desarrollaron prototipos iniciales que luego de exponerse a los usuarios finales se fueron refinando de acuerdo al *feedback* obtenido.

Para la etapa de *discovery*, se evaluó utilizar Scrum o Kanban como marcos de trabajo. Por un lado, Kanban pone como tema central la visualización del trabajo, limitando el *work-in-progress* (WIP), y maximizando la eficiencia del flujo de trabajo. Los equipos Kanban se centran en reducir el tiempo invertido en llevar tareas de inicio a fin. Para ello, se utiliza un tablero Kanban. Por otro lado, los equipos de Scrum se comprometen a entregar incrementos de software funcional a través de intervalos establecidos llamados sprints. Su objetivo es crear *loops* de aprendizaje para recopilar e integrar rápidamente el feedback de los clientes. Los equipos Scrum adoptan roles específicos, crean artefactos especiales y realizan ceremonias regulares para que las cosas sigan avanzando.

A continuación se puede visualizar una tabla comparativa entre Scrum y Kanban, dos marcos de trabajo diseñados para asistir a los equipos a adherirse a los principios ágiles a medida que realizan el trabajo. [7]

	Scrum	**Kanban**
Cadencia	Duración fija (ejemplo: 2	Flujo contínuo

	semanas)	
Metodología de release	Al final de cada sprint	Entrega continua
Roles	Roles definidos: product owner, scrum master, y equipo de desarrollo	Sin roles requeridos
Filosofía en relación al cambio	No se deberían introducir cambios al sprint en curso durante el sprint	Es posible introducir cambios en cualquier momento

Tabla 5: Comparativa de Scrum y Kanban

El equipo optó finalmente por usar Kanban durante la etapa de discovery. En primer lugar, permite comenzar a trabajar sin la necesidad de invertir tiempo en definir roles, ceremonias, o artefactos. Además, durante esta etapa se contaba con una alta probabilidad de que ocurran cambios en las tareas a realizar, debido al poco conocimiento que se poseía del dominio. Con cada pieza de conocimiento que se fue adquiriendo, era posible que se definieran nuevas tareas de mayor prioridad que las ya existentes. Finalmente, en esta etapa no se cuenta con información suficiente como para tener *time-boxes* precisas definidas, por lo que Scrum no parecía ser un marco de trabajo adecuado.

Kanban ofrece las siguientes ventajas al equipo:

- **Visibilidad del trabajo**: el uso de paneles Kanban como fuente de información, además de los otros méritos del método, ofrece una visión holística del progreso, los cuellos de botella, los impedimentos, los bloqueos y las ineficiencias del proceso a simple vista.

- **Mejora en la colaboración:** las transiciones de elementos de trabajo entre diferentes columnas en los tableros de Kanban ofrecen oportunidades para el descubrimiento del conocimiento, la colaboración, la comunicación y, lo más importante, el compromiso y la participación de todos los integrantes del equipo.

- **Reducción de actividades de poca utilidad:** Kanban es un sistema que mantiene una cantidad confiable de ideas de alta calidad entregadas justo a tiempo, al tiempo que elimina el trabajo innecesario.

- **Mejora en la moral del equipo:** permite al equipo ver los frutos de su propio trabajo como progreso visualizado en el tablero Kanban. [8]

En cuanto a las herramientas, se utilizó la herramienta *Trello* para mantener un tablero con todas las tareas pendientes, en progreso, y finalizadas, y de esta forma poder monitorear de mejor manera qué cosas quedan aún por hacer.

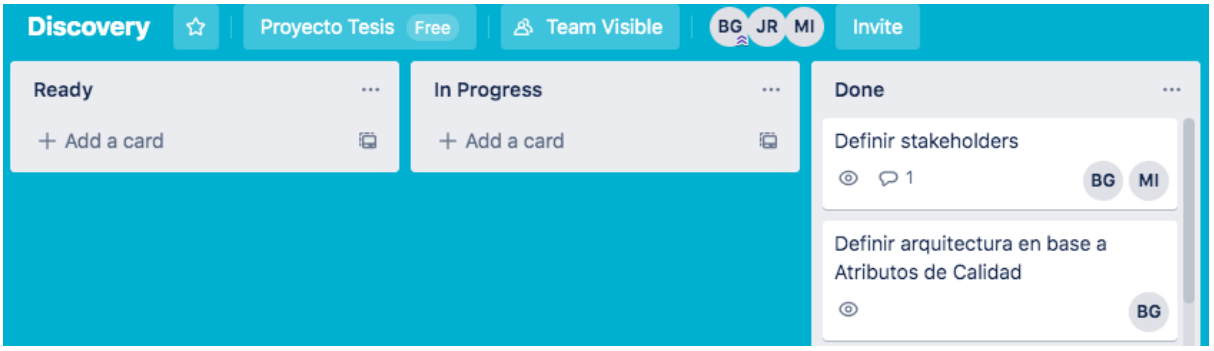


Imagen 12: Captura del tablero Kanban durante el proyecto

Durante esta primera etapa, se mantuvieron reuniones semanales de planificación. Dichas reuniones tuvieron como objetivo coordinar y validar los esfuerzos realizados por los integrantes del equipo en el transcurso de la semana anterior, así como también planificar el esfuerzo de la semana siguiente. Aquellas tareas que se consideren como finalizadas se clasificaron como *Done*. Por otro lado, las nuevas tareas especificadas para la semana siguiente eran agregadas a la columna *Ready*.

3.2.2.1.2. Actividades

- Standup semanal cada domingo 18 horas entre los miembros del equipo. Se discuten avances de cada uno, tareas pendientes, y problemas encontrados.

- Reunión semanal con el tutor del proyecto cada martes 19:30 horas. Se discuten avances de la semana, mejoras, y se planifican las tareas para la próxima semana.

- Reunión retrospectiva cada 2 semanas. Se discute qué cosas han funcionado bien, cuales se pueden mejorar y que se ha aprendido. Se definen planes de mejora, y nuevos acuerdos de trabajo.

3.2.2.2. Etapa de desarrollo

Una vez que se contó con un conjunto de historias de usuario viables y de valor en forma de un *user story map*, se pasó a la siguiente etapa del proyecto en la cual fue más marcada la presencia del *delivery cycle*. Esta etapa se centró en el desarrollo de las historias de usuario obtenidas durante la primera etapa. En esta etapa se utilizó un ciclo de vida incremental, y se obtuvo feedback a través del valor entregado a los clientes y usuarios.

Esta etapa contó con tres entregables principales:

1. **MVP - primer release:** el mismo fue entregado al llegar la segunda revisión. El objetivo de este release fue completar la creación y edición del perfil de las instituciones. Esto comprende el registro de una institución, así como toda la información asociada a las mismas, desde imagen de perfil, número telefónico y dirección, hasta los distintos títulos que ofrece. De esta forma, fue posible realizar una validación del mismo con un prototipo funcional.

2. **MVP - segundo release:** el mismo fue realizado al finalizar el séptimo *sprint*, a fines de octubre. El objetivo de este release fue completar el proceso de ingreso y aprobación de calificaciones. Para ello fue necesario desarrollar las funcionalidades correspondientes al ingreso de datos personales dentro del perfil del egresado, ingreso de calificaciones, solicitud de verificación a las instituciones académicas, y procesamiento de dichas solicitudes por parte de las instituciones académicas. Desarrollar estas funcionalidades le permitió al equipo realizar una validación de este proceso con un prototipo funcional.

3. **MVP - tercer release:** el mismo fue entregado al llegar la tercera revisión (diciembre). El objetivo de este release fue el desarrollo del flujo completo del sistema, siguiendo los siguientes pasos:

    a. Registro de institución junto con toda la información asociada a la misma.

    b. Registro de egresado junto con toda la información asociada al mismo.

    c. Registro de título, dando también la posibilidad de adjuntar archivos al mismo, como por ejemplo una imagen del diploma.

    d. Solicitación de verificación del título a la institución académica por parte del egresado.

    e. Visualización de solicitudes de verificación por parte de la institución.

    f. Visualización de perfil de egresado por parte de la institución.

    g. Aprobación, rechazo, o solicitud de cambio de un título subido por un egresado por parte de la institución.

    h. Generación de link para compartir el perfil de un egresado, por parte del mismo egresado.

    i. Visualización de perfil de egresado con todos los títulos verificados, por parte de un tercero con quien fue compartido dicho link.

    De este modo, fue posible la validación del sistema completo.

4. **Documentación final.** La misma será entregada al llegar la entrega final. Esta entrega tiene dos objetivos principales: por un lado, presentar una documentación completa que describa el transcurso del proyecto de inicio a fin, y por otro lado, producir un documento con los

aprendizajes y hallazgos realizados durante el proyecto acerca de la aplicabilidad de blockchain al dominio de la validación de diplomas, para completar la transferencia de conocimiento al cliente.

3.2.2.2.1. Marco de trabajo a utilizar: Scrum

El objetivo de esta sección es realizar una introducción a la metodología ágil Scrum, explicando los motivos por los cuales fue adoptada, y listar las ventajas de la misma.

Scrum es un marco de trabajo por el cual las personas pueden abordar problemas complejos adaptativos, a la vez que entregar productos del máximo valor posible productiva y creativamente. Ha sido usado para gestionar el trabajo en productos complejos desde principios de los años 90. Scrum no es un proceso, una técnica o método definitivo. En lugar de eso, es un marco de trabajo dentro del cual se pueden emplear varios procesos y técnicas. Es posible encontrar una explicación detallada de este marco de trabajo en el anexo 12.4. Marco de trabajo Scrum. [9]

En la etapa de desarrollo, se utilizó el marco de trabajo Scrum con algunas adaptaciones, de modo de poder controlar, medir, y mejorar el proceso de construcción del software. Scrum fue elegido debido a que en esta etapa se contó con trabajo que pudo ser reducido a iteraciones contenidas en *timeboxes* (*sprints*). Además, se contó con un conjunto de requerimientos lo suficientemente estables como para no esperar cambios en los mismos durante el transcurso de un *sprint*.

3.2.2.2.1. Adaptaciones de Scrum al proyecto

Uso de roles

El equipo utilizó los roles definidos por Scrum, aplicándolos de la misma forma que el marco de trabajo los establece, aunque con algunas excepciones. En primer lugar, al ser un equipo de sólo dos integrantes, cada integrante del equipo debió asumir varios roles. Esto tuvo como consecuencia que tanto el Scrum master como el dueño de producto formarán parte del equipo de desarrollo, ya que era necesario de ambos integrantes del equipo para producir incrementos significativos en cada sprint.

Finalmente, a pesar de que cada rol fue asignado a un integrante, ambos integrantes debieron estar familiarizados con todos los roles. Esto se debe a que en caso de que uno no esté disponible por alguna razón puntual, el otro debió actuar como respaldo para no frenar el avance del proyecto.

Adecuación de las daily scrums

La frecuencia de estas ceremonias se debieron adaptar por dos motivos. En primer lugar, debido a que la frecuencia diaria está pensada para equipos que le dedican una jornada laboral diaria al proyecto, lo cual al ser un proyecto de tesis no fue el caso, por lo que no se realizaba la misma cantidad de trabajo diario que en ambientes laborales. En segundo lugar, la frecuencia fue ajustada debido a que, al ser solamente dos integrantes, el equipo estaba en comunicación constante, por lo que no fue necesaria una reunión diaria para coordinar los esfuerzos.

Debido a los factores mencionados previamente, se mantuvieron reuniones semanales con el mismo propósito de las daily scrums.

3.3. Roles

Previo a referirse a los roles que cada integrante asumió, cabe destacar el hecho de que el equipo cuenta solamente con dos integrantes, por lo que ambos integrantes debieron estar familiarizados con todos los roles. Sin embargo, cada rol fue asignado a un integrante, mientras el otro actuó como respaldo en caso de que el integrante designado no pudiera desempeñar las actividades asociadas a su rol por alguna razón puntual.

- ***Discovery team:*** Bruno Gallo y Marcos Irisarri
- ***Developers*****:** Bruno Gallo y Marcos Irisarri
- **QA:** Marcos Irisarri
- **Gestión de la configuración:** Bruno Gallo
- **Tech Lead / Arquitecto:** Bruno Gallo

- **Scrum master:** Marcos Irisarri
- **Product owner:** Marcos Irisarri

Se distribuyeron los roles de forma que quede marcado en qué se especializa cada uno de los integrantes del equipo. Por un lado, Marcos posee un mayor conocimiento del dominio. Esto será vital no sólo para desempeñar su rol de *product owner*, sino también para desempeñar actividades de *quality assurance*. El conocimiento del dominio fue esencial para reconocer cuáles historias de usuario agregan valor al producto, y también para reconocer bugs en las funcionalidades desarrolladas.

Por otro lado Bruno posee mayor conocimiento técnico, lo cual le permitió desempeñar sus roles de *tech lead*, determinando cuáles historias son viables técnicamente, y cuáles deberán ser descartadas. Del mismo modo, dicho conocimiento fue de gran utilidad para diseñar la arquitectura del sistema, y para gestionar la configuración del mismo.

3.4. Conclusiones

En primer lugar, luego de casi finalizado el proyecto, es posible concluir que el *dual track agile* y su adaptación del mismo fueron elecciones apropiadas para el tipo de proyecto. Dado el casi nulo conocimiento que el equipo poseía acerca del dominio, elegir un marco de trabajo que permita comenzar el proyecto con una etapa que se centre en la investigación como lo fue el *discovery cycle* le brindó al equipo la oportunidad de interiorizarse en varios aspectos del dominio, algo que a la larga desembocó en lograr un producto de mejor calidad. Además, también pareció ser una decisión acertada combinar el *discovery cycle* con el marco de trabajo Kanban, puesto que este último permitió al equipo concentrarse en el trabajo a realizar, sin la necesidad de invertir tiempo en asignar roles, o en varias ceremonias. Kanban permitió que el equipo avance de forma veloz, logrando cumplir el objetivo de adquirir conocimientos, algo que era crucial para el éxito del proyecto.

Dual track agile también fue una buena elección porque una vez adquirido el conocimiento suficiente como para comenzar el desarrollo, el equipo pudo hacer hincapié en el *delivery cycle*, construyendo sobre lo aprendido y validado durante el *discovery cycle*. El marco de trabajo Scrum pareció adaptarse correctamente a esta etapa del proyecto. La principal ventaja que el equipo le encontró a este marco de trabajo fue la gran capacidad que este provee para incorporar cambios a medida que ocurren. Como se verá más adelante, durante el desarrollo del proyecto surgieron nuevos requerimientos, y el uso de Scrum permitió que el equipo se adapte a ellos aceptándolos, de forma tal que de todos modos fue posible cumplir con los objetivos de cada *release*.

4. Ingeniería de requerimientos

El objetivo de este capítulo es la presentación, explicación y justificación del proceso de ingeniería de requerimientos aplicado durante el proyecto.

Como fue mencionado en el capítulo 3, los integrantes del equipo no contaban con conocimientos detallados acerca del dominio del problema, por lo que fue de gran importancia un proceso de ingeniería de requerimientos que permita entender el problema y el dominio con el objetivo de ganar un mayor entendimiento acerca del mismo. El proceso de ingeniería de requerimientos contó con dos etapas principales, una de relevamiento y la otra de validación y especificación. Cada una de estas etapas tuvo el objetivo de generar entregables con distintas funciones.

En la primera etapa, hubo una mayor presencia del relevamiento de requerimientos, por lo que el equipo utilizó varias técnicas de relevamiento de requerimientos para comprender mejor el dominio y las necesidades de los distintos interesados. El objetivo de esta primera etapa fue la creación de una propuesta de valor que tomó la forma de un *value proposition canvas*. De este modo, fue posible identificar las distintas tareas, dolores, y ganancias de cada tipo de usuario, e idear un producto que facilite las tareas de los usuarios, al mismo tiempo que alivie los dolores y genere ganancias para los usuarios.

La segunda etapa se centró en la validación de los hallazgos realizados durante la primera etapa, y en la construcción de un entregable que pueda guiar al equipo en la etapa de desarrollo posterior. Dicho entregable tomó la forma de un *user story map*.

4.1. Etapa de relevamiento

Durante esta etapa, se utilizaron distintas técnicas de relevamiento de requerimientos con el fin de crear una propuesta de valor utilizando el *value proposition canvas*. El diseño del *value proposition canvas* tiene como propósito el desarrollo de productos que se ajustan a las necesidades reales de los clientes y

usuarios. En él, el foco está en el usuario y sus requisitos. El *value proposition canvas* trata de descubrir por qué el cliente necesita un producto o servicio, qué puede percibir el cliente como un valor adicional y qué le resulta molesto o desventajoso.

La principal ventaja del *value proposition canvas* es que ayuda a comprender sistemáticamente lo que quieren los clientes y crea productos y servicios que se adaptan de mejor manera a sus necesidades. Recopila información del cliente de una manera sencilla que se adapta a sus necesidades y requisitos, lo que permite un diseño más eficaz de los modelos de negocio. Eventualmente, esto conducirá a la rentabilidad y no se perderá tiempo en desarrollar ideas en las que los clientes puedan no estar interesados. Mediante el uso de *value proposition canvas*, los equipos pueden identificar las necesidades de los clientes de forma visual y estructurada, lo que tiene como resultado el diseño de una propuesta de valor que refleja perfectamente los requisitos de sus clientes, los aliviadores de dolores y los creadores de ganancias. [10]

El *value proposition canvas* tiene dos lados. Con el perfil del cliente, se gana un mayor entendimiento de los clientes y usuarios. Con el mapa de valor, se describe cómo se tiene pensado crear valor para esos clientes y usuarios. Se logra un *fit* cuando el mapa de valor concuerda con el perfil del cliente — cuando los productos y servicios generan aliviadores de dolores y creadores de ganancias que encajan con aquellas tareas, dolores y ganancias que son importantes para el cliente.

4.1.1. Técnicas de relevamiento de requerimientos

Con el objetivo de obtener suficiente información acerca de cada tipo de usuario como para poder construir el perfil de usuario de cada uno de ellos, se utilizaron las siguientes técnicas de relevamiento de requerimientos:

Técnica	**Fundamentación**
Brainstorming	Provee un ambiente libre y abierto que alienta a todo el equipo a participar. Ideas peculiares y extravagantes

	son bienvenidas y se trabaja sobre ellas para alcanzar una solución. Ayuda a que todos los miembros del equipo se sientan parte de la solución.
Análisis de documentos	La revisión de la documentación de un sistema existente puede ayudar comprender el dominio del sistema, así como también los requerimientos del mismo. Además de realizar análisis de documentaciones de sistemas existentes, también fue necesario el análisis de documentos legales debido a la sensibilidad de los datos con la cual se está tratando.
Entrevistas	Uno de los métodos de investigación cualitativa más utilizado, consiste en la realización de preguntas previamente preparadas a los distintos interesados con el fin de obtener información que luego será usada para escribir requerimientos formales. Es una buena técnica para investigar ciertos temas en profundidad, descubrir lo que las personas piensan y sienten sobre estos temas y lograr un ambiente donde puedan expresarse sobre ciertas cosas que no les gusten o que prefieran cambiar.
Encuestas	Correctamente diseñadas, las encuestas pueden ser útiles para relevar información de muchas personas de forma práctica y obtener datos estadísticos. Se debe evitar ambigüedad en las preguntas, dando en lo posible opciones para que las personas respondan.

Tabla 6: Técnicas de relevamiento utilizadas

4.1.1.1. Análisis de documentos

Se analizó la documentación del sistema *Checkdiploma* [11]. En la misma, se detalla el funcionamiento del sistema. Los elementos más relevantes que se extrajeron de la documentación fueron los siguientes:

- En el sistema se almacenan diplomas digitalizados en formato PDF.
- Para cada universidad, se crea una lista dentro del contrato inteligente.
- El proceso de almacenado de diplomas digitales funciona de la siguiente manera: para cada diploma que es subido a la plataforma web se calcula su valor *hash* y se publica dicho valor en la lista asociada a la universidad dentro del contrato inteligente. La plataforma también genera un token de seguridad con el que encripta el archivo pdf original y lo almacena en la plataforma *Google Cloud*. Este token de seguridad se muestra una sola vez y nunca se guarda. La universidad lo imprime y se lo entrega personalmente al alumno, al mismo tiempo que le entregan el diploma impreso.
- El sistema utiliza contratos inteligentes desplegados en la blockchain pública Ethereum.

También se analizó la documentación del estándar *Blockcerts* [12]. Se trata de un estándar abierto para crear, emitir, ver y verificar certificados basados en blockchain. El diseño inicial se basó en prototipos desarrollados en el *MIT Media Lab* y por *Learning Machine*. Para el desarrollo continuo, este proyecto de código abierto anima activamente a otros colaboradores a involucrarse. El *MIT Media Lab* no participa activamente en el desarrollo continuo.

Al igual que *Checkdiploma*, el estándar *Blockcerts* se basa en el almacenamiento del *hash* de las certificaciones en una blockchain, utilizándolo luego para verificar la integridad de las credenciales.

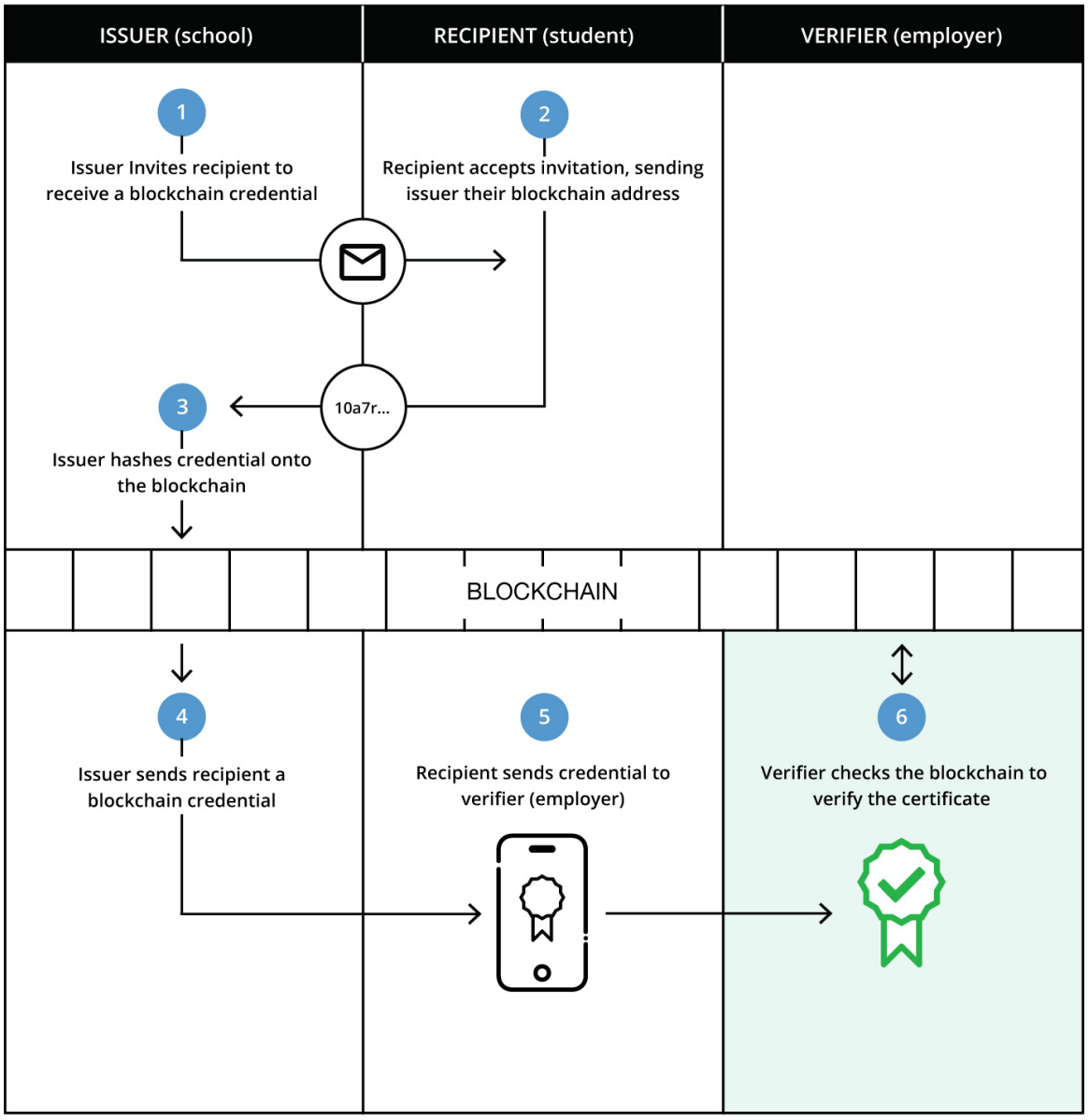


Imagen 13: Flujo de ejecución de *Blockcerts*

Finalmente, se analizó además la Ley N° 18.333, ley de protección de datos personales [5]. Esto se hizo en gran parte debido a la sensibilidad de los datos con la cuales se trató. A través de esta ley, el equipo se informó de varios aspectos legales que fueron de gran importancia en etapas posteriores a la hora de tomar decisiones. Los artículos más importantes que se extrajeron de dicha ley fueron los siguientes:

- **Artículo 8:** Los datos deberán ser eliminados cuando hayan dejado de ser necesarios o pertinentes a los fines para los cuales hubieren sido recolectados.

- **Artículo 9:** Principio del previo consentimiento informado.- El tratamiento de datos personales es lícito cuando el titular hubiere prestado su consentimiento libre, previo, expreso e informado, el que deberá documentarse.

- **Artículo 10:** Principio de seguridad de los datos.- El responsable o usuario de la base de datos debe adoptar las medidas que resultaren necesarias para garantizar la seguridad y confidencialidad de los datos personales. Dichas medidas tendrán por objeto evitar su adulteración, pérdida, consulta o tratamiento no autorizado, así como detectar desviaciones de información, intencionales o no, ya sea que los riesgos provengan de la acción humana o del medio técnico utilizado.

- **Artículo 13:** Derecho de información frente a la recolección de datos.- Cuando se recaben datos personales se deberá informar previamente a sus titulares en forma expresa, precisa e inequívoca:

    - La finalidad para la que serán tratados y quiénes pueden ser sus destinatarios o clase de destinatarios.

    - La existencia de la base de datos, electrónico o de cualquier otro tipo, de que se trate y la identidad y domicilio de su responsable.

    - El carácter obligatorio o facultativo de las respuestas al cuestionario que se le proponga, en especial en cuanto a los datos sensibles.

    - Las consecuencias de proporcionar los datos y de la negativa a hacerlo o su inexactitud.

    - La posibilidad del titular de ejercer los derechos de acceso, rectificación y supresión de los datos.

- **Artículo 15:** Derecho de rectificación, actualización, inclusión o supresión.- Toda persona física o jurídica tendrá derecho a solicitar la rectificación, actualización, inclusión o supresión de los datos personales que le corresponda incluidos en una base de datos, al constatarse error o falsedad o exclusión en la información de la que es titular.

4.1.1.2. Entrevistas y encuestas a usuarios

Cabe aclarar que las técnicas de relevamiento variaron según tipo de usuario con el que se trató. Cuando era posible acceder a un grupo de usuarios más amplio, como fue el caso de los egresados, se utilizaron encuestas, pues esta técnica permite capturar una muestra más representativa de una población más grande de forma más eficiente.

Por otro lado, en los casos en los que no se tenía acceso a una muestra tan grande, y cuando había una necesidad de acceder a información más profunda y específica, se utilizaron entrevistas, como fue el caso de los empleadores e integrantes de instituciones académicas. Además de esto, las entrevistas no sólo hacen que los entrevistados se sientan más cómodos expresando sus verdaderos sentimientos y opiniones, sino que también dan la oportunidad de capturar los matices y el lenguaje natural utilizado por los encuestados, obteniendo una perspectiva que no es posible obtener a través de encuestas [13]. Por esta razón, también se utilizaron encuestas para obtener información de egresados.

Las entrevistas realizadas a personal de recursos humanos fueron las siguientes:

- Alejandra Diaz - Nurse jefe del Instituto de Cardiología Intervencionista de Casa de Galicia
- Josefina Vidal - Supervisora en CPA Ferrere
- Lorena Quiroga - Gerenta de RRHH en CPA Ferrere
- Jean Pierre Valencia - Partner en TangoCode

- Paula Gallotti - Directora en DVelop

Las entrevistas realizadas a integrantes de instituciones académicas fueron las siguientes:

- Adriana Fernández y Victoria Gonzalez - Oficina Graduados de ORT
- Pablo San Nicolás y Gabriela Sánchez - Secretario de gestión académica y coordinadora de registro de documentación.

Imagen 14: Reunión con Adriana Fernandez y Victoria González

En cuanto a los egresados, se relevaron los datos utilizando principalmente encuestas, aunque también se le realizó una entrevista a Juan Palermo, egresado de ORT.

Para preparar las entrevistas a los distintos grupos de usuarios, se utilizó la técnica de *Design Thinking*. Al momento de realizar las entrevistas, también se realizaron *empathy probes*, otra técnica de *Design Thinking*. Las *empathy probes* fueron utilizadas con el fin de comprender de mejor manera los requerimientos del sistema

a desarrollar. Es posible visualizar el proceso de preparación con mayor detalle en el anexo 12.6. Preparación de entrevistas utilizando Design Thinking.

4.1.2. Resultados Obtenidos

4.1.2.1. Conclusiones generales

A través de las técnicas de relevamiento de requerimientos mencionadas previamente dirigidas a los distintos tipos de usuarios (empleadores y egresados), se llegó a las siguientes conclusiones:

- Desde el momento en que un estudiante aprueba el último examen hasta que recibe su diploma pasan al menos 3 meses en general. A la mayoría de los encuestados le parece poco eficiente este proceso.

Imagen 15: Resultado de encuesta a graduados

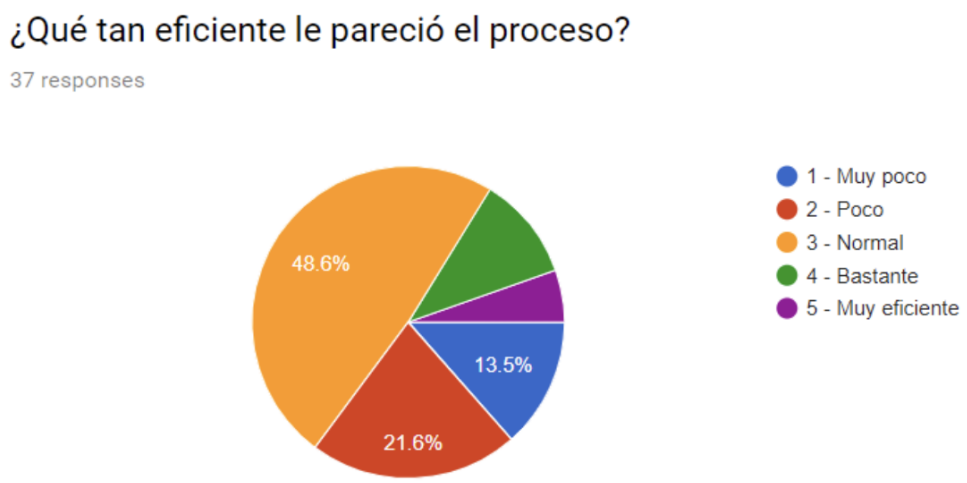


Imagen 16: Resultado de encuesta a graduados

- Los empleadores en general no verifican la validez de los títulos que los aplicantes dicen tener. Sólo algunos solicitan una copia o foto del mismo. Muchos se guían por referencias laborales o personales.

- La mayoría de los aplicantes no presenta el diploma en sus currículum vitae, aquellos que lo hacen presentan tanto copias digitales (foto) como físicas (fotocopia).

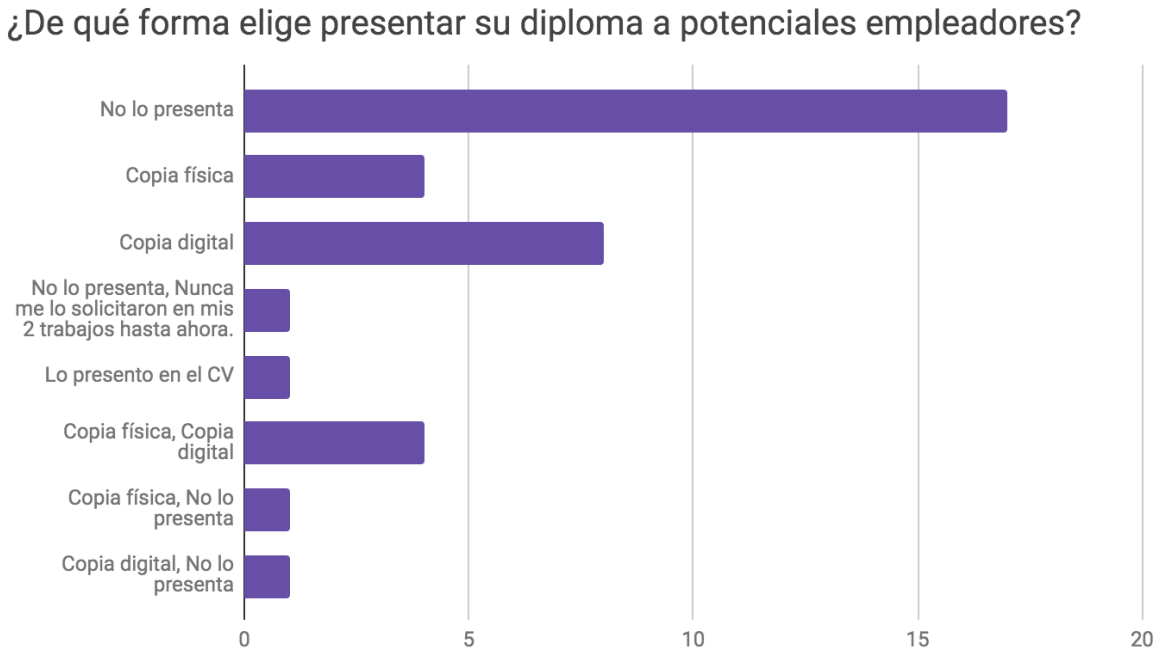


Imagen 17: Resultado de encuesta a graduados

- La mayoría de las personas que presentan su diploma a potenciales empleadores no toma ninguna medida para demostrar su autenticidad.

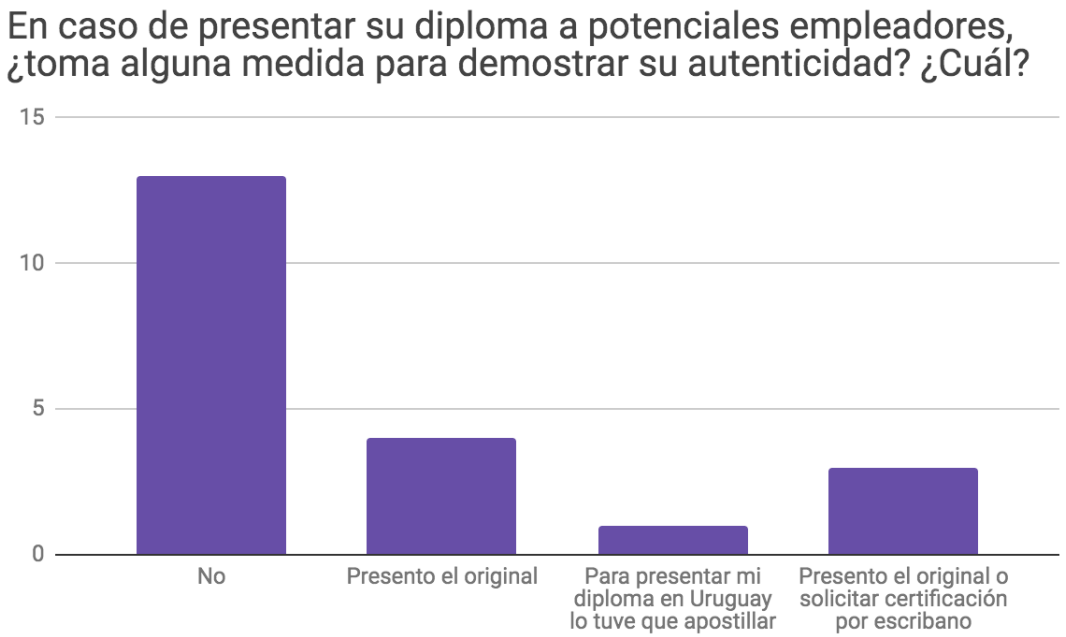


Imagen 18: Resultado de encuesta a graduados

- Los empleadores consideran que una herramienta útil a la hora de verificar calificaciones de los aplicantes agregaría valor a sus tareas de contratación de personal.

- En general, los egresados consideran que una herramienta que sus posibles empleadores puedan utilizar para verificar la validez de sus diplomas los diferenciaría de forma positiva frente a otros candidatos.

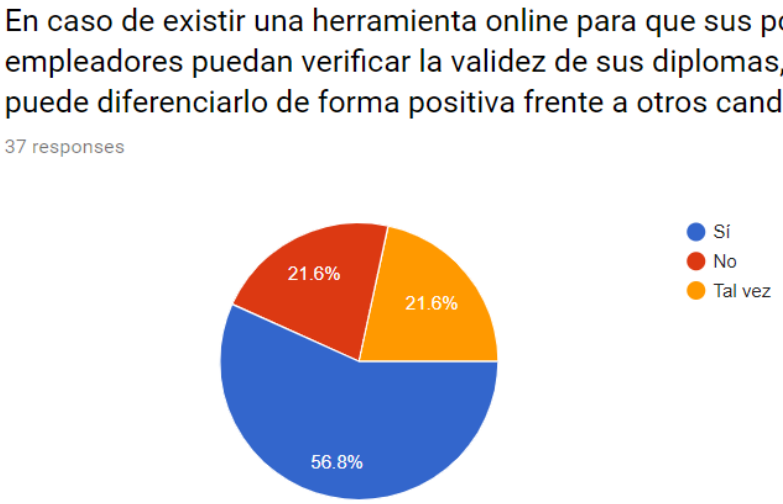


Imagen 19: Resultado de encuesta a graduados

- En caso que un empleador desee corroborar la validez de un título presentado por un candidato, debe ponerse en contacto con la universidad emisora del mismo. Es necesario presentar una carta firmada por el graduado autorizando a la universidad a revelar sus datos, debido a la ley N° 18.331 de protección de datos personales [5]. En caso de cumplirse todos los requisitos la universidad se limita a contestar si los datos presentados por el graduado son verídicos o no, no los corrige en caso de no ser correctos.

- No es adecuado el uso de firmas digitales para el sistema en cuestión. La firma digital, también conocida como firma electrónica, es una herramienta que permite firmar digitalmente con el mismo valor que la firma manuscrita. La cédula de identidad digital permite utilizar la firma digital, ya que contiene un chip con un certificado de firma digital protegido por el PIN elegido por el usuario al momento de retirarla [14]. Las firmas digitales tiene la desventaja de que para poder utilizarlas es necesario contar con un lector de tarjetas inteligente que se conecta a la computadora vía un cable USB [15]. Por otro lado, también existen los certificados digitales. Los certificados digitales son documentos que representan a personas físicas o jurídicas en el mundo digital-electrónico. En el caso de las personas físicas el certificado contiene datos como tipo de documento, nombres, apellidos, etc. Estos también fueron

descartados debido a su costo ($ 860 + IVA con 1 año de vigencia) [16].

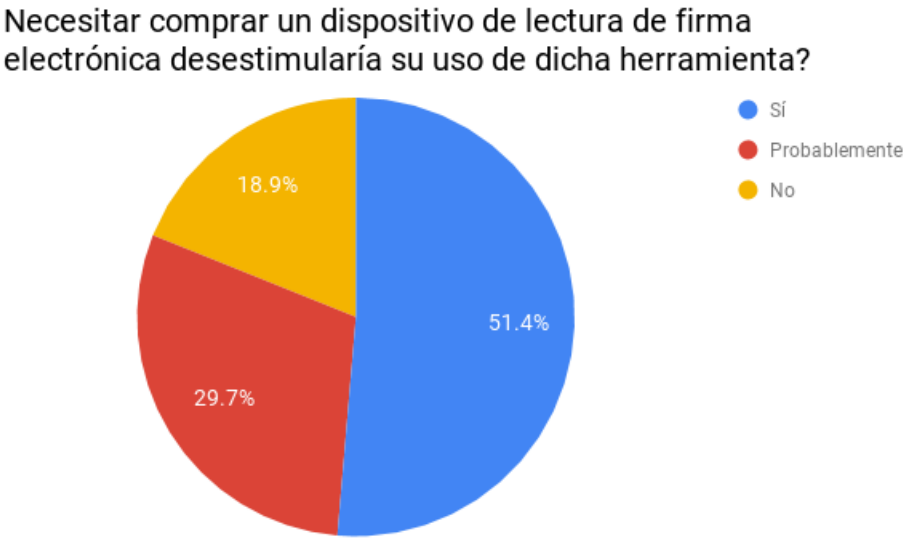


Imagen 20: Resultado de encuesta a graduados

Es posible visualizar las transcripciones de algunas entrevistas en el anexo 12.7. Transcripción de entrevistas.

4.1.2.2. Propuesta de valor

Además de las conclusiones mencionadas previamente, también se elaboró un perfil de cliente de cada uno de los tipos de usuarios que tendría el sistema: egresados, instituciones académicas, y empleadores. Para cada uno de ellos, se listaron las distintas tareas, dolores, y ganancias extraídas utilizando las técnicas de relevamiento mencionadas.

A continuación, se detallarán las tareas más importantes, los dolores más severos, y las ganancias más relevantes para cada tipo de usuario:

Empleadores

- Tareas:
    - Tareas funcionales: leer currículum vitaes, filtrar según calificaciones, y verificar referencias.

    - Tareas sociales: seleccionar candidatos competentes.
    - Tareas de apoyo: navegar portales de universidades.
- Dolores:
    - Resultados indeseables, problemas, y características: dificultad para contactar instituciones educativas (funcional).
    - Riesgos: candidatos que mienten sobre sus calificaciones, y contratar candidatos no calificados.
- Ganancias:
    - Ganancias requeridas: poder verificar escolaridades de estudiantes.
    - Ganancias esperadas: usabilidad del sistema y seguridad del sistema.
    - Ganancias deseadas: poder ver imágenes del diploma en la red.

Egresados

- Tareas:
    - Tareas funcionales: crear currículum vitae, obtener sello de autenticidad de documento, y crear carta firmada dando permiso a la institución de confirmar sus calificaciones.
- Dolores:
    - Resultados indeseables, problemas, y características: es molesto tener que ir hasta la institución educativa (dolor auxiliar), y también es molesto tener que ir hasta la fotocopiadora (dolor auxiliar).
    - Obstáculo: no poder obtener sello de autenticidad debido a que la institución educativa está cerrada.
- Ganancias:

- Ganancia requeridas: poder ofrecer forma de verificar calificaciones sin necesidad de moverse de su casa.
- Ganancias esperadas: usabilidad del sistema, seguridad del sistema, y poder mostrar imágenes de las calificaciones u otros archivos.

Instituciones académicas

- Tareas:
  - Tareas funcionales: recibir llamados de empleadores, verificar calificaciones en el sistema, y confirmar calificaciones a empleadores.
  - Tareas social: tener status de institución que se preocupa por sus alumnos.
  - Tarea emocional: sentirse seguro en brindarle herramientas a los egresados para que tengan éxito en su búsqueda de trabajo
- Dolores:
  - Resultados indeseables, problemas, y características: tener que corroborar varias veces las calificaciones de un mismo egresado (funcional).
  - Riesgos: personas que cometen fraude utilizando diplomas de la institución, dañando la reputación de la misma.
- Ganancias:
  - Ganancias requeridas: poder verificar las calificaciones de los estudiantes, y poder verificar la identidad del estudiante.
  - Ganancias esperadas: usabilidad del sistema, seguridad del sistema, y poder determinar qué datos los estudiantes deben ingresar.

Una vez realizados los perfiles de los usuarios, se utilizaron distintas técnicas, tales como el *brainstorming*, para idear un producto que se adapte a las necesidades de los usuarios, intentando lograr el *fit* de la mejor forma posible. Es posible ver los perfiles de los usuarios, así como el mapa de valor en el anexo 12.6. Value proposition canvas.

4.2. Etapa de validación y especificación

Como el título de esta sección lo infiere, el objetivo inicial de esta etapa fue validar la propuesta de valor generada en la etapa anterior. Una vez que se contó con una propuesta de valor ya validada, el siguiente paso fue generar un entregable que permita especificar dicha propuesta en forma de requerimientos, con el fin de guiar al equipo en la etapa de desarrollo. Como fue mencionado anteriormente, este entregable fue un *user story map*.

4.2.1. Validación

Tanto la etapa de relevamiento como la de validación y especificación se corresponden con el *discovery cycle* mencionado en el capítulo anterior. Como se explicó en ese capítulo, durante esta etapa se utilizaron prototipos para obtener *feedback* de los distintos usuarios, y de este modo poder validar o ajustar las ideas generadas en la etapa de relevamiento. En cuanto al tipo de prototipos a utilizar, se decidió utilizar prototipos desechables, pues estos se adaptan de mejor manera al fin que el equipo les dio, que fue la validación. La fundamentación de esta elección se encuentra en el anexo 12.8. Análisis de tipo de prototipado.

Se realizaron prototipos del perfil de egresado, incluyendo sus varias vistas: vista de egresado, vista de institución académica, y vista de empleador. A través del uso de estos prototipos desechables se logró no sólo realizar ajustes de ideas en cuanto a funcionalidades ofrecidas, sino que también surgieron cambios con el objetivo de mejorar la usabilidad del sistema. Cada vista fue validada con el tipo de usuario que le correspondía. La vista del empleador fue validada con Josefina Vidal, de la consultora CPA Ferrere (una de las entrevistadas durante la etapa de relevamiento).

La vista de institución académica fue validada con la experta en el dominio María Noel Severi, integrante de la academia ULearn. Por otro lado, la vista del egresado fue validada con Juan Palermo, egresado de ORT, y otro de los entrevistados. También se realizó una validación de todas las vistas con Aric Drotts, quien desempeña funciones de QA en la empresa TangoCode, en la cual trabaja uno de los integrantes del equipo. Esta validación fue realizada con la usabilidad en mente.

Sin embargo, antes de que llegara el momento de validar el perfil de la institución académica, se llegó a la conclusión de que María Noel Severi no sería más adecuada como experta en el dominio en cuestión. Debido a esto, no se logró validar dicho perfil en etapas iniciales. Esto obligó al equipo a buscar fuentes alternativas donde validar esta sección del sistema. A pesar de esto, no fue fácil encontrar dichas fuentes, por lo que el equipo se vio obligado a avanzar teniendo esto en cuenta. En etapas posteriores, se logró validar esta sección del sistema, y a raíz de dicha validación surgieron nuevos requerimientos que se debieron incorporar y priorizar. Esta dificultad para encontrar interesados con quienes validar los prototipos se ve reflejada como un riesgo que se materializó. Se analiza en mayor profundidad en la sección 7.3.1. R1 - Difícil acceso a la información.

El proceso de prototipado se encuentra explicado en mayor detalle en el anexo 12.9. Prototipado.

4.2.2. Especificación

Una vez que se contó con un conjunto de ideas validadas en la mejor medida posible, se pasó al proceso de especificación. El objetivo de esta etapa fue generar un *user story map* lo suficientemente conciso como para comenzar a guiar al equipo en la etapa de desarrollo.

4.2.2.1. Técnicas de especificación de requerimientos

De acuerdo con el marco metodológico ágil elegido, los requerimientos serán tratados de forma distinta que en los marcos metodológicos tradicionales. A la hora de especificar requerimientos, se tendrán los siguientes principios en cuenta:

- Los requerimientos alcanzarán el nivel de detalle necesario para ser desarrollados en el momento que serán desarrollados.
- Los *features* del producto serán negociados a través de conversaciones a lo largo del proyecto.
- Al esperarse el cambio, los requerimientos presentan un grado de libertad que permite su manipulación con el fin de alcanzar objetivos de negocio.
- Al esperarse que nuevos requerimientos emerjan, se evitará invertir demasiado tiempo en la especificación de requerimientos, ya que es posible que estos sean descartados más adelante con el fin de desarrollar otros de mayor valor.

En lugar de contar con requerimientos detallados de antemano, se usarán *placeholders* para los requerimientos llamados *product backlog items* (PBIs), los cuales serán discutidos y refinados de forma progresiva a lo largo del proyecto. Cada ítem representa un valor de negocio deseable. Los PBIs tendrán el formato de historias de usuario.

Refinamiento progresivo

Los requerimientos de mayor prioridad serán construidos antes. Estos requerimientos que pronto serán construidos serán reducidos a colecciones de ítems más pequeños y detallados. Se emplea esta técnica de refinamiento progresivo para dividir requerimientos grandes y poco detallados en un set de ítems más pequeños y detallados, previo a ser desarrollados.

Historias de usuario

Las historias de usuario son un formato conveniente utilizado para expresar valor de negocio de un PBI. El formato típico especifica una clase de usuario (el rol de usuario), algo que dicho rol quiere lograr (el objetivo), y el por qué (el beneficio).

Una forma de razonar acerca de las historias de usuario es con las tres C's, ideada por Ron Jefferies [17]:

- **Carta.**
  Se especificarán las historias de usuario en cartas en la herramienta tecnológica VersionOne. Estas incluirán un nivel de detalle limitado, pues su función es capturar la esencia de los requerimientos, ya que los detalles emergen en conversaciones futuras.

- **Conversación.**
  Las historias de usuario actuarán como meros recordatorios de conversaciones a tener más adelante, cuando sea el momento de desarrollar la funcionalidad descrita en las mismas. De esta forma, se permite un intercambio de información más rico y colaborativo que aseguran que los requerimientos correctos sean expresados y entendidos.

- **Confirmación.**
  Las historias de usuario contendrán información de confirmación, en la forma de condiciones de satisfacción. Estas serán los criterios de aceptación que clarifican el comportamiento deseado [18].

4.2.2.2. Priorización de historias utilizando método MoSCoW

Para realizar la priorización de las historias de usuario, se utilizó el método MoSCoW [19]. El equipo realizó la priorización previendo que en el caso de que el proyecto se desvíe de la estimación inicial, o surjan nuevos requerimientos, se puedan quitar del alcance del proyecto aquellas historias de menor prioridad, que no son clave para el éxito del producto de software. Como se verá en la próxima sección, esto resultó ser de gran utilidad.

Generalmente, se utilizan valores numéricos para asignar prioridad a los requisitos, siendo 5 la prioridad más alta y 1 la más baja, por ejemplo. Sin embargo estos valores no tienen un significado real ni semántico para los involucrados. Por esta razón se utilizó el método MoSCoW. Esta técnica de priorización ayuda a todo el

equipo a entender las necesidades reales del sistema software y entender qué tan crítica es cada funcionalidad.

MoSCoW es un acrónimo, cuyas siglas en inglés significan: *Must have*, *Should have*, *Could have*, and *Would like but won't get*. No se trata simplemente de poner un valor numérico a cada una de las historias de usuario, se busca aportar un valor semántico que refleja la importancia de cada una. A continuación se describen cada uno de los posibles valores:

- **M – MUST**: Requisitos totalmente imprescindibles que tienen que estar incluidos ya que si no se completan el proyecto no alcanza a ser exitoso.
- **S – SHOULD:** Requisitos que deberían de llevarse a cabo si es posible. Son requisitos importantes y de gran valor para el producto que se está construyendo, sin embargo el proyecto aún puede considerarse exitoso sin estos requisitos.
- **C – COULD:** Requisitos que sería bueno tener y podrían incluirse en caso de disponer de tiempo suficiente, no es compleja su implementación ni tampoco son prioritarios. Estos requisitos podrían quedar en el backlog para ser implementados en una fase posterior.
- **W – WON'T:** Requisitos que no se implementarán en la fase que se está planteando, pero lo pueden estar en un futuro. Estos requisitos no están dentro del alcance y objetivos actuales del proyecto, esto no implica que no sean importantes.

Se puede ver cada historia de usuario con su prioridad asignada en el anexo 12.10. Priorización de historias de usuario.

4.2.2.3. Confección de *user story map*

Una vez definida la técnica a utilizar para especificar los requerimientos, se pasó a confeccionar el *user story map* [20].

Para lograrlo, se siguió una serie de pasos que se detallarán a continuación. Cabe aclarar que los primeros dos pasos ya fueron implementados con la creación del perfil de cada usuario y el mapa de valor.

1. **Enmarcar el problema:** ¿Cuál es el problema que el producto le soluciona a los usuarios, o qué trabajo los ayuda a realizar? Es crítico determinar el objetivo para luego mapear el trabajo que le sigue, ya que los equipos deben asegurarse que están trabajando hacia el objetivo. El formato de historias de usuario (como [tipo de usuario] quiero [acción] para [beneficio]) es de gran ayuda para pensar en la interacción con el producto desde el punto de vista del usuario.

2. **Comprender los usuarios:** ¿Quién es el público objetivo para el producto? Es probable que haya más de uno. Los distintos tipos de usuarios pueden tener distintos objetivos y formas de interactuar con el producto. Es importante no sólo comprender al público objetivo, sino también construir historias desde su punto de vista.

3. **Mapear actividades de usuario:** Usualmente, todos los usuarios que interactúan con un producto lo harán a través de una serie común de actividades. Estas actividades - también llamadas temas o funciones - son la espina dorsal del user story map. Por ejemplo, los usuarios de un sitio de *e-commerce* desearían buscar productos a la venta, verlos por categoría, agregarlos al carro de compras, y completar la compra. Estas actividades comprenderían las épicas a lo largo de la parte superior del mapa, para luego ser descompuestas en historias de usuario más pequeñas. Una épica es una historia de usuario que es lo suficientemente grande como para poder ser dividida en historias de usuario más pequeñas.

4. **Mapear historias de usuario debajo de las actividades:** Una vez que la espina dorsal esté definida con las actividades, el equipo puede construir el resto del esqueleto del mapa, descomponiendo cada actividad o tema en historias de usuario más pequeñas. Por ejemplo, debajo de la actividad del

carro de compras, pueden existir historias como “como comprador, quiero editar y eliminar productos de mi carro, para poder cambiar de opinión antes de comprar”.

5. **Flujo y priorizar:** Con los temas de alto nivel y las historias detalladas en su lugar, el próximo paso es priorizar las historias, organizándolas de forma vertical tal que las más importantes estén arriba. Luego, el equipo debe mapear cómo es el flujo de los usuarios con el producto, típicamente de izquierda a derecha. Estas acciones ayudan a decidir cuáles historias son vitales y cuáles son menos importantes para entregar un producto que deleite al público objetivo.

6. **Identificar espacios, dependencias, requerimientos técnicos, y alternativas:** El story map le da al equipo la habilidad de visualizar problemas potenciales que pueden enlentecerlo más adelante, como cuellos de botella, dependencias, o información faltante. Identificar estos riesgos previo al diseño o desarrollo ayuda a minimizar y mitigarlos, y a proponer soluciones alternativas.

7. **Planificar *sprints* y *releases*:** Es aquí donde el equipo convierte un ejercicio visual en trabajo ejecutable. Con las historias priorizadas de arriba a abajo, el equipo puede ver el trabajo que permitirá entregar la mayor cantidad de valor con el menor esfuerzo, y agrupar estas historias en sprints de desarrollo y *releases* del producto. El equipo debe crear secciones horizontales a través del mapa, agrupando historias por prioridad dentro de cada actividad de usuario. La agrupación por *release* fue realizada con el objetivo de cada *release* en mente, de forma tal que en cada *release* se incluyeron aquellas historias que permitieron cumplirlos.

El *user story map* sufrió cambios durante el transcurso de la etapa de desarrollo del proyecto. Esto se debió a que, como fue mencionado en la sección de validación de este capítulo, se logró validar la sección del sistema correspondiente al perfil de instituciones académicas luego de ya comenzado el desarrollo del sistema. Esto

ocasionó que surgieran nuevas historias que se debieron incorporar al *backlog* y ser priorizadas. En la sección 6.2.8. Gestión del alcance se explica esto en más detalle.

Esta ocurrencia le marcó al equipo la importancia de realizar validaciones, y el riesgo que supone validar en etapas más tardías, ya que esto puede tener como consecuencia el re-trabajo. Por esta razón, el equipo vio la necesidad de incluir un *release* más, no planificado inicialmente. Este *release* fue ubicado temporalmente al finalizar el séptimo *sprint*, a fines de octubre. El objetivo fue validar la sección del sistema correspondiente al perfil del egresado y sus calificaciones, incluyendo su información personal, ingreso de calificaciones y verificación por parte de las instituciones académicas.

A diferencia de lo sucedido luego de validar el primer *release*, no surgió la necesidad de realizar re-trabajo luego de la validación realizada con el segundo *release*. Esto pudo deberse en gran parte a que esta sección del sistema sí había sido validada anteriormente con prototipos, lo cual resaltó la importancia de realizar validaciones tempranas utilizando prototipos que puedan ser desechables.

Se puede apreciar la versión inicial y final del *user story map* en mayor detalle en el anexo 12.11. User story maps.

4.3. Requerimientos

En esta sección se realizará una descripción de alto nivel de los principales requerimientos funcionales y no funcionales relevados para todo el sistema.

4.3.1. Requerimientos funcionales

Como fue mencionado previamente, se utilizaron historias de usuario para especificar los requerimientos del sistema. Como fue descrito en la sección de confección del *user story map*, en primer lugar se identificaron las épicas, y luego cada una de ellas fue descompuesta en historias de usuario más pequeñas.

Para hacer esta sección más legible, se detalla en qué consisten las funcionalidades de cada épica, mencionando luego las historias de usuario asociadas a cada una de

ellas. Luego, es posible leer el detalle de cada historia de usuario en el anexo 12.12. Historias de usuario. Este anexo contiene las historias agrupadas por *sprint*, incluyendo la descripción y los criterios de aceptación de cada una.

4.3.1.1. Gestión de instituciones

Esta épica comprende tareas que deben poder realizar los administradores del sistema para gestionar las instituciones. En un principio, las instituciones académicas se registraban en el sistema realizando una solicitud de registro a los administradores del sistema, quienes luego aceptaban o rechazaban estas solicitudes. Una vez validado el sistema, se cambió la forma en que se registran las instituciones académicas en el sistema.

Con la nueva forma, son los propios administradores los encargados de registrar las instituciones académicas en el sistema, también pudiendo editar los perfiles de ellas. Por esta razón deben tener la posibilidad de visualizar una lista con todas las instituciones existentes en el sistema, y tener acceso al perfil de cada una de ellas. Además de esto, los administradores deben determinar un correo electrónico de ingreso para el perfil de la institución. Con estas credenciales es que ingresará al sistema el encargado de verificar las calificaciones de los egresados de parte de la institución académica.

Historias de usuario asociadas:

- Procesar solicitud de institución académica - *Must*
- Registro y edición de instituciones académicas - *Must*

4.3.1.2. Registro de egresado

Esta épica incluye el proceso de registro de los egresados, así como también el posterior ingreso al sistema utilizando el usuario creado.

Cualquier egresado se puede registrar en el sistema, ya que el mecanismo de verificación de identidad está implementado en otra sección del sistema. Esta decisión fue tomada debido a que la verificación de la identidad del egresado es un

proceso que varía dependiendo de la institución educativa. Cada institución educativa puede tener distintos requisitos a la hora de verificar la identidad de uno de sus egresados. Por ejemplo, algunas instituciones educativas pueden asignar un número de estudiante a sus alumnos, mientras que en otras se puede utilizar el número de documento de identidad de los alumnos.

Una vez registrado el egresado, y luego de que este verifique su correo electrónico, el mismo debe poder ingresar al sistema utilizando su correo y la contraseña elegida.

Esta épica también incluye el ingreso con la cuenta de Google, aunque fue de las historias de menor prioridad que fueron excluidas del *release* final.

Historias de usuario asociadas:

- Registro de egresado - *Must*
- Ingreso de egresado - *Must*
- Ingreso con Google - *Won't*

4.3.1.3. Gestión de perfil de egresado

Una vez que el egresado ingresa al sistema, el mismo es redirigido a su perfil. Esta épica incluye el ingreso y edición de los datos personales del egresado dentro de su perfil, tales como imagen de perfil, y nombre, apellido, teléfono, y fecha de nacimiento.

Historias de usuario asociadas:

- Imagen de perfil de egresado - *Should*
- Datos de egresado - *Should*

4.3.1.4. Registro de institución académica

Como fue mencionado anteriormente, previo a realizar la validación con un integrante de una institución académica, el proceso de registro de una institución en

el sistema consistía en enviar una solicitud de registro a los administradores del sistema, quienes luego aceptaban o rechazaban estas solicitudes.

Esta épica incluye el proceso anterior de registro de las instituciones académicas. También incluye el ingreso al sistema utilizando el usuario creado. Una vez registrada la institución académica, y luego de que esta verifique su correo electrónico, la misma debe poder ingresar al sistema utilizando su correo y la contraseña elegida. Este ingreso se mantuvo igual luego de la validación, excepto por la parte de verificación de correo electrónico, la cual fue removida.

Historias de usuario asociadas:

- Registro de institución académica - *Must*
- Ingreso de institución académica - *Must*

4.3.1.5. Gestión de perfil de institución académica

Una vez que el integrante de la institución académica ingresa al sistema, el mismo es redirigido a su perfil. Esta épica incluye el ingreso y edición de los datos asociados a la institución académica, tales como imagen de perfil, nombre, teléfono, y dirección.

La institución académica también debe poder ingresar y editar los títulos que dicta, especificando qué tipo de títulos son (universitario, técnico, actualización profesional, etc). Como la cantidad de títulos que ofrece una institución académica puede superar el centenar, debe ser posible importarlos mediante un archivo CSV (*comma separated value*). Esta historia surgió a través de la validación realizada.

Además de poder gestionar los títulos, las instituciones académicas deben poder gestionar su lista de requisitos. Los requisitos son datos que los egresados deben completar para verificar su identidad ante la institución académica, y además otorgarle permiso a la institución para que ésta verifique las calificaciones del egresado a través del sistema. Estos requisitos pueden ser campos. destinados por ejemplo al ingreso de un número de estudiante o de documento. También pueden

ser archivos, como por ejemplo una carta firmada llamada “consentimiento”, la cual autoriza a la institución académica a verificar o rechazar las calificaciones presentes en el sistema.

Otra historia surgida durante la validación fue aquella que permite a las instituciones adjuntar archivos que sirvan de ayuda a los estudiantes a completar los requisitos. Esto fue realizado con la carta de consentimiento en mente, de modo que las instituciones académicas puedan ya proveer a sus estudiantes la carta de consentimiento con algunos datos en blanco. De esta forma, los estudiantes sólo deberían llenar los datos en blanco, firmar la carta, y volverla a subir.

Finalmente, esta épica también incluye las restricciones de campos en los requisitos, aunque esto formó parte de las historias de menor prioridad que finalmente fueron excluidas del *release* final.

Historias de usuario asociadas:

- Datos de institución académica - *Should*
- Imagen de perfil de institución académica - *Should*
- Títulos de institución académica - *Must*
- Requisitos de egresados - *Must*
- Archivo de requerimientos - *Should*
- Carga de títulos mediante CSV - *Should*
- Tipo de título - *Should*
- Restricciones de campos - *Won’t*

4.3.1.6. Gestión de calificaciones de egresado

Esta épica contiene todas las historias asociadas a la carga de calificaciones por parte de graduados dentro del sistema.

En el perfil del egresado, este debe poder seleccionar la institución académica de la cual se graduó de entre todas aquellas registradas en el sistema. Una vez seleccionada, debe poder ingresar sus calificaciones para alguno de los títulos ingresados ofrecidos por la institución académica. Finalmente, a cada una de estas calificaciones, el egresado debe poder adjuntar archivos, tales como el diploma escaneado en forma de imagen.

Historias de usuario asociadas:

- Seleccionar institución académica por parte de egresado - *Must*
- Ingreso de calificaciones - *Must*
- Archivos de calificaciones - *Must*

4.3.1.7. Solicitar verificación a institución educativa

Esta épica contiene todas las historias asociadas a la solicitud de verificación de calificaciones cargadas por parte del egresado para las instituciones académicas.

En primer lugar, el egresado debe poder ingresar los datos requeridos por la institución académica. Una vez rellenados todos los campos, el egresado debe poder solicitar la verificación de una calificación a la institución académica. Cada vez que lo haga, la institución académica deberá recibir un correo electrónico donde se le notifique de la situación.

Finalmente, esta épica también incluye el recordatorio a la institución académica para que verifique una calificación solicitada, aunque esto fue de las historias de menor prioridad que finalmente fueron excluidas del *release* final.

Historias de usuario asociadas:

- Ingreso de requisitos por parte de egresado - *Must*
- Solicitud de verificación de calificación - *Must*

- Notificación a institución académica - *Could*
- Recordatorio a institución académica - *Won't*

4.3.1.8. Aprobación o rechazo de solicitud de egresado

En esta épica están incluidas todas las historias de usuario asociadas al procesamiento de una solicitud de verificación de calificaciones por parte de una institución académica.

En primer lugar, la institución académica debe poder visualizar una lista de solicitudes pendientes, así como también dirigirse al perfil del egresado emisor de la solicitud. Dentro del perfil del solicitante debe poder visualizar la información personal de egresado y los datos ingresados para las calificaciones asociadas a la institución académica en cuestión.

Una vez que la institución académica pueda comprobar la veracidad de todos los datos ingresados, puede aprobar la solicitud, solicitar cambios, o rechazar la calificación recibida. El egresado debe recibir un correo electrónico donde se le notifique de la decisión tomada por la institución. Las instituciones académicas deben poder deshacer cualquiera de estas acciones previamente mencionadas.

Esta épica también incluye la notificación a los administradores cada vez que una institución académica rechaza una calificación, aunque esto fue de las historias de menor prioridad que finalmente fueron excluidas del *release* final.

Historias de usuario asociadas:

- Lista de solicitudes de verificación - *Must*
- Visualización de archivo de calificación - *Must*
- Aprobación o rechazo de solicitud de calificación - *Must*
- Revocación de verificación - *Must*

- Notificación a egresado - *Could*
- Notificación a administrador - *Won't*

4.3.1.9. Compartir perfil de egresado

Esta épica se refiere a lo que sucede una vez que un egresado ya cuenta con calificaciones verificadas. Cuando esto sucede, el egresado debe poder generar un vínculo para compartir su perfil con quien él lo desee. Luego, aquellos a quienes les fue provisto dicho vínculo podrán ingresar al sistema pudiendo acceder únicamente al perfil del egresado que les compartió el vínculo. En él podrán visualizar todas las calificaciones verificadas del egresado sin necesidad de estar registrados en el sistema.

Historias de usuario asociadas:

- Generar vínculo para compartir perfil de egresado - *Must*
- Ingreso a perfil de egresado con link generado - *Must*

4.3.2. Requerimientos no funcionales

A continuación se detallan los requerimientos no funcionales del sistema. Estos surgieron de dos fuentes principales: la información obtenida durante la etapa de relevamiento, y también de los objetivos planteados por el equipo. La clasificación de los atributos de calidad fue realizada utilizando aquellos atributos definidos en el libro *Software Architecture in Practice* [21].

4.3.2.1. Restricciones

RNF - RE01

Prioridad: *Must*

No se podrá definir una arquitectura *on premise*, es decir, una arquitectura que permita instalar y ejecutar el software dentro de las premisas de la organización. En

su lugar, el sistema debe ser desplegado utilizando el proveedor de servicios *Amazon Web Services*.

RNF - RE02

Prioridad: *Must*

El sistema a desarrollar debe ser un *SaaS* (*software as a service*), es decir, debe ser con suscripción y estar alojado de forma centralizada.

4.3.2.2. Usabilidad

RNF - US01

Prioridad: *Should*

Los usuarios deben poder navegar a todas las pantallas de la aplicación con una cantidad menor o igual a 3 clicks. Esto significa que, por ejemplo, las instituciones académicas deben poder navegar hasta el perfil de un egresado que ha realizado una solicitud de verificación con esta cantidad de clicks o menos.

RNF - US02

Prioridad: *Should*

Los usuarios deben poder deshacer acciones erróneas en el sistema sin sufrir consecuencias permanentes. Ya sea solicitar una verificación por parte de un egresado, o aprobar una calificación por parte de una institución académica, toda acción debe poder ser deshecha.

RNF - US03

Prioridad: *Should*

La aplicación no debe requerir un tiempo de aprendizaje mayor a 40 minutos. En caso de ser un egresado, en este tiempo el mismo debe aprender a editar su perfil, ingresar calificaciones, rellenando todos los datos necesarios, y solicitar la

confirmación de su calificación. En caso de ser una institución académica, en este tiempo el responsable de la misma debe aprender a gestionar su perfil, incluyendo títulos y requisitos, y procesar solicitudes de verificación de calificaciones.

RNF - US04

Prioridad: *Should*

Los usuarios deben contar con toda la información necesaria para comprender el estado de la aplicación. En todo momento los egresados deben estar al tanto de los estados de sus calificaciones, del mismo modo que las instituciones deben estar al tanto del estado de las validaciones correspondientes a su institución dentro del perfil de cada egresado.

RNF - US05

Prioridad: *Should*

Los usuarios deben ser capaces de pausar actividades tales como la creación de calificaciones por parte de los egresados, o la carga de títulos por parte de una institución académica, y reanudarlas cuando lo deseen.

4.3.2.3. Seguridad

RNF - SE01

Prioridad: *Must*

El sistema debe estar disponible para ser usado por usuarios autenticados. Debe contar con un mecanismo de autenticación.

RNF - SE02

Prioridad: *Must*

Los usuarios solamente deben poder acceder a aquellos datos para los cuales tienen autorización. La información a la que deberá tener acceso cada tipo de usuario es la siguiente:

- **Administrador**:
    - Creación y edición de instituciones académicas, incluyendo su información de institución, requisitos, y títulos.
    - Modificación de email de acceso de cada institución.
    - No deberá poder aprobar o rechazar calificaciones en nombre de la institución.
- **Institución académica:**
    - Edición de su perfil de institución académica, incluyendo su información de institución, requisitos, y títulos.
    - Lista de solicitudes de verificación pendientes, aceptadas, con cambios solicitados, y rechazadas.
    - Perfil de aquellos egresados que hayan solicitado una verificación a su propia institución. En el perfil del egresado sólo deberán ver la información asociada a su propia institución.
    - No deberá poder modificar el email de acceso de su institución.
    - No deberá poder acceder a perfiles de egresados que no le hayan solicitado una verificación.
- **Egresados:**
    - Edición de su perfil de egresado, incluyendo su información personal, e información asociada a instituciones, incluyendo requisitos ingresados y calificaciones.
    - Generación de vínculo para compartir su perfil con quien lo desee.
- **Usuario externo que utiliza el vínculo generado por egresado:**

- Perfil del egresado que le compartió el vínculo, pudiendo solamente visualizar aquellas calificaciones aprobadas marcadas con visibilidad “sólo autorizado” por el egresado.

RNF - SE03

Prioridad: *Must*

El sistema debe contar con un mecanismo de encriptación para proteger las comunicaciones de modo de asegurar la información que está en tránsito.

RNF - SE04

Prioridad: *Must*

El sistema debe contar con un mecanismo que asegura la integridad de los datos almacenados, de modo de poder detectar modificaciones no autorizadas.

4.3.2.5. Modificabilidad

RNF - MO01

Prioridad: *Should*

El código debe ser fácilmente modificable a lo largo de todos los elementos que componen el sistema, de forma tal que se facilite la incorporación de nuevas funcionalidades minimizando el impacto de cambio.

RNF - MO02

Prioridad: *Should*

El sistema debe permitir la implementación de cambios de forma tal que sólo deba ser desplegada la porción del sistema que sufrió el cambio, en lugar de tener que desplegar todo el sistema.

4.4. Conclusiones

En primer lugar, teniendo en cuenta las encuestas de satisfacción de los usuarios se puede concluir que el proceso de relevamiento de requerimientos fue exitoso. Las técnicas de relevamiento de requerimientos elegidas tuvieron varios beneficios. Por una parte, al combinarse con el *value proposition canvas*, permitieron diseñar un producto que se ajustó a las necesidades de los distintos usuarios y les aportó valor a los mismos. Por otro lado, la utilización de la técnica de análisis de documentos fue de gran ayuda en etapas posteriores, al momento de tomar determinadas decisiones de diseño relacionadas con el almacenamiento en blockchain. Esto se verá más en profundidad en el capítulo siguiente, 5. Arquitectura y desarrollo.

En segundo lugar, teniendo en mente lo ocurrido durante el desarrollo del proyecto, se puede concluir que la validación del sistema utilizando prototipos es una actividad de gran valor en los proyectos. Esta conclusión se puede extraer de cómo a diferencia de lo ocurrido con el *feedback* del perfil de las instituciones, no surgió la necesidad de realizar re-trabajo luego del *feedback* obtenido en relación a aquellas secciones del sistema que habían sido previamente validadas con prototipos desechables.

Finalmente, comenzar con el desarrollo de la única parte del sistema que no estaba validada puede ser visto como un error porque la validación tardía de algo que ya estaba construido causó re-trabajo. Esto le dejó al equipo varias lecciones. En primer lugar, la más evidente fue la de no comenzar con el desarrollo de una porción del sistema hasta que la misma no haya sido validada con los interesados correspondientes. Además de esto, con el objetivo de disminuir el re-trabajo, si en algún momento algún integrante del equipo se ve obligado a desarrollar algo que no haya sido validado aún, una buena alternativa es utilizar prototipos que tengan la interfaz de usuario funcional, pero que utilicen *mocks* para la interacción con el *back-end*. Esto podría ser otra forma de disminuir el re-trabajo. Sin embargo, más allá del re-trabajo ocasionado por esta validación tardía, esta tuvo la ventaja de que

se obtuvo un *feedback* más profundo al tratarse de un prototipo funcional, algo que era necesario.

5. Arquitectura y desarrollo

Este capítulo tiene como objetivo la presentación de los desafíos que se debieron enfrentar a la hora de definir la arquitectura, la arquitectura finalmente definida, la solución desarrollada, y las principales decisiones arquitectónicas tomadas a lo largo del proyecto. La arquitectura presentada fue diseñada para soportar los principales requerimientos del proyecto en cuanto a seguridad, usabilidad, y modificabilidad.

5.1. Descripción general de la arquitectura

A continuación, se realiza una breve descripción de la arquitectura del sistema con sus distintos elementos, utilizando un diagrama que muestra una representación básica de la misma y luego incluyendo una breve descripción de cada elemento representado.

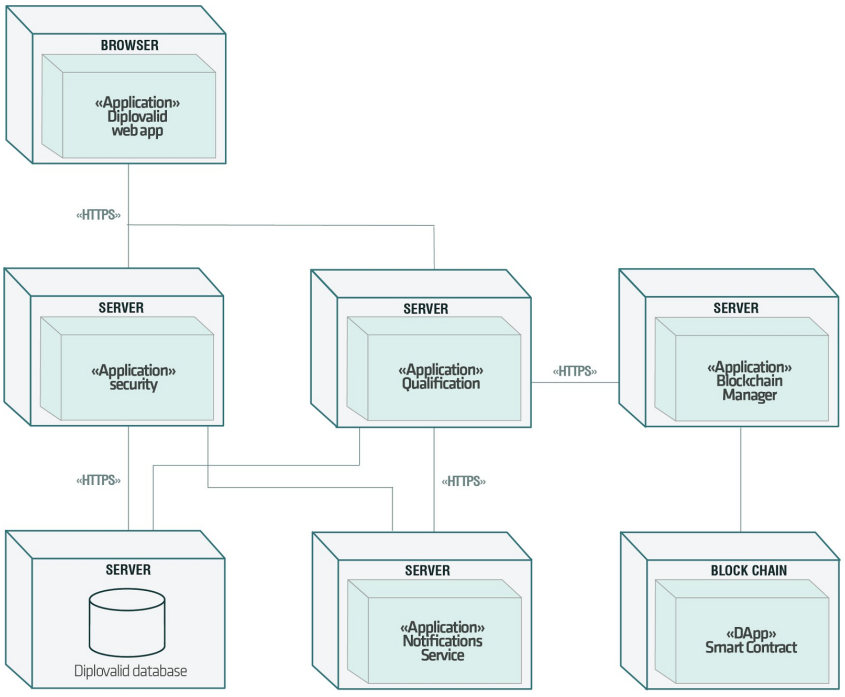


Imagen 21: Diagrama de arquitectura a alto nivel

5.1.1. Componentes

Diplovalid Web App

Aplicación web a la que acceden los distintos usuarios de Diplovalid; los egresados para subir información asociada a sus calificaciones y solicitar verificación, las instituciones educativas para modificar los datos de sus perfiles y procesar solicitudes de verificación, y los administradores para crear usuarios a las instituciones educativas y gestionar sus perfiles.

Security

Servicio encargado de la creación y gestión usuarios dentro del sistema, así como también de las de sesiones de los usuarios y sus permisos asociados.

Notifications

Servicio en el cual se centraliza el envío de notificaciones. Actualmente sólo se cuenta con notificaciones por email, aunque es posible extender la funcionalidad para agregar otros tipos de notificaciones.

Qualifications

Servicio encargado del manejo de la información de instituciones académicas y egresados, solicitudes de verificación por parte de egresados, y el estado de cada calificación.

Blockchain manager

Servicio encargado de encapsular la interacción con el contrato inteligente alojado en la blockchain.

Smart contract

Contrato inteligente alojado en la blockchain, encargado de realizar consultas y modificaciones a la información almacenada en la blockchain.

5.2. Atributos de calidad

Como fue mencionado en el capítulo anterior, en cuanto a atributos de calidad se utilizaron aquellos definidos en el libro *Software Architecture in Practice* [21]. Se definió la arquitectura del sistema en base a los atributos de calidad que fueron más recurrentes durante la etapa de relevamiento, definidos en la sección de requerimientos no funcionales. Estos fueron principalmente la seguridad y usabilidad. Además, al tratarse del desarrollo de un MVP también se tuvo que tener en cuenta la modificabilidad, debido a la alta probabilidad de que se deban introducir cambios en etapas posteriores. Esto se debe a que una vez terminado el proyecto, se realizará una transferencia a Effectus (la empresa cliente), y quedará en manos de ellos decidir si desean continuar con el proyecto o no.

Para cada uno de los atributos se definieron una serie de escenarios de requerimientos de atributos de calidad. Luego, se escogieron patrones y tácticas arquitectónicas con el fin de satisfacerlos.

5.2.1. Seguridad

Previo a detallar los requerimientos de seguridad, es pertinente definir qué es la seguridad de un sistema. La seguridad es la medida de la habilidad que posee un sistema para proteger sus datos e información de accesos no autorizados mientras le provee acceso a personas y sistemas que sí están autorizados. La forma más simple de definir la seguridad es alrededor de tres características:

1. **Confidencialidad:** protección de los datos o servicios de accesos no autorizados, como por ejemplo datos de calificaciones de los egresados.

2. **Integridad**: propiedad de los datos o servicios de no estar sujetos a manipulación no autorizada. En el sistema en cuestión, se trata de asegurar que los datos relacionados a calificaciones, como por ejemplo imágenes de diplomas, no hayan sido alteradas de forma no autorizada.

3. **Disponibilidad**: Propiedad del sistema de estar disponible para su uso legítimo.

Otras características que apoyan estas últimas tres son:

4. **Autenticación**: verificación de las identidades de las partes de una transacción. Esto es de gran importancia en el sistema a desarrollar, debido a que por la ley de protección de datos el sistema debe permitir a las instituciones académicas corroborar que el estudiante es quien dice ser previo a aceptar o rechazar las calificaciones.

5. **No repudio:** se centra en garantizar que el emisor de un mensaje no pueda negar haberlo enviado, y que el receptor no pueda negar haberlo recibido.

6. **Autorización**: asignación de privilegios a usuarios.

Los siguientes escenarios se desprenden en gran parte de la ley 18.331 (ley de protección de datos personales) [5], y en particular, del siguiente artículo:

- **Artículo 10:** Principio de seguridad de los datos.- El responsable o usuario de la base de datos debe adoptar las medidas que resulten necesarias para garantizar la seguridad y confidencialidad de los datos personales. Dichas medidas tendrán por objeto evitar su adulteración, pérdida, consulta o tratamiento no autorizado, así como detectar desviaciones de información, intencionales o no, ya sea que los riesgos provengan de la acción humana o del medio técnico utilizado.

Los escenarios definidos para la seguridad fueron los siguientes:

- **Interacción con *back-end API* - usuario autenticado y autorizado:** Procesamiento de la petición de forma adecuada.

- **Interacción con *back-end API* - usuario autenticado no autorizado:** No se procesa la petición, se retorna una respuesta de tipo *Unauthorized*.

- **Interacción con *back-end API* - usuario no autenticado:** No se procesa la petición, se retorna una respuesta de tipo *Unauthorized*.

- **Modificación no autorizada de datos:** el sistema debe detectar una modificación no autorizada en la base de datos. Este escenario está directamente relacionado con la integridad de los datos, “evitar su adulteración”.

- **Comunicación entre *front-end* y *back-end*:** El *back-end* debe poder asegurar que el *front-end* fue el emisor de una petición, y que la misma no ha sido modificada por terceros, y vice-versa.

Es posible ver los escenarios con mayor detalle en el anexo 12.13. Escenarios de requerimientos de atributos de calidad, en la sección Seguridad.

Una vez definida qué es la seguridad a grandes rasgos, se procede a clasificar los escenarios según las características de seguridad, y detallar qué tácticas y mecanismos se utilizaron para asegurar su cumplimiento.

Los primeros tres escenarios se centran en la interacción con la API del back-end (usuario autenticado y autorizado, usuario autenticado no autorizado, y usuario no autenticado). Estos escenarios tienen como principal objetivo la confidencialidad, disponibilidad, autenticación, y autorización. Para asegurar su cumplimiento, se utilizaron las siguientes tácticas:

- **Identificar actores:** se trata de identificar toda fuente de cualquier *input* externo al sistema. En el caso del sistema en cuestión, se identificaron los usuarios a través de IDs.

- **Autenticar actores:** significa asegurarse que un actor (usuario o computador remoto) es quien dice ser. Una forma de hacerlo es a través de contraseñas, certificados digitales, o identificación biométrica. Para implementar esta táctica se combinó el uso de contraseñas y códigos de un solo uso (*one-time*) en autenticación de 2 factores.

- **Autorizar actores:** significa asegurar que un autor autenticado tiene derecho a acceder a ciertos datos o servicios. Esto generalmente se logra mediante mecanismos de control de acceso, ya sea por actores o por clase de actores (grupos, roles, o listas de individuales). En el caso del sistema en cuestión, de modo de asegurar que los usuarios autenticados tienen los permisos necesarios para acceder o modificar los datos, se empleó un mecanismo de permisos que permite distinguir entre administradores del sistema, instituciones educativas, egresados, y usuarios externos no autenticados (empleadores). A su vez, dentro de la aplicación web, las instituciones sólo tienen acceso a perfiles de estudiantes que han ingresado calificaciones para su institución.

En el contexto del sistema en cuestión, estas tres tácticas son vitales para poder asegurar que cada usuario sólo puede acceder a la información que le concierne, y de este modo poder cumplir el requerimiento no funcional RNF - SE02. Los egresados sólo deben poder acceder a la información asociada a sus perfiles. Del mismo modo, cada institución sólo debe poder acceder a su perfil y a las calificaciones ingresadas por los egresados para su propia institución. Para ello, es necesario identificar tanto a egresados como a instituciones académicas, y luego utilizar un mecanismo de autenticación al momento en que estos ingresen al sistema. Finalmente, se debe autorizarlos únicamente para que puedan consultar los perfiles correspondientes.

El cuarto escenario (Modificación no autorizada de datos) se centra en la integridad de los datos. Es en este punto que una blockchain resulta especialmente útil debido a la inmutabilidad que la caracteriza. Se debió idear un mecanismo que utilice esta característica de la *blockchain* a su favor, y al hacerlo surgieron tres aspectos centrales sobre los cuales se tuvo que decidir:

1. Qué almacenar en la blockchain

2. En qué momento hacerlo

3. Manejo de concurrencia

Qué almacenar en la blockchain

En cuanto a esta disyuntiva, el enfoque inicial fue de almacenar toda la información en la *blockchain*, tanto datos en formato JSON como archivos. Sin embargo, fue la misma inmutabilidad que llevó al equipo a tomar otro camino debido a los siguientes artículos de la ley número 18.331 (ley de protección de datos personales) [5]:

- **Artículo 8:** Los datos deberán ser eliminados cuando hayan dejado de ser necesarios o pertinentes a los fines para los cuales hubieren sido recolectados.

- **Artículo 15:** Derecho de rectificación, actualización, inclusión o supresión.- Toda persona física o jurídica tendrá derecho a solicitar la rectificación, actualización, inclusión o supresión de los datos personales que le corresponda incluidos en una base de datos, al constatarse error o falsedad o exclusión en la información de la que es titular.

Estos artículos dictan que los datos deben poder ser eliminados en determinadas circunstancias, lo cual no sería posible en caso de ser almacenados en una *blockchain*. Esto obligó al equipo a idear una solución que permita almacenar una representación de los datos que no presente problemas al permanecer en una *blockchain*.

En este punto resultó de utilidad lo investigado durante la etapa de relevamiento, mediante el análisis de la documentación de sistemas similares ya existentes. Al igual que lo hacen los sistemas analizados, se decidió no almacenar los datos en su totalidad en la *blockchain*, en su lugar se adoptó la táctica de **verificar integridad de mensaje** [21]. Esta táctica se basa en el empleo de técnicas como *checksums* o *hashes* para verificar la integridad de los datos. Puntualmente, se optó por almacenar un *hash* de la información en la *blockchain*, mientras se almacena la información no hasheada en una base de datos tradicional. Almacenar hashes en la *blockchain* no presenta un problema dado que estos son irreversibles, por lo que es

imposible obtener la información inicial a partir de su *hash*. Además, al ser libres de colisiones, en caso de que se realice una modificación no autorizada en la base de datos tradicional, el *hash* almacenado en la *blockchain* dejaría de coincidir, evidenciando dicha modificación. De esta forma se logra asegurar la integridad de la información almacenada en la base de datos tradicional. Finalmente, sí es posible remover la data de las bases de datos tradicionales en caso de que un usuario así lo solicite, y la información en la blockchain si bien no es posible eliminarla permanece encriptada no siendo posible revertir esta encriptación.

En qué momento almacenar información en la blockchain

En cuanto a la segunda disyuntiva, la primer alternativa que se evaluó fue la de utilizar triggers de bases de datos. Estos desencadenan la ejecución de código cada vez que se crea, actualiza, o elimina un registro. Sin embargo, este enfoque haría que todas las modificaciones, legítimas o no, se propaguen a la *blockchain* haciendo que deje de cumplir su propósito. Por esta razón, se optó por actualizar la *blockchain* “manualmente”. Cada vez que se crea, actualiza, o elimina un registro, se realiza un llamado al microservicio *Qualifications*. Este, además de actualizar la base de datos, realiza un llamado al servicio *Blockchain Manager* para propagar el cambio a la información almacenada en la *blockchain*.

Manejo de concurrencia

Finalmente, en cuanto a la tercer disyuntiva, el manejo de concurrencia, se debió investigar el funcionamiento de la concurrencia para los contratos inteligentes. Tras investigar, se encontró que el orden de las transacciones incluidas en un bloque no está determinado por el momento de ejecución de las transacciones. Los resultados pueden variar dependiendo de cómo se ordenan las mismas [22]. Además, también existe el conflicto entre transacciones. Dos transacciones están en conflicto si acceden a la misma ubicación de memoria, y al menos una de ellas es una escritura. Por cada par de transacciones en conflicto, una es descartada, y la otra es aceptada [22]. Este conocimiento obligó al equipo a implementar un mecanismo de

manejo de concurrencia dentro de nuestro sistema, con el objetivo de evitar conflictos en las transacciones asociadas a nuestro contrato inteligente.

A continuación se puede observar un diagrama de secuencia de cómo se realiza la persistencia en la *blockchain* del *hash* de la información de un estudiante, utilizando un mecanismo de manejo de concurrencia.

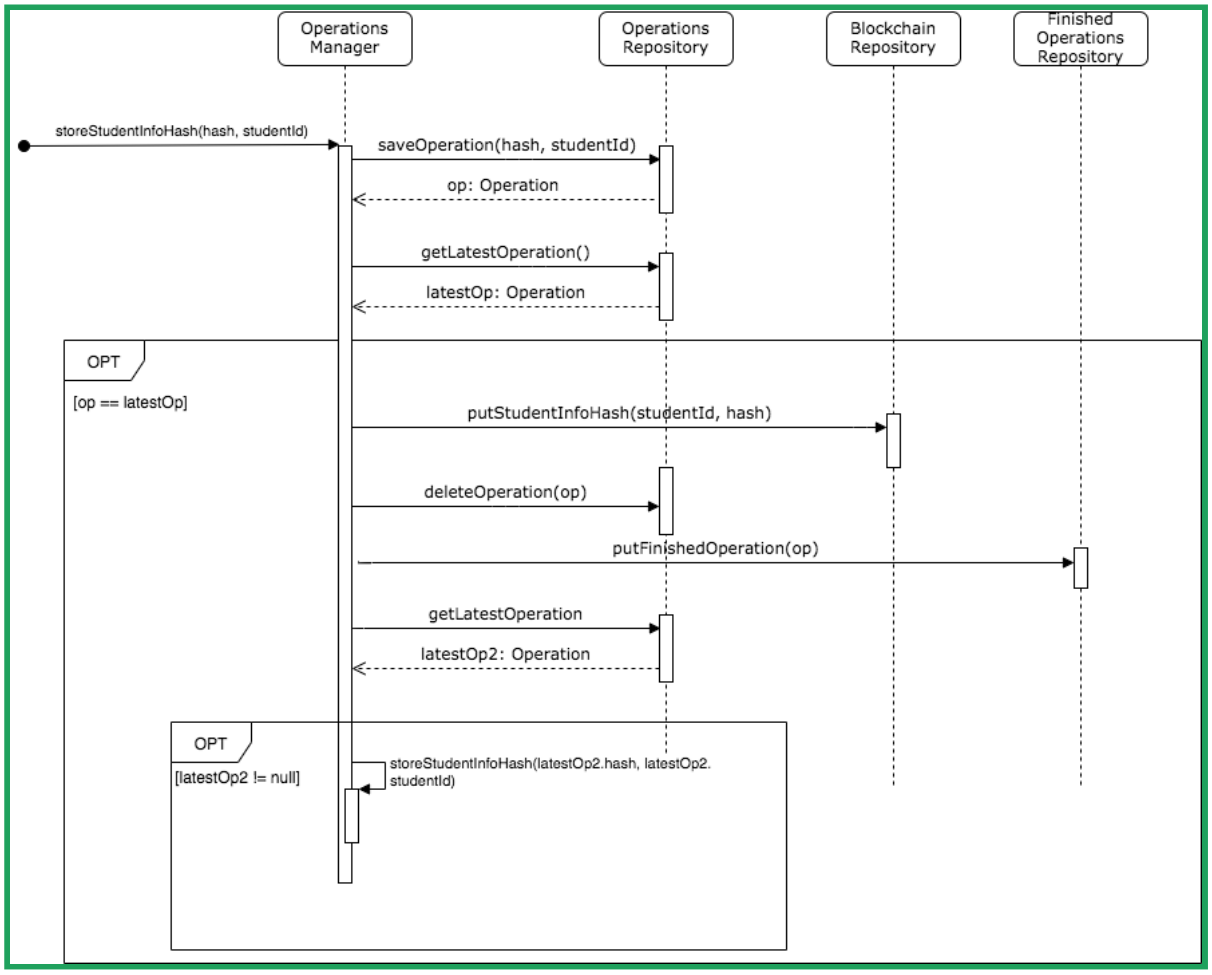


Imagen 22: Diagrama de secuencia de la escritura en *blockchain*

Cabe aclarar que la decisión de utilizar una *blockchain* para preservar la integridad afecta negativamente la performance del sistema, ya que las actualizaciones y consultas a la misma no tienen la misma velocidad que aquellas realizadas a las bases de datos tradicionales. Se realizó un *trade-off* entre seguridad y performance, debido a que durante el relevamiento de requerimientos, los distintos potenciales usuarios dieron mayor prioridad a la seguridad que a la performance.

El quinto escenario (Comunicación entre *front-end* y *back-end*) se centra en la autenticación. Para su cumplimiento se utilizó la táctica de encriptar data: toda comunicación entre *front-end* y *back-end* fue encriptada con una clave que sólo estos dos comparten, de modo que solamente sea posible la comunicación entre ellos.

Además de estas tácticas y mecanismos, se emplearon otras para asegurar la seguridad dentro del sistema:

- **Limitar acceso:** Involucra limitar el acceso a recursos tales como memoria, conecciones de red, o puntos de acceso. Esta táctica se debió emplear dentro del sistema con el objetivo de limitar la exposición del contrato inteligente, asegurando que solamente actores autorizados (en este caso, el *back-end*) puedan interactuar con él.

- **Encriptar datos:** Los datos deben ser protegidos de accesos no autorizados. Usualmente se logra la confidencialidad aplicando alguna forma de encriptación tanto a los datos como a la comunicación. La encriptación provee protección extra más allá de la autenticación a los datos persistidos. Las comunicaciones, por otro lado, no tienen controles de autorización. En estos casos, la encriptación es la única forma de protegerla. En el caso de el sistema en cuestión, es de suma importancia encriptar las comunicaciones debido a la sensibilidad de los datos que representan las calificaciones de los egresados. En cuanto a la implementación de esta táctica, toda contraseña se almacena en formato encriptado. Además, se utilizó criptografía asimétrica para la comunicación con el contrato inteligente alojado en la *blockchain*. Finalmente, como fue mencionado previamente, toda comunicación entre *front-end* y *back-end* fue encriptada.

- **Mantener *audit trail*:** Se trata de mantener un registro de las acciones realizadas por usuarios y sistemas, con el objetivo de facilitar el rastreo e identificación de ataques. También son útiles para analizarlos luego de un ataque con el fin de mejorar las defensas para un futuro. Para implementarla

se utilizaron los *logs* de *Cloudwatch*, un servicio provisto por *Amazon Web Services*.

5.2.1.1. Elección de blockchain

En cuanto a las elecciones relacionadas con blockchain, el equipo debió primero decidir qué tipo de blockchain utilizar. Se debió considerar tanto las blockchain públicas abiertas como las privadas cerradas.

Blockchain pública

En las blockchain públicas, todos pueden formar parte de ellas y explorarlas. Estas son sus 3 características principales:

1. Cualquiera puede descargar el código y comenzar a ejecutar un nodo público en su dispositivo local. Validando las transacciones en la red y participando así en el proceso de consenso, el proceso para determinar qué bloques se agregan a la *blockchain* y cuál es el estado actual.

2. Cualquier persona en el mundo puede enviar transacciones a través de la red y esperar verlas incluidas en la *blockchain* si son válidas.

3. Cualquiera puede leer cualquier transacción en el explorador de bloques público.

Tienen los siguientes efectos:

- Potencial para alterar los modelos comerciales actuales a través de la desintermediación

- Sin costos de infraestructura para los usuarios (el único costo es la tarifa de cada transacción). No es necesario mantener servidores o administradores de sistemas. Esto reduce radicalmente los costos de crear y ejecutar aplicaciones descentralizadas (dApps).

Blockchain privada

En las blockchain privadas, los permisos de escritura se mantienen centralizados para una organización o un conjunto específico de participantes (partes confiables). Los permisos de lectura pueden ser públicos o restringidos de forma arbitraria. Las aplicaciones probables incluyen administración de bases de datos, auditorías y otras que son internas a una sola compañía, por lo que la legibilidad pública puede no ser necesaria en absoluto. Las *blockchains* privadas son una forma de aprovechar la tecnología *blockchain* mediante la creación de grupos y participantes que pueden verificar las transacciones internamente. Las *blockchains* privadas tienen su caso de uso, especialmente cuando se trata de escalabilidad y cumplimiento del estado de las reglas de privacidad de datos y otros problemas regulatorios.

Tienen los siguientes efectos:

- Reducción de los costos de transacción y la redundancia de datos, y reemplazamiento de los sistemas heredados (legacy), simplificando el manejo de documentos y eliminando los mecanismos de cumplimiento semi manuales.

- Reduce los costos, pero no es disruptivo [23].

Comparación

	Públicas sin permisos	**Privadas con permisos**
Acceso	Lectura y escritura, públicos para cualquiera	Lectura y escritura, sólo con invitación
Actores de la red	No se conocen entre ellos	Se conocen entre ellos
Velocidad	Lenta	Rápida
Costo	Pago por transacción	Sin costo por transacción. Costo de mantener los nodos corriendo

Tabla 7: Comparación de blockchains

En una reunión con Alejandro Narancio, CEO de *Infuy*, surgió la idea de utilizar una blockchain privada y cerrada, en la cual cada institución académica pueda formar

parte de la blockchain teniendo su propio nodo. De este modo, al formar parte de la red las instituciones se aseguran que están contribuyendo a la seguridad de la red, participando en el proceso de consenso.

Teniendo en cuenta ambas opciones analizadas y la sugerencia del experto en el dominio, se decidió que la mejor opción es utilizar una blockchain privada con permisos. Esto se debe a que no se desea que cualquier persona pueda escribir o leer en la *blockchain*, sino que solamente aquellos autorizados para hacerlo, en este caso sería el *back-end* de la aplicación.

Esta idea fue posteriormente validada con Pablo San Nicolás, secretario de gestión académica de ORT, quien agregó que aunque le parecía una buena idea, “todo el asunto está todavía muy en pañales” como para saber si esa sería la forma definitiva. Por esta razón, aunque se continuó con la idea de tener una *blockchain* privada, se tuvo en cuenta que esta no era una decisión particularmente estable.

Además, como la idea propuesta por un experto en el dominio fue que cada institución académica en lo posible tenga un nodo, de esta forma se logra dividir los costos de infraestructura, quedando solamente el costo de tener cada nodo corriendo. Con las *blockchains* públicas, se debería cubrir el costo de cada transacción. Tomando en cuenta la cantidad de egresados y el costo promedio de una transacción, la suma de los costos de todas las transacciones podría tornarse considerable. A modo de ejemplo, a principios de julio del año 2019, la transacción promedio en Ethereum valía U$S 0,24 [24], y en bitcoin valía U$S 4,04 [25]. En Uruguay, en el año 2016 egresaron 12.665 alumnos. Viendo estos números, es posible llegar a la conclusión de que los costos de las transacciones podrían dispararse en un futuro en caso de optar por una *blockchain* pública.

5.2.2. Usabilidad

La usabilidad está relacionada con cuán fácil es para un usuario lograr una tarea deseada y el soporte que el sistema le provee al usuario. La usabilidad se ve compuesta por las siguientes áreas:

- Aprender funcionalidades del sistema: Si el usuario no está familiarizado con el sistema (o un aspecto particular del mismo) ¿qué puede hacer el sistema para facilitar su aprendizaje? Esto puede incluir ayudas.

- Usar un sistema de forma eficiente: ¿Qué puede hacer el sistema para que el usuario sea más eficiente durante su operación?

- Minimizar el impacto de errores: ¿Qué puede hacer el sistema para que los errores del usuario tengan un impacto mínimo? Por ejemplo, permitirle al usuario cancelar un comando erróneo.

- Adaptar el sistema a las necesidades del usuario: ¿Cómo puede el sistema adaptarse al usuario para facilitar la realización de las tareas?

- Incrementar confianza y satisfacción: ¿Qué hace el sistema para que el usuario se sienta confiado de que las acciones se están procesando? Por ejemplo, dar feedback que indique al usuario que el sistema está realizando una tarea de larga duración [21].

Los escenarios definidos para la usabilidad fueron los siguientes:

- **Navegación:** El sistema debe permitir al usuario navegar con facilidad.

- **Errores de usuario:** El sistema debe permitir al usuario deshacer la acción, recuperándose exitosamente de su error.

- **Visualización de pantalla:** En cualquier momento de la experiencia del usuario, se debe poder visualizar información acerca de la situación del usuario, manteniendo consistencia a través de la interfaz.

- **Estado de actividades:** En cualquier momento de la experiencia del usuario, se debe poder pausar la actividad momentáneamente, almacenando el estado actual para poder continuarla la actividad en otro momento.

Es posible ver los escenarios con mayor detalle en el anexo 12.13. Escenarios de requerimientos de atributos de calidad, en la sección de Usabilidad.

Con el objetivo de hacer que el sistema sea usable se facilitó la experimentación con la interfaz de usuario a través de la construcción de prototipos. De este modo, se permitió que posibles usuarios interactúen con bosquejos de la interfaz de usuario, proveyendo feedback valioso.

Se emplearon las siguientes tácticas de usabilidad para asegurar el cumplimiento de los escenarios determinados previamente:

- **Cancelar:** permitir al usuario cancelar cualquier acción que esté en proceso. De este modo, se facilita al usuario prevenir errores.

- **Deshacer:** para permitir deshacer una acción, se mantendrá información suficiente del estado anterior, con el fin de poder restaurarlo. Con esto se permite al usuario recuperarse de errores (escenario 2). Esta táctica es especialmente relevante en este sistema, debido a que es necesario permitirle a las instituciones académicas deshacer una confirmación si esta ha sido errónea o accidental.

- **Pausar / resumir:** se permitirá que los usuarios pausen y resuman operaciones que pueden volverse extensas, tales como la creación de calificaciones, o el ingreso de títulos por parte de instituciones académicas, los cuales pueden exceder los cien (escenario 4) [21].

5.2.3. Modificabilidad

La modificabilidad es un atributo de calidad del software que se relaciona con el costo de introducir cambios y la facilidad con la cual un sistema los incorpora [26].

Los escenarios surgen del hecho que el equipo se encontró desarrollando un MVP. Esto tiene como implicancia que sea altamente probable que se deban introducir cambios en etapas posteriores. Además, de la mano de lo mencionado en los capítulos 1 y 2, el equipo tuvo como objetivo el desarrollo de un producto de calidad, por lo cual la modificabilidad es un atributo de gran importancia.

- **Implementación de modificación de código:** ante la necesidad de realizar una modificación en alguna porción del sistema, el impacto de cambio no se debe propagar al resto del sistema, quedando recluido en la porción del sistema en la cual se implementó.

- **Despliegue de modificación de código:** cuando se realiza una modificación en alguna porción del sistema, dicho cambio debe poder ser desplegado de forma independiente, sin necesidad de re-desplegar todo el sistema.

Es posible ver el escenario con mayor detalle en el anexo 12.13. Escenarios de requerimientos de atributos de calidad, en la sección Modificabilidad.

Se decidió aplicar el patrón de microservicios como respuesta a estos requerimientos previamente mencionados. El patrón de microservicios define una arquitectura que estructura la aplicación en un conjunto de servicios poco acoplados que colaboran entre sí. Existe evidencia empírica de que el patrón de microservicios impacta positivamente la modificabilidad [27]. Además, este patrón también conlleva el siguiente conjunto de tácticas:

- **Dividir módulo:** cuando los módulos a modificar incluyen un conjunto grande de capacidades, es mayor la probabilidad de que los costos de modificación sean altos. Este es el caso de los sistemas monolíticos. Al refinar al sistema en varios servicios más pequeños, los módulos son más pequeños y es probable que los costos de cambio se reduzcan.

- **Aumentar coherencia semántica:** es una forma de aumentar la cohesión. La cohesión es una medida de cuán fuertemente relacionadas están las responsabilidades de un módulo. El objetivo del patrón de microservicios es dividir al sistema en servicios altamente cohesivos. De este modo, se reduce el impacto del cambio.

- **Encapsular:** es una forma de disminuir el acoplamiento. El acoplamiento es la medida de cuán fuertemente asociados están los módulos dentro de un sistema. La encapsulación trata de introducir interfaces explícitas a un

módulo. Dividiendo el sistema en servicios que esconden su implementación interna y exponiendo sólo algunas interfaces, se logra una mayor encapsulación dentro del sistema. De este modo, se asegura que los distintos servicios sólo pueden interactuar entre ellos a través de las interfaces expuestas.

A continuación, el diagrama de componentes del sistema. En el mismo se pueden apreciar los distintos microservicios que lo componen.

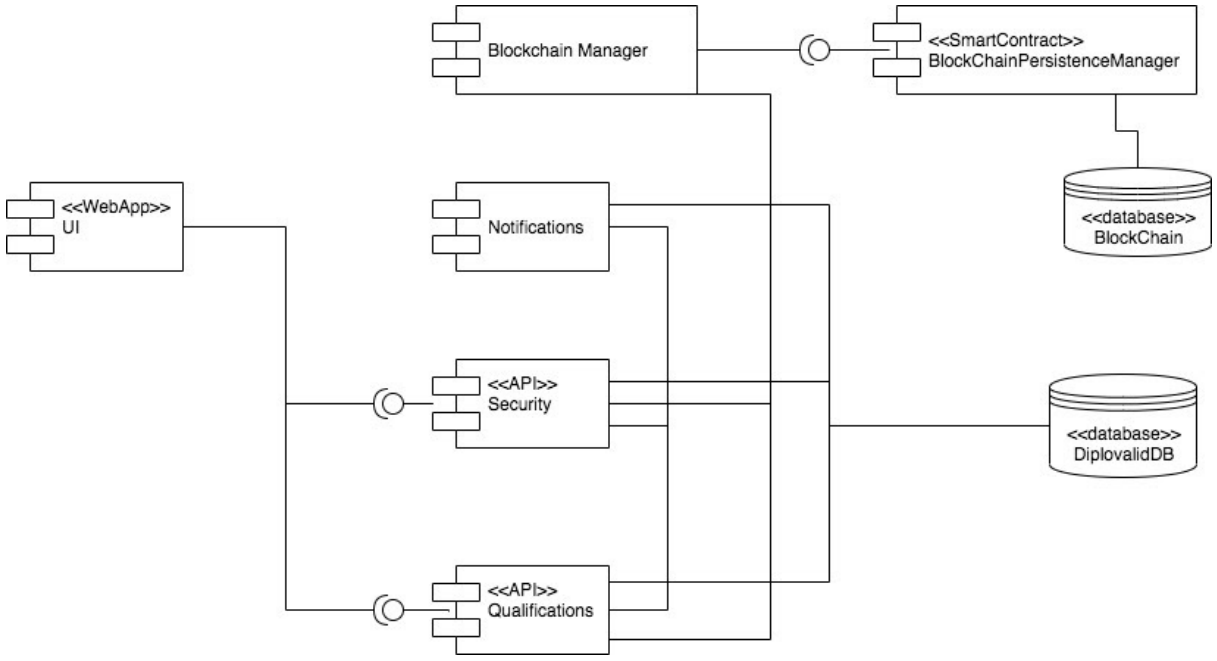


Imagen 23: Diagrama de componentes del sistema

Además de estas tácticas asociadas al patrón de microservicios, se utilizó la técnica de *defer binding*, con el objetivo de poder introducir cambios sin necesidad de re-desplegar código [21]. Un ejemplo de esto es el almacenamiento de los templates que se utilizan en los emails enviados en archivos externos. De este modo, en caso de que se quiera realizar una modificación en los mismos, no existe necesidad de re-desplegar el código, sino que basta con modificar el archivo correspondiente al *template*.

5.3. Desarrollo y elección de tecnologías

Una vez definida la arquitectura y los atributos de calidad pertinentes, se pasó a seleccionar las tecnologías para la construcción del sistema. Se realizaron varios análisis comparativos dependiendo de la tecnología en cuestión. Se listaron las ventajas y desventajas de cada tecnología, y se decidió por aquella que mejor cumpla los requisitos del proyecto.

5.3.1. Back-end

Hosting

En primer lugar se debieron tener en cuenta las restricciones. Una restricción es una decisión de diseño sin grado de libertad [21]. En el caso de esta arquitectura, y como fue mencionado en la sección de requerimientos no funcionales en el RNF - RE01, una de las principales restricciones a la que se enfrentó el equipo fue la de no contar con la posibilidad de definir una arquitectura *on premise*, es decir, una arquitectura que permita instalar y ejecutar el software dentro en las premisas de la organización. Esta restricción fue impuesta por el cliente.

Con el fin de aceptar esta restricción, se decidió optar por servicios de *cloud computing*. Estos ofrecen recursos de computación, tales como almacenamiento de datos y poder de cómputo según demanda, sin la necesidad de una gestión activa por parte del usuario.

Como lo dicta el requerimiento no funcional RNF - RE01, todos los componentes que conforman el back-end, con excepción del contrato inteligente, se encuentran desplegados utilizando servicios de *Amazon Web Services*. Sin embargo, se realizó un análisis de los distintos servicios requeridos por el sistema, con el fin de asegurar que esta sea una opción adecuada. Dicho análisis puede ser encontrado en el anexo 12.14. Análisis de proveedor de servicios web.

Amazon Web Services provee varias ventajas. En primer lugar realiza el manejo de infraestructura, de forma tal que el equipo no debió realizarlo. Ofrece varias

soluciones que escalan de forma automática y según la demanda. Además permite el manejo de variables de entorno tanto en AWS Lambdas como en AWS EC2 (las dos opciones analizadas en el anexo 12.15. Análisis de hosting para back-end: Amazon Lambda vs Amazon EC2). Esto permitió extraer del código el valor de distintas variables como identificadores de cuenta y contraseñas, evitando así una vulnerabilidad en la seguridad de la aplicación. También presenta el servicio de *CloudWatch*, el cual permite visualizar los logs tanto de instancias de EC2 como de funciones lambda.

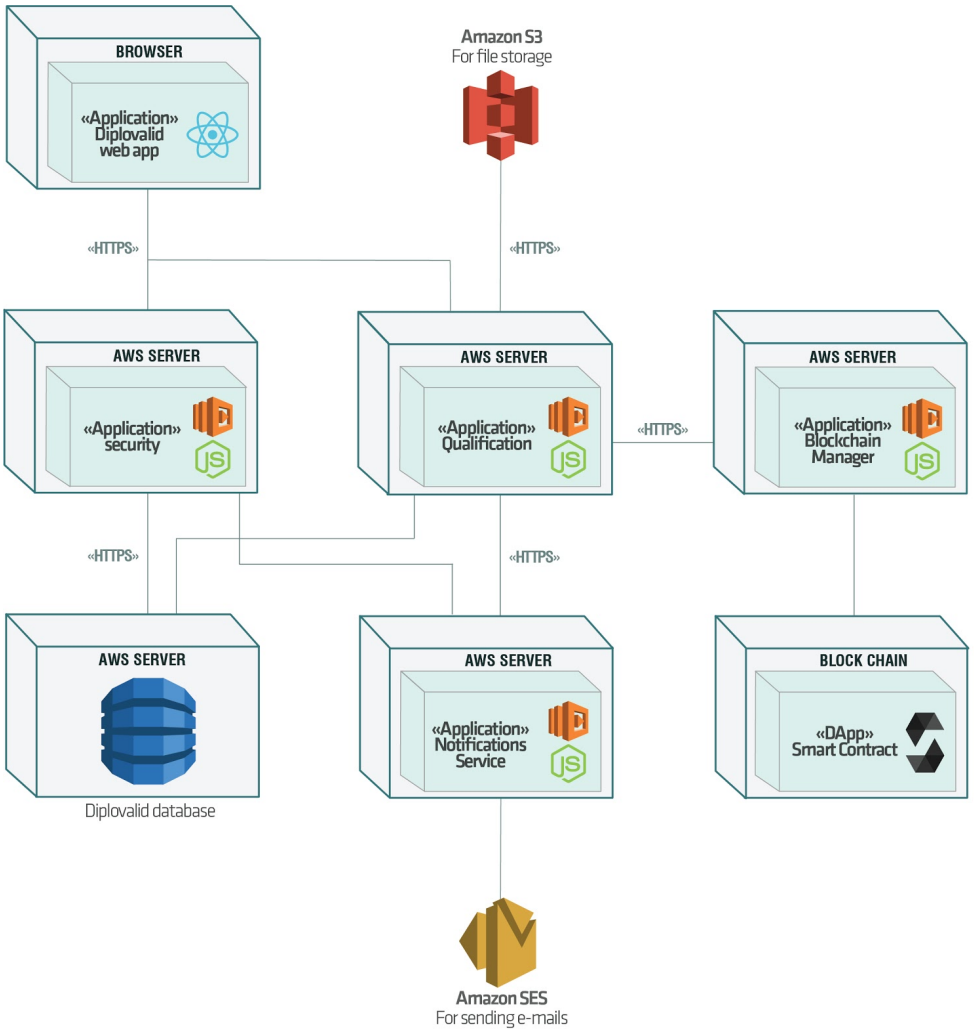


Imagen 24: Diagrama de módulos en AWS

- **S3:** *Simple Storage Service* - Servicio de almacenamiento ofrecido por *Amazon Web Services*. Se utilizó tanto para almacenar las imágenes de perfil

de los distintos usuarios, como para almacenar los archivos asociados a calificaciones.

- **DynamoDb:** Servicio de base de datos no relacional ofrecido por *Amazon Web Services*. Se utilizó para almacenar los datos del sistema.

- **SES:** *Simple Email Service* - Servicio de envío de correos electrónicos ofrecido por *Amazon Web Services*. Se utilizó para enviar emails a los usuarios, como por ejemplo, para verificación de cuentas.

Servicios

Se utilizó el lenguaje Node JS para desarrollar la lógica del *back-end*. Los microservicios *Security*, *Qualifications*, *Notifications*, y *Blockchain Manager* fueron implementados con dicha tecnología. El análisis que fundamenta dicha decisión puede ser encontrado en el anexo 12.16. Análisis de tecnologías para back-end. Los mismos fueron desplegados en forma de funciones lambda utilizando el *framework* serverless. Se decidió por la utilización de funciones lambda sobre contenedores EC2. Dicho análisis comparativo puede ser encontrado en el anexo 12.15. Análisis de hosting para back-end: Amazon Lambda vs Amazon EC2. Para el manejo de las dependencias dentro de cada microservicio se utilizó *NPM*. Esta herramienta también fue utilizada para la configuración del proceso de construcción (*build*).

Almacenamiento de datos

Se decidió por utilizar una base de datos no relacional para almacenar los datos de la aplicación. El análisis que fundamenta dicha decisión puede ser encontrado en el anexo 12.17. Análisis de base de datos: SQL vs NoSQL. Se utilizó *DynamoDb* como base de datos no relacional, un servicio ofrecido por Amazon Web Services, con el objetivo de aprovechar la co-ubicación de recursos. Además, *DynamoDb* se integra fácilmente con AWS Lambda, ya que es posible interactuar con la misma a través del SDK de *Amazon Web Services*.

A continuación, se realizará el análisis de la seguridad en cada capa del back-end:

Autenticación en API

Para realizar la autenticación en la API, y con el objetivo de mejorar la seguridad, se utilizó el estándar JSON Web Token (JWT). Se trata de un estándar abierto basado en JSON propuesto por IETF (RFC 7519) [28] para la creación de tokens de acceso que permiten la propagación de identidad y privilegios o *claims* en inglés. En este caso, el microservicio *security* genera un token indicando los privilegios o permisos del usuario. Dicho token podrá ser utilizado para acceder a las interfaces del resto de los microservicios. El token está firmado por una clave de *security*, que será conocida por el resto de los microservicios, y estará almacenada en una variable de ambiente. De este modo, el resto de los microservicios podrán verificar que el token es legítimo, y podrán decodificarlo sin tener que interactuar con el microservicio *security*, disminuyendo el tiempo de respuesta.

Inicialmente, se ideó el mecanismo de autenticación de modo que solamente el microservicio security conozca la clave con la cual se firma al token. Con esta condición, cada vez que un microservicio reciba una petición, este debería consultar al microservicio security para que descifre el token y retorne los privilegios o permisos asociados al mismo. Sin embargo, se notó una marcada caída en los tiempos de respuesta del sistema, los cuales superaron el doble del tiempo de respuesta inicial, por lo que finalmente se optó por el modelo descrito anteriormente.

Interacción entre microservicios y base de datos

Tanto las interacciones con los microservicios como con la base de datos tradicional, se realizarán a través de la AWS SDK. Para utilizar la misma, se requieren las credenciales asociadas a la cuenta de Amazon, las cuales se almacenarán en variables de ambiente.

Interacción con el contrato inteligente

Se definirá el contrato inteligente de modo que sólo el creador del mismo pueda interactuar con él, y se interactuará con el mismo a través de los *NPM Modules Web3* y *Truffle Wallet Provider*. Para utilizarlos, se necesitan las credenciales

asociadas a la cuenta que creó el contrato, las cuales también se almacenarán en variables de ambiente.

5.3.2. Front-end

La aplicación de *front-end* fue desarrollada utilizando *React*. Es posible ver el análisis que respalda esta decisión en el anexo 12.18. Análisis de tecnologías para front-end. En primer lugar, React es un lenguaje declarativo, lo cual facilita la creación de interfaces de usuario interactivas. Permite el diseño de vistas de forma simple, representando y actualizando los componentes adecuados cada vez que los datos cambian. Las vistas declarativas hacen que el código sea más predecible y fácil de hacer *debug*. En segundo lugar, es basado en componentes, permitiendo la construcción de componentes encapsulados que manejan su propio estado, para luego en conjunto conformar interfaces de usuario complejas [29]. Finalmente, utiliza mecanismos que permiten mejorar la performance de la interfaz realizando actualizaciones de forma más veloz, permitiendo la construcción de una interfaz altamente dinámica.

Hosting

El componente de *front-end*, al igual que la mayoría de los del *back-end*, se encuentra desplegado en *Amazon Web Service*. Más precisamente, utilizando el servicio *Simple Storage Service* (S3). Dicho servicio permite hacerlo de una manera muy simple. Basta con configurar un *bucket* para hostear una página web, y luego subir el contenido de la misma a dicho *bucket*.

Autenticación en front-end

Con el objetivo de incrementar el nivel de seguridad de la aplicación, se utilizó la autenticación de múltiples factores. Esta es un método de control de acceso informático en el que a un usuario se le concede acceso al sistema solo después de que presente dos o más pruebas diferentes de que es quien dice ser. En el caso del sistema en cuestión, un mail es enviado al usuario con un código que luego deberá utilizar para poder ingresar al sistema.

5.3.3. Blockchain

Al momento de decidir qué plataforma de blockchain utilizar, se debieron tomar en cuenta las necesidades del sistema. El principal requerimiento que el sistema tiene en cuanto a la tecnología de *blockchain* a utilizar es que esta debe permitir el desarrollo y ejecución de contratos inteligentes. Esto hizo que no se tuviera en cuenta la blockchain más popular, Bitcoin. El otro requerimiento principal es que la tecnología a utilizar permita la fácil migración a una *blockchain* pública, puesto que según lo discutido con San Nicolás, la decisión de utilizar una *blockchain* privada no es definitiva. Por esta razón, también se descartó la posibilidad de usar *Hyperledger Fabric*, un *framework* de código abierto para implementar *blockchains* privadas.

Se consideraron varias opciones en relación a qué plataforma utilizar. Finalmente, la decisión se redujo a Ethereum, EOS y RSK.

5.3.3.1. Ethereum

En 2013, un programador ruso-canadiense, Vitalik Buterin, lanzó el informe técnico de Ethereum y presentó al mundo las plataformas de contratos inteligentes. Desde entonces, Ethereum se ha fortalecido y se ha reunido para construir una de las comunidades de desarrolladores más saludables en el espacio criptográfico.

Si bien Buterin estaba intrigado por bitcoin y su potencial utilidad, sintió que la tecnología blockchain tenía mucho más que ofrecer que sólo ser un sistema de pago. Imaginó un futuro en el que los desarrolladores pudieran crear sus aplicaciones en la *blockchain*. Ethereum iba a ser una supercomputadora global descentralizada que alquilaría poder de cómputo a los desarrolladores para crear sus aplicaciones descentralizadas o dApps.

El desarrollo de Ethereum comenzó a principios de 2014 a través de una empresa suiza, Ethereum Switzerland GmbH (EthSuisse).

Ethereum utiliza una variante del consenso de Nakamoto, que se basa en la prueba de trabajo, generalmente se le conoce como PoW (*proof-of-work*). En ese sentido,

Ethereum es relativamente similar a Bitcoin. Sin embargo, durante su etapa final, se actualizará a un nuevo protocolo de consenso llamado Casper FFG, que generalmente se conoce como *proof-of-stake* o PoS.

En PoW, se tiene ciertos nodos llamados mineros que poseen equipos especializados. Estos tipos de equipos gastan potencia computacional y compiten para resolver operaciones criptográficamente difíciles. Aquel que la resuelve antes que el resto puede agregar su bloque a la *blockchain* y obtener la recompensa correspondiente.

Ethereum planea pasar al protocolo PoS ya que es más escalable y requiere menos recursos energéticos.

Desarrollo en Ethereum

Para comprender cómo funcionan los contratos inteligentes además de Ethereum, necesitamos comprender el concepto de gas.

Los contratos inteligentes en Ethereum están codificados en el lenguaje de programación Solidity. Solidity es bastante similar a JavaScript y es Turing completo. Un lenguaje es Turing completo si teóricamente puede resolver cualquier problema siempre que tenga suficiente tiempo y recursos. Es por eso que fue fundamental integrar un mecanismo de detención dentro del sistema, que detendrá el contrato inteligente cuando sea necesario. Ethereum utiliza la "tarifas de gas" como medidor de recursos. El gas es una unidad que mide la cantidad de esfuerzo computacional que se necesitará para ejecutar ciertas operaciones. En pocas palabras, cada paso de ejecución en el contrato cuesta gas y una vez que el costo del gas excede la tarifa prepaga, la ejecución se cancela.

Todas las operaciones que los usuarios desean ejecutar en ethereum deben proporcionar gas tanto para cubrir sus datos (también conocido como gas intrínseco) como para cubrir todo el cómputo realizado [30].

Entre las principales ventajas con las que cuenta Ethereum se encuentra que, al ser la primera plataforma para contratos inteligentes, aún es la plataforma que cuenta

con mayor cantidad de aplicaciones distribuidas corriendo, con 2518 aplicaciones distribuidas en julio del 2019. Para poner este número en contexto, la plataforma que le sigue contaba con 285 en la misma fecha [31]. Además, probablemente también sea la plataforma más cercana a ganar la aceptación general, dado que Microsoft y AWS ofrecen Ethereum blockchain-as-a-service. Por estas razones previamente mencionadas, se podría decir que una gran ventaja de Ethereum es la comunidad y soporte disponible. Este no es un aspecto menor, ya que al tratarse de una tecnología nueva, el apoyo de la comunidad y material disponible son de gran importancia.

5.3.3.2. RSK

Rootstock (RSK) es una plataforma de contratos inteligentes que está conectada a la blockchain de Bitcoin a través de tecnología de *sidechain*. Rootstock nació para ser compatible con las aplicaciones de Ethereum (el modelo web3 / EVM / Solidity) pero usando bitcoin como la criptomoneda subyacente. La idea detrás de la creación de RSK era dar las funcionalidades de contratos inteligentes de blockchain a Bitcoin.

La *blockchain* de Bitcoin tiene varias ventajas. Es relativamente antigua con seguridad comprobada, amplia distribución y popularidad. Además, también tiene una comunidad saludable con un fuerte poder de cómputo. RSK quiere que sus usuarios disfruten de los beneficios de Bitcoin como una reserva de valor y que, al mismo tiempo, puedan contar con la funcionalidad de contratos inteligentes y una mayor escalabilidad.

Desarrollo en RSK

Los contratos inteligentes en RSK se ejecutan dentro de la máquina virtual RSK (RVM). Las características principales de RVM son las siguientes:

- A nivel de código de operación, la RVM es compatible con la EVM (máquina virtual Ethereum), lo que significa que RSK puede ejecutar contratos de Ethereum.
- Los usuarios podrán ejecutar Ethereum DApps con la seguridad de la blockchain de Bitcoin. Esencialmente disfrutando lo mejor de ambos mundos.
- La comunidad RSK constantemente sugiere mejoras para la performance, las cuales están documentadas en numerosos RSKIPs (propuestas de mejora de RSK).

Este es un enfoque ingenioso que ha sido adoptado por RSK. En lugar de crear su propio lenguaje y obligar a los desarrolladores a trabajar de cierta manera, les permitirán usar el lenguaje de contrato inteligente más popular (Solidity) para crear dApps.

5.3.3.4. EOS

EOS se basa en el software EOSIO creado por block.one. Block.one está encabezado por Brendan Blumer y Dan Larimer. Larimer es el creador del *proof-of-stake* delegado y las organizaciones autónomas descentralizadas, también conocidas como DAO.

Si bien Ethereum marcó el camino para las plataformas de contrato inteligentes, aún es una plataforma muy lenta. Debido a su diseño, solo puede administrar 15-20 transacciones por segundo. Una plataforma con una latencia tan alta no puede soportar dApps modernas.

Eos tiene como objetivo apoyar aplicaciones descentralizadas a escala industrial. La forma en hacen es abordando los contratos inteligentes desde un ángulo diferente al de Ethereum. En lugar de ser una supercomputadora descentralizada, Eos planea

ser un sistema operativo descentralizado. Los usuarios de Eos poseerán recursos a cambio de su participación. Entonces, si se posee 1/1000 de la participación en EOS, entonces se poseerá 1/1000 de la potencia computacional y los recursos totales en EOS.

EOS delega la validación de la *blockchain* a 21 miembros de un comité de consenso, bajo un mecanismo de consenso de *proof-of-stake* delegado (DPoS).

En DPoS, cualquiera que tenga tokens en una *blockchain* integrada en el software EOS puede seleccionar a los productores de bloques a través de un sistema de votación de aprobación continua. Cualquiera puede participar en la elección del productor de bloques y se les dará la oportunidad de producir bloques proporcionales al total de votos que reciban en relación con todos los demás productores.

De este mecanismo se desprende la principal desventaja de EOS. Dado que EOS fue diseñado para depender solo de 21 productores de bloques para confirmar todas las transacciones, algunos han argumentado que la red está más centralizada que Ethereum y algunas otras criptomonedas. Mientras que EOS depende de la votación de los dueños de *tokens*, la baja participación de votantes podría conducir a una mayor centralización. Además, a algunos les preocupa el hecho de que los usuarios habituales no pueden auditar la red a menos que ejecuten personalmente un nodo completo.

Desarrollo en EOS

Como fue mencionado, EOS utiliza un modelo de gobernanza. Siempre que se cuente con tokens EOS, se tiene derecho a recursos propios como RAM, ancho de banda de red y ancho de banda de CPU a cambio. Dicho esto, estos recursos son muy escasos y es por eso que solo se puede conservar los tokens EOS, sin usarlos, durante un período de 3 años. A los titulares que no usen sus tokens por ese período se les cancelará su cuenta.

EOS utiliza *WebAssembly* (WASM) para desarrollar contratos inteligentes. Si bien WASM no es un lenguaje de programación, le da a los desarrolladores la posibilidad de codificar en el idioma que deseen, para luego compilarlo a un código de bytes que pueda ejecutarse en un navegador compatible.

Las razones por las que EOS eligió WASM son las siguientes:

- Flexibilidad: los desarrolladores pueden codificar en el idioma de su elección.
- Velocidad y eficiencia: *WebAssembly* se ejecuta a velocidad nativa aprovechando las capacidades comunes de hardware disponibles en una amplia gama de plataformas.
- Abierto y depurable: está diseñado para ser impreso en un formato de texto legible para depurar, probar, experimentar, optimizar, aprender, enseñar y escribir programas a mano.
- Seguro: *WebAssembly* describe un entorno de ejecución de espacio aislado seguro para la memoria que incluso puede implementarse dentro de máquinas virtuales JavaScript existentes.

5.3.3.3. Blockchain a utilizar

Tomando en cuenta las ventajas de cada una de las plataformas vistas anteriormente, el foco que el proyecto tiene en la seguridad, y lo hablado con Alejandro Narancio y Pablo San Nicolás, se decidió utilizar la plataforma RSK. Aunque es cierto que en un principio se utilizará una blockchain privada, también es cierto que, como fue mencionado previamente, esta decisión no es definitiva. Por ello, en caso de migrar el proyecto a una pública en un futuro, RSK cuenta con varias ventajas.

Adicionalmente RSK es compatible con Ethereum, por lo que si esta decisión cambiara podría fácilmente usarse el mismo contrato inteligente en la red Ethereum sin hacer ningún cambio en el mismo.

RSK es una plataforma de contratos inteligentes asegurada por la red Bitcoin. Al igual que los desarrolladores pueden desarrollar dApps para Ethereum y proyectos

similares, RSK les permite crear contratos inteligentes utilizando la seguridad de la red de Bitcoin y al mismo tiempo utilizando las herramientas de Ethereum y el código estándar probado y revisado por la comunidad.

RSK permite además la configuración de una red privada en unos pocos pasos, siendo posible levantar un nodo en poco tiempo. No es necesaria demasiada infraestructura para hacerlo, y basta con las direcciones IPs de los nodos pertenecientes a la red.

A pesar de lo previamente mencionado, el sistema no utiliza una blockchain privada para funcionar. Esto se debe a que, como fue mencionado en la sección de restricciones dentro de los requerimientos no funcionales, no se pudo contar con una arquitectura *on premise*, lo que significa que *Effectus* no pudo proveer servidores en los cuales ejecutar los nodos. Respetando dicho requerimiento, la alternativa a esto hubiera sido utilizar *Amazon Web Services*, aunque esta alternativa también se debió descartar debido a los costos.

Finalmente, con el objetivo de agilizar el proceso de desarrollo, se optó por utilizar una red de prueba. Estas cuentan con la ventaja de que las transacciones no tienen costo. Sin embargo, se realizó la prueba de levantar una red privada, y se incluyó un tutorial de cómo levantar un nodo privado. El mismo se pueden ver en mayor detalle en el anexo 12.19. Configuración de nodos de una blockchain privada.

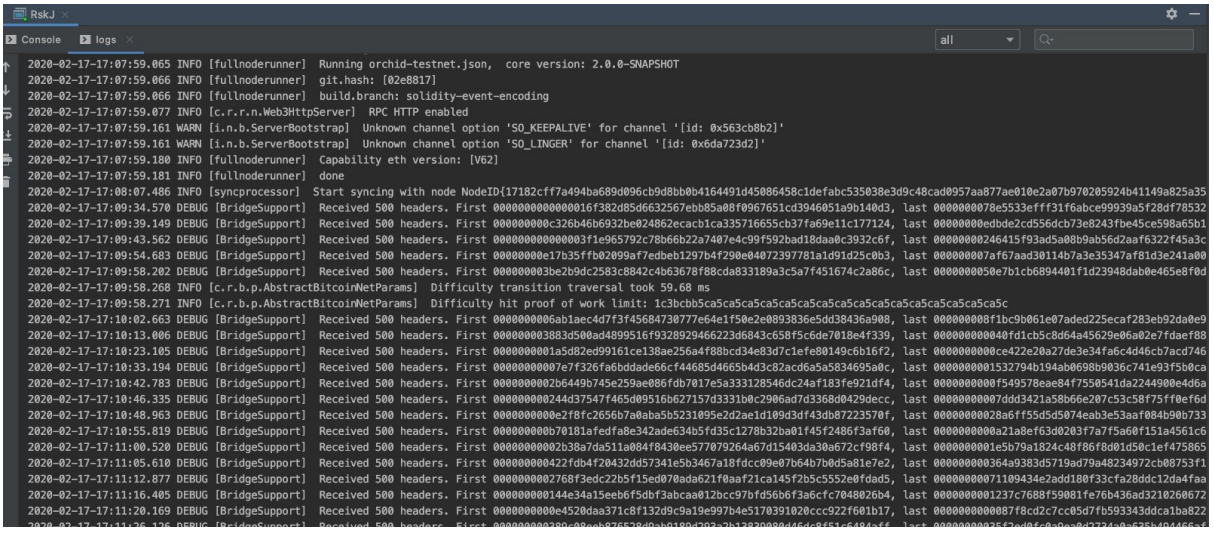


Imagen 25: Salida de consola de un nodo RSK

5.4. Conclusiones

Para concluir, se puede afirmar que la definición y tratamiento de los distintos atributos de calidad fue realizada de forma exitosa.

En lo que a seguridad respecta, el sistema toma varias medidas para asegurar su cumplimiento a lo largo de sus varias capas. Estas incluyen la autenticación de varios pasos en el *front-end*, la encriptación de la comunicación entre *front-end* y *back-end*, y el chequeo de integridad de la data utilizando la *blockchain*. Además de esto, el sistema brinda la sensación de seguridad a sus usuarios, como puede ser visto en las encuestas de satisfacción. El equipo quedó satisfecho con el nivel de seguridad que la utilización de una *blockchain* le provee al sistema, permitiendo verificar la integridad de toda información asociada a las calificaciones de los egresados.

Teniendo en cuenta las encuestas de satisfacción realizadas a los usuarios, y más precisamente aquellos resultados de las afirmaciones asociadas a la usabilidad, es posible concluir que el atributo de calidad usabilidad fue tratado de forma adecuada, desde su definición en la sección de requerimientos no funcionales, pasando por los escenarios de requerimientos, y finalmente llegando a las tácticas electas e implementadas. Se hace referencia a las encuestas de satisfacción debido a que la usabilidad tiene como principal objetivo a los usuarios, y a facilitar el uso de la aplicación por parte de los mismos. Los resultados de las encuestas de satisfacción pueden ser vistos en la sección 8.5.3. Pruebas con usuarios.

Por último, las tácticas empleadas en cuanto a la modificabilidad permitieron introducir cambios de forma más fácil durante el transcurso del proyecto. Como fue mencionado anteriormente, hubo un cambio en los requerimientos asociado a una validación tardía con integrantes de instituciones académicas. Gracias a las tácticas y al patrón electo, el equipo logró implementar y desplegar estos cambios de forma más ágil, ya que los cambios estaban contenidos en microservicios puntuales, y sólo

fue necesario re-desplegar aquellos microservicios que sufrieron cambios, en lugar de toda la aplicación.

Finalmente, el equipo cree que la elección de las distintas tecnologías fue adecuada. Por un lado, *Amazon Web Services* le brindó al equipo la posibilidad de trabajar con varios servicios que escalan según las necesidades del sistema sin necesidad de gestionar la arquitectura y con configuración mínima, como fue el caso de almacenamiento de archivos a través de S3, envío de correos electrónicos a través de SES, o despliegue de código utilizando *AWS Lambdas*. Por otro lado, almacenar los datos de la aplicación en una base de datos NoSQL le dio al equipo la capacidad de asimilar cambios en las estructuras de los datos sin necesidad de reestructurar la base de datos, ahorrando tiempo valioso, ya que estos cambios fueron una constante a lo largo del proyecto. Además, el equipo logró desarrollar los distintos componentes en los lenguajes electos sin contratiempos mayores, aprovechando los beneficios que ofrecen, como las numerosas librerías y gran comunidad y soporte. Por último, el equipo logró codificar y desplegar contratos inteligentes en la *blockchain* RSK sin mayores contratiempos, pudiendo acceder a documentación y otro material en línea cuando surgieron contratiempos.

6. Gestión del proyecto

En esta sección se definen las actividades realizadas para realizar la planificación, la ejecución del proyecto, la gestión del mismo a lo largo del tiempo, y los resultados obtenidos.

6.1. Etapas e hitos

El proyecto tuvo una duración aproximada de 12 meses, y el mismo se dividió en 3 etapas: *discovery*, *delivery*, y documentación. Existieron además instancias de evaluación formales impuestas por la universidad en fechas específicas. Se aprovechó la existencia de estas instancias para realizar un balance del avance del proyecto hasta el momento y los pasos a seguir. En algunos casos la fecha de las revisiones coincidieron con el cambio de etapa del proyecto.

6.1.1. Roadmap

En el siguiente diagrama se representan los diferentes hitos a los que el equipo se enfrentó a lo largo del proyecto.

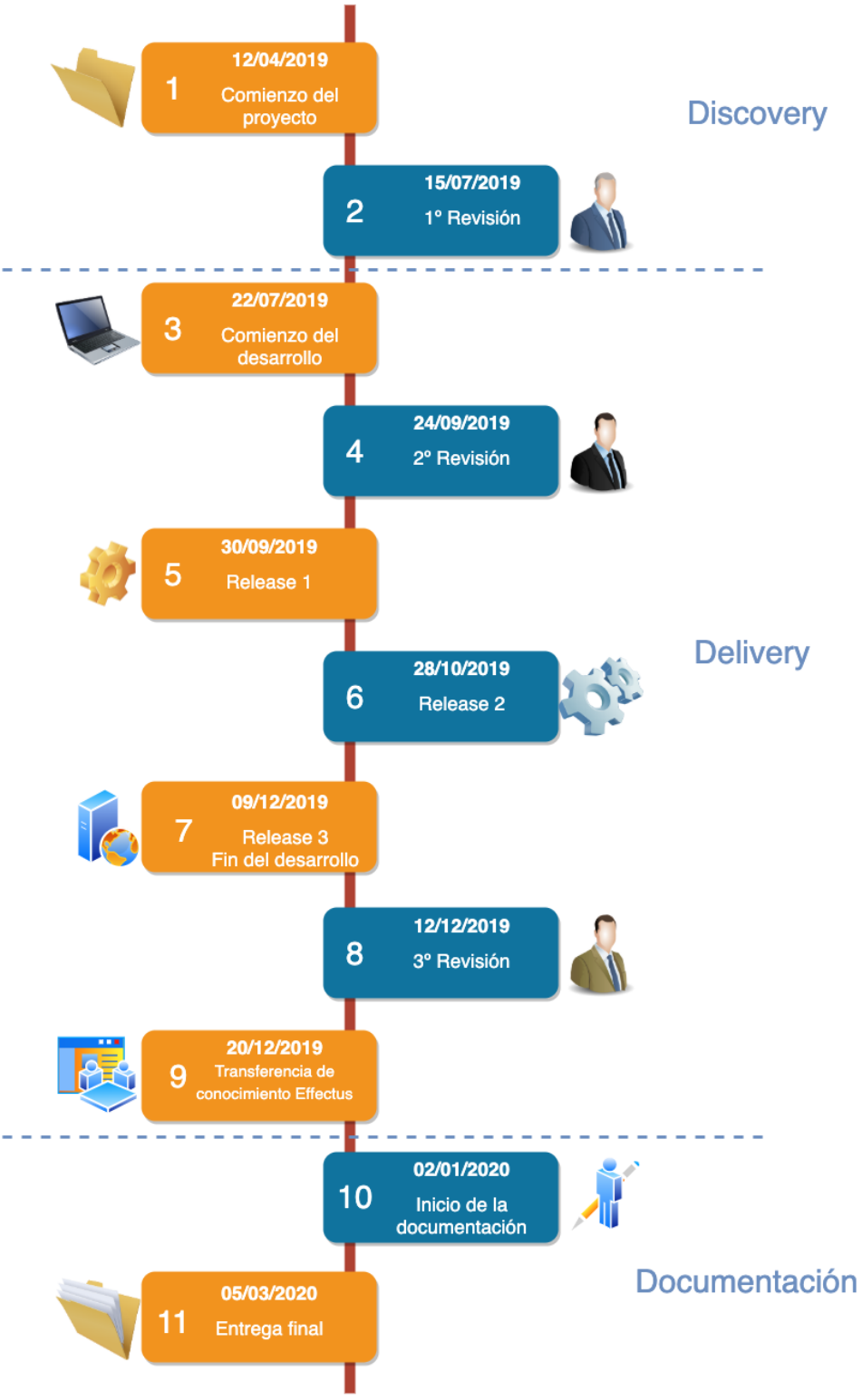


Imagen 26: roadmap del proyecto

6.1.2. Discovery

Los primeros meses de vida del proyecto, entre abril y julio del año 2019, fueron destinados a la etapa de *discovery.* Esto implicó el relevamiento de requerimientos, armado de prototipos y validación de los mismos. Durante este período también se fueron elaborando diversos documentos que sirvieron como base de soporte para la etapa de desarrollo o *delivery*. Entre esos documentos se destacan las primeras versiones de los planes de calidad y de riesgo, y del *user story map*.

6.1.3. Delivery

Durante la segunda quincena del mes de julio de 2019, luego de la primer revisión en la universidad, se dio inicio a la etapa de *delivery* durante la cual se desarrolla el producto con los requerimientos definidos en la etapa anterior. Como fue mencionado en el Capítulo 3. Marco metodológico, se eligió trabajar con una versión adaptada de SCRUM durante esta etapa, donde se segmentó el tiempo en iteraciones o *sprints* de dos semanas de duración.

Una vez completados los primeros 5 *sprints*, a fines de setiembre de 2019, llega la fecha de la segunda revisión del proyecto en la universidad. Se toma esta instancia como un hito en el proyecto, haciendo en esta fecha el primer *release* del producto. Al tener que preparar una presentación para la revisión, es una buena oportunidad para hacer un corte en el proyecto y evaluar el progreso hasta el momento. Como marcaba el objetivo del primer *release* (completar la creación y edición del perfil de las instituciones para su validación), este *release* es posteriormente presentado a Pablo San Nicolás para su validación. El feedback obtenido desemboca en nuevos requerimientos, los cuales se traducen en la necesidad de realizar re-trabajo para cambiar algunas funcionalidades.

Con el *feedback* obtenido tanto de la revisión como de la validación con San Nicolás, se sigue adelante con el desarrollo de la aplicación hasta fines de octubre. En esta fecha se hace un nuevo *release* para validar nuevamente el progreso de la aplicación. Esta vez, el objetivo del *release* comprendía desarrollar y validar el

proceso de ingreso y aprobación de calificaciones, por lo que fue validado con varios estudiantes y egresados, y nuevamente con San Nicolás.

A mediados de diciembre de 2019 se realiza la tercer y última revisión en la universidad, aprovechando esta oportunidad para hacer un nuevo *release* de la aplicación. A esta altura se ha completado ya el desarrollo de las principales funcionalidades del sistema, las historias que quedan pendientes en el backlog son de baja prioridad o se han definido como posibles mejoras para una próxima versión. Luego de esta revisión, se agenda una última validación del producto con los usuarios donde se certifica que el producto desarrollado cumple con los objetivos planteados y brinda valor a los usuarios.

Una vez culminado el desarrollo y validado el producto con los usuarios se procede a desarrollar el documento de transferencia de conocimiento a Effectus. En este documento se incluye un análisis del problema a solucionar, la solución planteada, ventajas y desventajas del uso de blockchain, comparación con otras alternativas de almacenamiento de datos, y las conclusiones obtenidas. Una vez completo el documento, se realiza una reunión en las oficinas de Effectus donde se realiza una demo de producto desarrollado y se evacuan algunas dudas. Se hace entrega del documento y el código fuente, quedando el equipo a disposición para evacuar dudas en el futuro. Dicho documento puede encontrarse en el anexo 12.1. Documento de transferencia de conocimiento a Effectus.

Imagen 27: transferencia de conocimiento a Effectus

Se da por concluida la etapa de *delivery* y se pasa a la siguiente etapa.

6.1.4. Documentación

Se hace necesario incluir esta etapa por tratarse de un proyecto académico, donde todo el esfuerzo realizado debe verse reflejado en la documentación para ser corregido. En otro contexto probablemente no sería necesaria.

Se reservan los meses de enero y febrero de 2020 para la elaboración de la documentación a entregar en los primeros días de marzo 2020. Durante el mes de enero se redacta una primer versión del documento con todas las secciones completas. Luego durante febrero se va refinando el documento con comentarios del tutor y correcciones cruzadas entre miembros del equipo.

6.2. Aplicación del marco de trabajo

Como se mencionó en el capítulo del marco metodológico, durante la etapa de *discovery* se trabajó con una metodología Kanban. No existieron roles ni ciclos de trabajo, las distintas tareas se iban colocando en un tablero creado con la herramienta *Trello,* con las columnas *Ready, In progress, y Done*. Semanalmente el equipo se reunió para evaluar el avance, y definir los próximos pasos a seguir.

En esta sección se hace foco principalmente en la etapa de *delivery*, durante la cual se trabajó con una metodología ágil adaptada a los requerimientos del proyecto y las necesidades del equipo. A continuación se describen algunas de las decisiones tomadas y las adaptaciones más relevantes.

6.2.1. Definición de historias de usuarios

Se describe a continuación los diferentes criterios usados para definir el estado de una historia de usuario.

6.2.1.1. *Definition of ready*

Las historias de usuario deben estar listas o *ready* para que el equipo pueda comprometerse con ellas y comenzar el trabajo de desarrollo en el sprint. La forma de definir si una historia está lista para trabajar en ella es estableciendo criterios o características que nos aseguren que la historia es clara, no ambigua, y con el suficiente detalle para comenzar a trabajarla.

1. **Cumplir con el modelo INVEST para refinar historias de usuario**

Es importante que las historias de usuario cumplan con el modelo *INVEST* para que sean **I**ndependientes, **N**egociables, **V**aliosas, **E**stimables, **P**equeñas (***S****mall***) y **T**esteables. Asegurarse de que las historias cumplen con estas características facilitó el trabajo de entendimiento por parte del equipo y a su vez también ayudó a realizar una rápida estimación.

2. **La historia de usuario debe asegurar que generará valor al negocio** [32]

Se hizo hincapié en la importancia de que todas las historias de usuario cuenten con el “Para” dentro de su estructura:

Como <Rol>

Quiero <Funcionalidad>

Para <Justificación> ← Aquí se establece el valor

De esta forma el equipo se aseguró de que haya un objetivo detrás de lo que se desarrolle y de que esto ayude a cubrir una necesidad en particular. De igual forma fue importante establecer un ordenamiento en cuanto al valor que las historias de usuario generarán al negocio, y esto se realizó a través de la priorización.

3. **La historia de usuario cuenta con sus criterios de aceptación y estos son claros**

Toda historia de usuario debió tener criterios de aceptación que ayuden a facilitar el desarrollo del producto y a tener claro cuál es el resultado que se está esperando por parte del dueño del producto y los interesados.

4. **Conversación entre el equipo**

Otra característica es el hecho de que una historia de usuario haya pasado por una conversación. Tradicionalmente, esta conversación se daría entre el *product owner* y el equipo de desarrollo con el fin de resolver dudas relacionadas con las historias de usuario, y discutir los criterios de aceptación y su estructura. En el caso de este equipo, como todos los roles estaban distribuidos entre sus dos integrantes, estas conversaciones se daban entre todo el equipo *Scrum*.

5. **La historia de usuario cuenta con una estimación de tamaño / complejidad**

La historia de usuario ha sido estimada en cuanto a su tamaño / complejidad, lo que demuestra que quedó clara, hubo conversación y ya no existieron dudas profundas que eviten que el equipo de desarrollo pueda realizar una estimación certera.

6.2.1.2. Criterio de aceptación

Para medir la calidad de un criterio de aceptación se utiliza el método SMART en el que se han de cumplir en lo máximo posible los siguientes criterios:

S - *Specific* (Específicos)

M - *Measurable* (Medibles)

A - *Achievable* (Alcanzables)

R - *Relevant* (Relevantes)

T - *Time-boxed* (Limitados en el tiempo)

Se usará el método BDD mediante la definición de escenarios de prueba usando el estilo *Given - When - Then* (Dado - Cuando - Entonces) [33].

La parte *given* o *dado* describe el estado de la aplicación antes del escenario que se está especificando. Actúa como una precondición.

La parte *when* o *cuando* describe el comportamiento que se está especificando.

Finalmente, la parte *then* o *entonces* describe los cambios que se espera haya producido el escenario.

6.2.1.3. Definition of done

Se definen los siguientes requisitos para poder determinar que una historia de usuario está terminada:

1. Código escrito para la funcionalidad deseada.
2. Código cumple con el estándar definido según análisis ejecutado por la herramienta correspondiente.
3. El proyecto compila sin errores.
4. Se escribieron tests unitarios para la lógica del *back-end*, se corren y pasan todos.
5. Se despliega el proyecto en ambiente de prueba igual al de producción sin errores.
6. QA verifica el funcionamiento.
7. Se arreglan errores hallados por QA.
8. Refactoring completo.
9. Cambios en la configuración u otra información necesario para el build debidamente documentados.

10. PCR (*peer code review*) realizada.

6.2.2. Sprints

Se optó por trabajar con sprints de 2 semanas de duración. Por tratarse de un proyecto de requerimientos en constante cambio y evolución, esta elección permite la obtención de feedback frecuente y análisis del trabajo realizado. Además posibilita ir realizando los ajustes necesarios tanto al proyecto como al proceso en forma continua y con mayor regularidad. Otro factor que incidió para definir la duración de los sprints fue la experiencia positiva que ambos miembros del equipo han tenido en sus trabajos particulares trabajando con sprints de 2 semanas de duración.

Se presenta a continuación el calendario con la fecha de comienzo y fin de cada uno de los sprints.

Sprint	Inicio - Fin	Sprint	Inicio - Fin
Sprint 1	22/07/2019 - 08/05/2019	Sprint 6	30/09/2019 - 14/10/2019
Sprint 2	08/05/2019 - 19/08/2019	Sprint 7	14/10/2019 - 28/10/2019
Sprint 3	19/08/2019 - 02/09/2019	Sprint 8	28/10/2019 - 11/11/2019
Sprint 4	02/09/2019 - 16/09/2019	Sprint 9	11/11/2019 - 25/11/2019
Sprint 5	16/09/2019 - 30/09/2019	Sprint 10	25/11/2019 - 09/12/2019

Tabla 8: calendario de sprints

6.2.3. Ejecución de *sprints*

Se divide el análisis de la evaluación de los sprints en 3 etapas, cada una correspondiente al trabajo realizado entre releases.

6.2.3.1. Primer *release* - *Sprints* 1 a 5

Durante los primeros *sprints* se trabajó en la configuración del ambiente de desarrollo, repositorios, y servidores donde hostear la aplicación.

Luego se comienza con el desarrollo de las funcionalidades que permitan el registro e ingreso al sistema tanto de instituciones académicas como de egresados. Se hace foco en el desarrollo del perfil de la institución académica.

Al finalizar esta etapa, que coincide con la segunda revisión en la universidad, se realiza el primer release el cual se valida con Pablo San Nicolás quien actúa como experto de dominio en el ámbito académico. Durante esta reunión surgen nuevos requerimientos que deben ser priorizados e incluidos en el *backlog*, en la sección 6.2.8. Gestión del alcance de este capítulo se detalla de qué forma esto fue resuelto.

6.2.3.2. Segundo *release* - *Sprints* 6 y 7

Teniendo validadas las funcionalidades relacionadas al perfil de la institución académica en la aplicación y las nuevas historias priorizadas, se trabaja en en el perfil de los egresados. Se implementan las funcionalidades para el registro de la información personal de cada egresado, el ingreso de las calificaciones obtenidas, y se incorpora el proceso de verificación de estas calificaciones.

Es durante esta etapa que se realiza la integración de la tecnología blockchain al proyecto, mediante el desarrollo de un contrato inteligente que registre la información del graduado y sus calificaciones. Se implementa también la comunicación entre el *back-end* del sistema y el contrato inteligente.

Una vez completados estos *sprints*, se hace un nuevo *release*. Dicho release se valida tanto con un grupo de alumnos y egresados como con Pablo San Nicolás, con la finalidad de obtener *feedback* y realizar ajustes. Se valida de esta manera debido a que durante este *release* se desarrollaron funcionalidades que utilizarían tanto los egresados como las instituciones académicas. De esta devolución no

surge la necesidad de implementar nuevas funcionalidades ni hacer cambios significativos a lo ya desarrollado.

6.2.3.3. Tercer release - Sprints 7 a 10

Durante esta etapa se implementan en primer lugar los cambios que surgieron luego de la validación del primer release. Entre otros la necesidad de que el perfil del administrador en la aplicación pueda encargarse de precargar los datos y carreras correspondientes a una determinada institución académica, con la finalidad facilitarle el uso de la aplicación a dicha institución. Se agrega también la funcionalidad que permite al graduado compartir su perfil en la aplicación (con sus títulos validados) con terceros a través de un link.

6.2.4. *Sprint planning*

Al comienzo de cada sprint, y como parte del mismo, se realizó la ceremonia *sprint planning* que consiste en la estimación, selección, y planificación de las historias a trabajarse durante las próximas semanas [34].

Se hizo hincapié en que esta ceremonia la realicen en conjunto ambos integrantes del equipo, ya que es muy valiosa la discusión sobre la complejidad de una tarea. Esto permitió minimizar los errores de estimación, debido a que es necesario llegar a un consenso luego de la discusión. A medida que el equipo fue ganando experiencia al llevar a cabo tareas similares, el proceso fue mejorando y los errores de estimación comenzaron a reducirse.

Al momento de armar el sprint, se seleccionaron aquellas tareas que el equipo pudo comprometerse a completar en un período de 2 semanas, de acuerdo a la complejidad de las mismas y el tiempo dedicado al proyecto de parte de cada integrante. El equipo también debió considerar aquellas historias del *sprint* anterior que quedaron sin finalizar previo a comprometerse con nuevo trabajo a realizar.

La estimación de las tareas se hizo en *story points*, unidad de medida para comparar la complejidad de distintas tareas respecto a otras. La técnica utilizada para realizar la estimación fue *planning poker*, debido a su sencillez de uso y

eficacia [35]. Se decidió utilizar como posibles valores la secuencia de fibonacci hasta el número 13 inclusive. Una historia con un valor estimado superior a 13 se consideró demasiado grande y debió ser dividida en 2 o más historias. Como herramienta se utilizó la aplicación móvil *Scrum Poker*, que permite a cada miembro del equipo realizar su voto una vez leída en voz alta la historia a estimar.

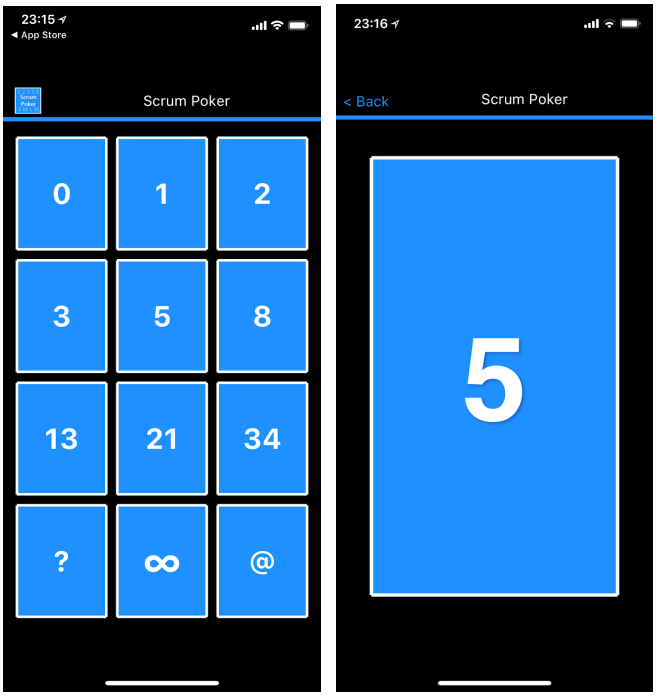


Imagen 28: capturas de la aplicación Scrum Poker

Una vez emitidos los votos, en caso de haber diferencias, cada miembro del equipo dio su punto de vista y se llegó a un consenso para finalmente asignar un valor de dificultad a la historia.

Durante esta reunión, también se fijaron los objetivos de cada *sprint*. El objetivo del *sprint* debería lograrse durante el sprint a través de la implementación de las historias de usuario seleccionadas y proporciona una guía al equipo de desarrollo de por qué se está construyendo el incremento

6.2.5. *Sprint retrospective*

Al finalizar cada *sprint*, y como parte del mismo, se realizó la ceremonia *sprint retrospective*. Durante esta instancia el equipo reflexiona sobre el sprint pasado, y

se brinda un espacio para que los miembros del equipo charlen sobre cosas que han ido bien, y cosas que quisieran cambiar. Para finalizar la reunión se definen una serie de *action items*, que son acciones que el equipo se compromete a ejecutar con el objetivo de mejorar el trabajo que se viene realizando. Luego, al comienzo de la próxima *retrospective* se comienza con un seguimiento sobre los *action items* pendientes. En caso de haberse ejecutado se marcan como completados. En caso contrario quedan pendientes para la próxima reunión retrospectiva.

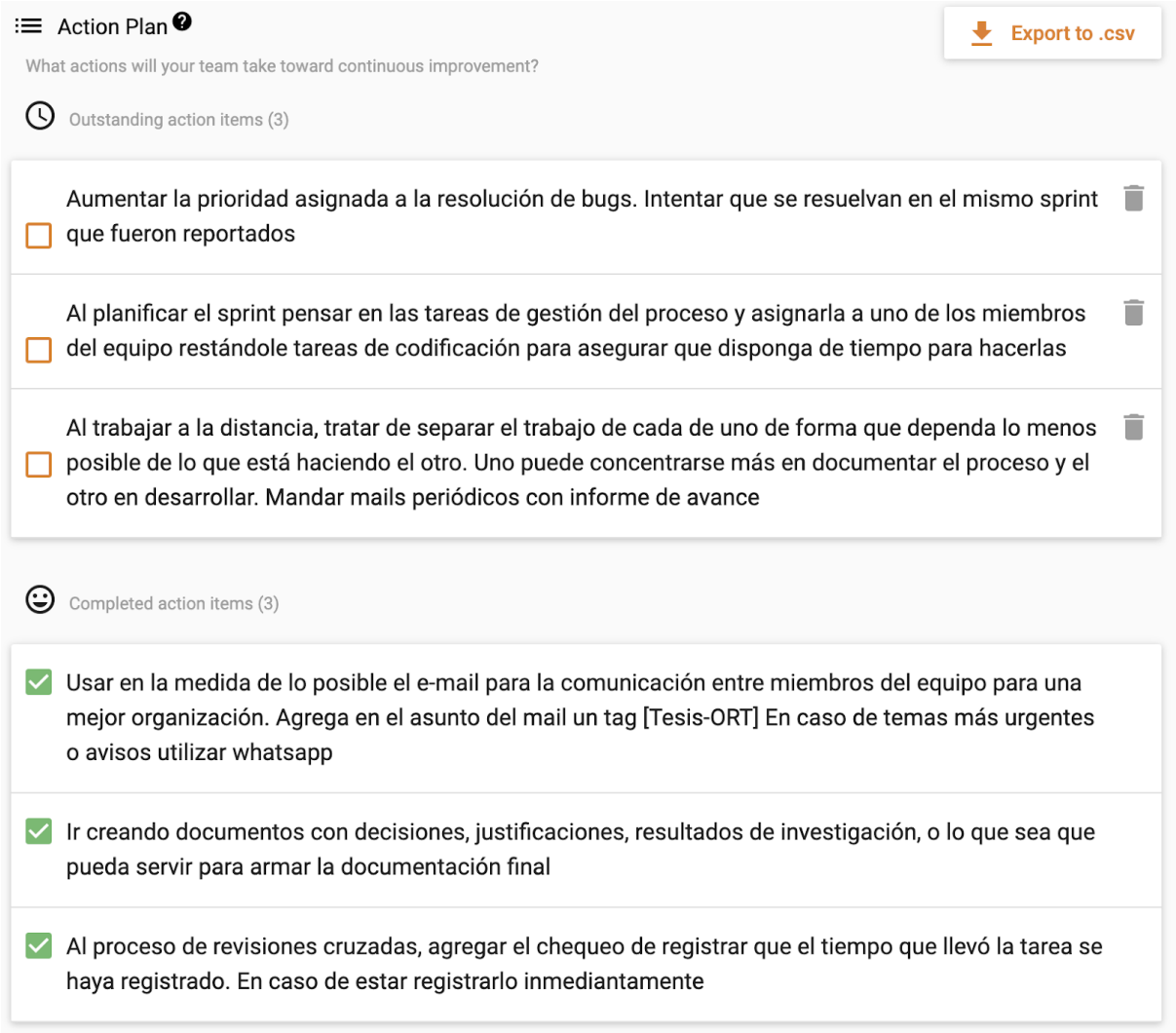


Imagen 29: captura de Retrium mostrando *action items* pendientes y completados

La herramienta elegida para realizar el seguimiento de estas reuniones fue Retrium, ya que ofrece una interfaz amigable y fácil de usar. Permitió al equipo realizar la reunión de forma remota, ofreciendo además distintas formas de llevar adelante la

ceremonia. Es posible ver su funcionamiento de mejor manera en el anexo 12.20. Sprint retrospectives con Retrium.

Adicionalmente, cada 2 sprints se incorporó a esta actividad la revisión de los riesgos del proyecto. Esto se describe en más detalle en el Capítulo 7. Gestión de riesgos.

6.2.6. *Daily meetings*

Tal como se aclara en el marco metodológico, el equipo no consideró necesario tener reuniones diarias para reportar el avance de cada uno. Al tratarse de un equipo de sólo dos integrantes, sin una disponibilidad horaria fija para dedicar al proyecto, se optó por mantener reuniones una vez por semana. La forma más habitual de tener estas reuniones fue por videollamada a través de *Google Hangouts*. En algunos casos los miembros del equipo se encontraron juntos por alguna razón al momento de la reunión y entonces se hizo en forma presencial. Alternativamente, si por alguna razón no fue posible reunirse a la hora establecida y no hubo posibilidades de cambiarla, entonces cada miembro del equipo comunicó su estado mediante un mensaje de *whatsapp*.

De cualquier forma ambos integrantes del equipo estuvieron en constante comunicación, incluso compartiendo clases en la facultad. Frente a cualquier inconveniente o novedad relevante no fue necesario esperar a la reunión semanal para reportarlo.

6.2.7. Gestión de *sprints*

Un aspecto de suma importancia de la gestión del proyecto fue poder realizar un seguimiento del trabajo realizado y trabajo restante. Para esto se seleccionó la herramienta VersionOne, la cual permitió tener un registro de todas las tareas pendientes en el backlog, así como el armado y planificación de sprints.

Imagen 30: captura de VersionOne donde se muestra el backlog pendiente

Durante los *sprints*, las historias se asignaron a una de las siguientes seis columnas:

- *Ready*: la historia está lista para ser desarrollada, es decir, que cumple con la *definition of ready* detallada anteriormente.
- *Dev in progress*: la historia está siendo desarrollada por un integrante del equipo de desarrollo.
- *Code review*: existen pull requests pendientes de código asociado a la historia.
- *QA ready*: la historia está lista para pasar por el proceso de aseguramiento de calidad.
- *QA in progress*: el proceso de aseguramiento de calidad para la historia se encuentra en transcurso.
- *Done*: la historia ha aprobado el proceso de aseguramiento de calidad y está finalizada.

En cuanto al registro del esfuerzo, se optó por usar la herramienta Toggl, en la cual cada miembro del equipo fue registrando hora de comienzo y hora de fin de cada tarea realizada en un día de trabajo. Esta herramienta permite además poner un título a cada registro, y una categoría. Esto último fue particularmente útil al momento de crear reportes donde se viera reflejado el porcentaje de esfuerzo dedicado en cada una de las categorías.

Imagen 31: captura de Toggl

En el caso de las tareas de desarrollo, se optó por poner como título el identificador que tiene en VersionOne la historia de usuario que se está trabajando.

6.2.8. Gestión del alcance

Por tratarse de un proyecto de duración fija (*date-driven*), a medida que surgieron nuevas funcionalidades y necesidad de realizar re-trabajo, la solución fue analizar el alcance del proyecto para asegurar la finalización en fecha.

Luego de la validación del primer *release* con Pablo San Nicolás, que incluyó el perfil de las instituciones académicas, surge la necesidad de incluir varias historias de usuario nuevas al proyecto. Una vez priorizadas estas historias, suman 18 *story points* al proyecto. Siendo imposible postergar la fecha de finalización del desarrollo para agregar nuevos sprints, ni aumentar la velocidad del equipo, se hace una etapa de repriorización. Durante esta etapa se seleccionan aquellas historias que se consideran imprescindibles en una primer versión del producto, y se seleccionan

también aquellas que se consideran de prioridad baja y pueden quedar para una próxima versión del producto. De estar forma se logra reducir en 17 story points el trabajo pendiente, siendo posible incluir al backlog todas las nuevas historias sin afectar la fecha de finalización del proyecto.

Las historias que se decidió incluir de la primer versión del producto fueron las siguientes:

- Archivos de requerimientos (3 puntos).
- Registro y edición de instituciones académicas (8 puntos).
- Carga de títulos mediante un archivo CSV (5 puntos).
- Tipo de título (2 puntos).

Las historias que se decidió excluir de la primer versión del producto fueron las siguientes:

- Notificación a administrador (2 puntos).
- Recordatorio a institución (2 puntos).
- Ingreso con Google (8 puntos).
- Restricciones de campos (5 puntos).

Es posible ver estas historias en mayor detalle en el anexo 12.12. Historias de usuario.

Imagen 32: backlog con las tareas que quedaron fuera de esta versión

Cabe aclarar que algunas de las nuevas historias que se incluyeron en la primera versión del producto con mayor prioridad, se desarrollaron luego de algunas historias ya existentes de menor prioridad. Estas nuevas historias no fueron desarrolladas antes debido a que no iban de acuerdo al objetivo del *release* en transcurso. Debido a esto, se determinó implementarlas luego de finalizado el segundo *release*, y de esta forma lograr el cumplimiento de su objetivo.

6.3. Métricas de la gestión

Es fundamental en cualquier proyecto de software poder saber en todo momento si el proyecto está siendo bien ejecutado, en tiempo y forma y si es necesario realizar ajustes para poder alcanzar los valores esperados de rendimiento. Con el objetivo de tener esta información disponible el equipo fue recolectando diferentes métricas a medida que avanzó la ejecución del proyecto.

6.3.1. Velocidad

Una vez creadas y estimadas todas las historias de usuario, se suman los story points de todas ellas. Este cálculo da que será necesario completar 134 story points para finalizar el desarrollo. El objetivo del equipo es terminar el desarrollo a mediados de diciembre, para disponer de tiempo para realizar la transferencia de conocimiento a Effectus y comenzar a trabajar en la documentación. Se calcula

entonces que hay espacio para completar 10 sprints de dos semanas de duración cada uno, entre la fecha de comienzo del desarrollo y la planificada de finalización. De acuerdo a estos valores, se estima la velocidad ideal del equipo varía entre 13 y 14 *story points* por sprint.

Se presenta a continuación una gráfica comparativa entre la velocidad ideal del equipo y la velocidad real.

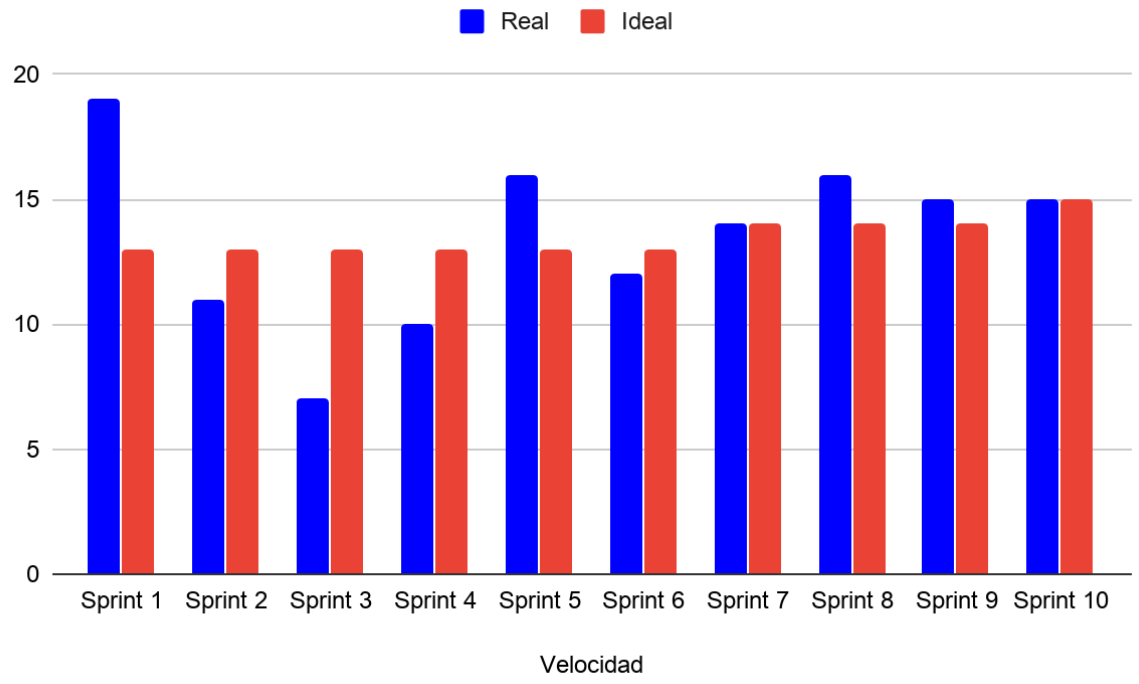


Imagen 33: velocidad del equipo

Se puede observar en la gráfica que la velocidad ideal estimada fue aumentando levemente durante el transcurso del proyecto. Luego de cada release y validación con usuarios, surgieron cambios y nuevos requerimientos. Como se menciona en 6.2.8. Gestión del alcance, frente a la imposibilidad de completar todas las historias a tiempo, se decidió re priorizar y seleccionar aquellas historias que se consideraron esenciales para una primer versión del producto. Las restantes historias quedaron especificadas y priorizadas en el backlog, para ser incluidas en una próxima versión. Para alcanzar el objetivo de terminar el desarrollo en fecha, fue necesario aumentar

la velocidad ideal durante los siguiente sprints. También fue necesario aumentar el esfuerzo para llegar a completar el trabajo en fecha.

Llama la atención también que durante el primer sprint la velocidad fue notoriamente mayor a la planificada. Diversos factores influyen en esta variación:

- Falta de experiencia del equipo al momento de realizar estimaciones. Algunas historias fueron estimadas por encima del esfuerzo que finalmente costaron.

- Entusiasmo y mayor dedicación por parte del equipo. Por tratarse del primer sprint el equipo comenzó a trabajar muy motivado dedicando más horas de las inicialmente estimadas.

Otra observación que se desprende de la gráfica es el hecho que en algunos sprints se observa un descenso en la velocidad del equipo, seguido por un aumento notorio en el sprint siguiente. Esto puede observarse entre los sprints 4 y 5, o 7 y 8. Esto ocurre debido a alguna historia de usuario que no alcanza a completarse en un sprint y se completa en el siguiente. En estos casos, se suma el total de los puntos de la historia al sprint en el cual se completó. De esta forma se ve reflejada una disparidad en el esfuerzo de ambos sprints, pero en la realidad se desarrolló una cantidad de trabajo pareja en ambos.

Una particularidad es que a diferencia de lo que suele ocurrir en el ámbito laboral, los integrantes del equipo no dedican una cantidad de horas fijas diaria ni semanalmente al proyecto. Existe un compromiso de dedicar entre 15 y 20 horas semanales cada uno, pero semana a semana esto puede variar debido compromisos laborales, personales, o académicos. Adicionalmente, cuando fue necesario completar ciertas funcionalidades antes de cierto release o revisión, se hizo el esfuerzo extra que fuera necesario para asegurar el cumplimiento de entrega en las fechas pactadas.

6.3.2. Burnup chart

El *burnup chart* permite observar cómo fue la evolución a lo largo del proyecto del trabajo realizado sprint a sprint respecto a la planificación inicial. Permite visualizar además los cambios que hubieron en el tamaño total del *backlog* del proyecto a lo largo del mismo.

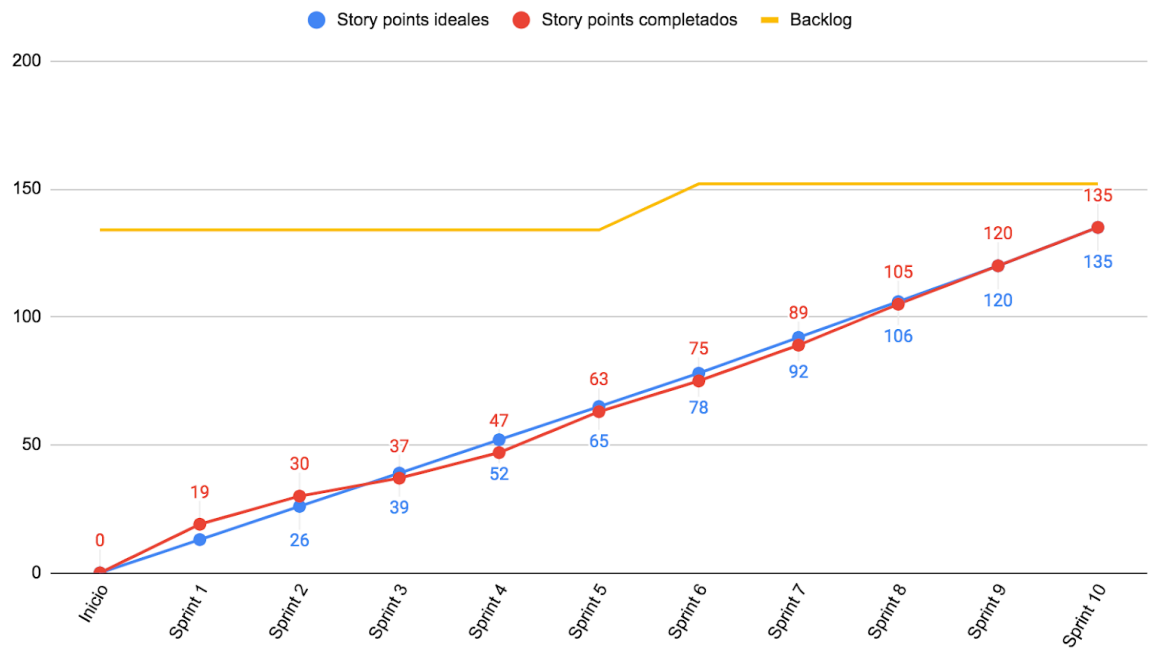


Imagen 34: burnup chart

Puede observarse como después del primer release, con el feedback obtenido, aumenta la cantidad de story points presentes en el *backlog* del proyecto debido al surgimiento de nuevos requerimientos y cambios propuestos. Inmediatamente se realiza una etapa de priorización donde se seleccionan las historias a incluir en la entrega final, y cuales dejar para una próxima versión de la aplicación. Es por esta razón que al final de la gráfica se puede apreciar un espacio entre los story points completados y el tamaño del *backlog*. Ese espacio es la representación de la sumatoria de los puntos de las historias que fueron dejadas para una próxima versión.

Del mismo modo se puede apreciar un aumento en el trabajo pendiente aunque en menor cantidad. Esto se debió a que, aunque se hayan re-priorizado las historias,

existió una diferencia de un punto entre las historias excluidas y aquellas historias incluidas en el alcance del proyecto. En la sección 6.2.8. Gestión del alcance de este capítulo puede verse el detalle de las historias que fueron excluidas de esta versión del producto.

Después del segundo release no surge la necesidad de incluir nuevas historias de usuario, por lo que una etapa de priorización no es necesaria y la planificación se mantiene.

6.3.3. Distribución del esfuerzo

Se presenta un gráfico que ilustra la distribución del esfuerzo del equipo en las diferentes áreas del proyecto en base a la cantidad de horas dedicadas.

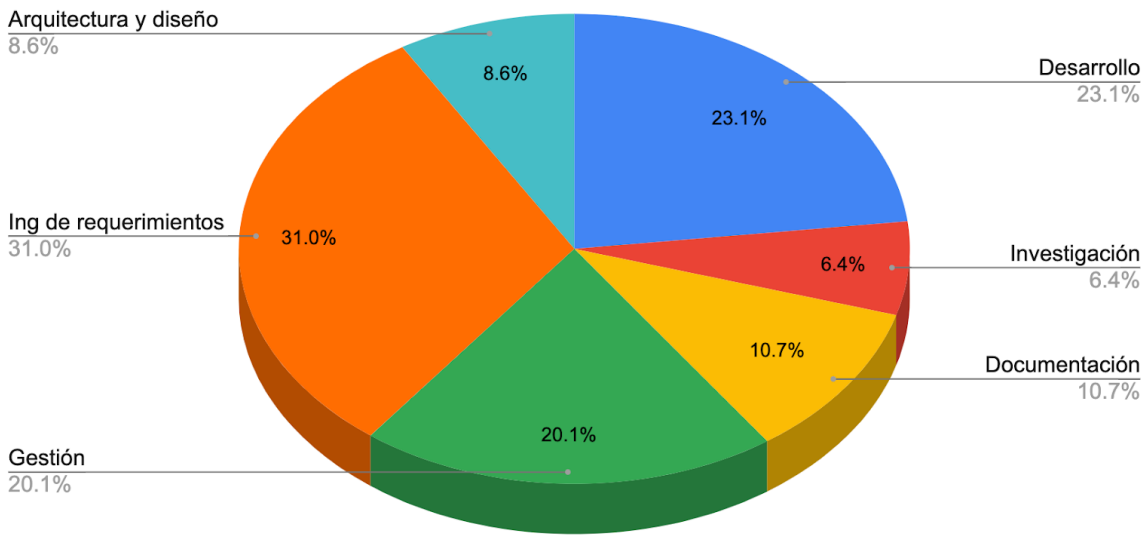


Imagen 35: distribución del esfuerzo

Puede observarse una distribución pareja entre las tareas de desarrollo, de gestión y de ingeniería de requerimientos. Esto es coherente dado el tipo de proyecto que se desarrolló, teniendo que dedicar buena parte del tiempo al comienzo del mismo al relevamiento de requerimientos, y siendo este un proyecto en el cual el equipo

debió autogestionarse. También es notoria una dedicación considerable a tareas de investigación, relacionado principalmente con el uso de una tecnología como blockchain con la que ninguno de los integrantes tenía experiencia previa.

Se detallan a continuación las actividades incluidas en cada una de las categorías.

- **Arquitectura y diseño:** definición y validación de la arquitectura del sistema. Diseño de cada uno de los componentes que la componen.

- **Ingeniería de requerimientos:** tareas realizadas con la finalidad de relevar requerimientos y escribir las historias de usuario a incluir en la aplicación desarrollada. Incluye la realización de entrevistas, encuestas, realización de prototipos y validación de los mismos.

- **Gestión:** tareas relacionadas a la gestión y seguimiento del proyecto. Incluye el registro de horas trabajadas, reuniones del equipo y con el tutor, gestión del *backlog* y *sprints*, registro de incidentes y seguimiento de su resolución.

- **Desarrollo:** principalmente tareas de codificación de nuevas funcionalidades y resolución de errores encontrados. Incluye también tareas de configuración de ambientes de trabajo, *deploy*, y *releases*.

- **Investigación:** lectura de bibliografía especializada sobre el tema a investigar, cursos online, artículos de internet, y realización de prototipos para validar el funcionamiento de los investigado. Documentación de los resultados obtenidos.

- **Documentación:** registro escrito del desarrollo del proyecto desde el comienzo. Armado del documento a entregar a Effectus como transferencia del conocimiento, y del documento a presentar como entrega final en la universidad.

6.4. Conclusiones

La adaptación del marco de trabajo Scrum fue de gran utilidad debido a varias razones. En primer lugar, respetar la *definition of ready* fue esencial para el correcto desarrollo de las historias. Esto se debió a que cada vez que se comenzaba a desarrollar una historia, el integrante del equipo de desarrollo podía estar seguro de que no sólo la historia agregaba valor, sino que también contaba con un conjunto de criterios de aceptación en forma de escenarios que podían ser utilizados para comprobar que la historia haya sido terminada. Esto sumado a la *definition of done* facilitaron de gran manera el proceso de construcción de software, y evitaron que las historias pasen de las columnas *QA* a *In Progress* a debido a la no finalización de las mismas.

En segundo lugar, contar con las ceremonias de este marco de trabajo le dieron la oportunidad al equipo de mejorar la coordinación y la comunicación, estableciendo horarios fijos de reuniones en las cuales se discutía el avance del proyecto y el trabajo pendiente. Estas fueron particularmente importantes durante los tramos del proyecto en los que el equipo no se encontraba en el mismo país, existiendo una diferencia horaria de cinco horas entre los dos integrantes. Durante estos períodos, la comunicación entre el equipo disminuía, y estas ceremonias fueron la principal razón por la cual se mantuvo un nivel de coordinación aceptable.

Como fue mencionado previamente, otra gran ventaja que el marco de trabajo Scrum le brindó al equipo fue la de permitir adaptarse a los cambios en los requerimientos ocurridos, y a través de la gestión del alcance, incorporar aquellas nuevas historias de mayor prioridad al alcance del proyecto a medida que surgieron.

Por último, en cuanto a la herramienta utilizada para realizar la gestión del proyecto, el equipo no quedó conforme con VersionOne. Se consideró que esta herramienta es demasiado compleja y poco amigable en comparación con otras opciones utilizadas en otros ámbitos.

7. Gestión de riesgos

"Un riesgo es un evento o condición incierta que, si sucede, tiene un efecto en por lo menos uno de los objetivos del proyecto" [36]

El trabajo realizado con la gestión de riesgos resultó clave en la gestión del proyecto debido a la incertidumbre que el equipo enfrentó desde el inicio. El propósito de la gestión de riesgos fue prevenir la desviación de lo planificado de forma de no sufrir un impacto negativo en el proyecto y poder culminar el mismo exitosamente. En la siguiente sección se profundizará en los procesos que se utilizaron para llevar a cabo la planificación, identificación, análisis, seguimiento, y los planes de respuesta de los riesgos del proyecto.

7.1. Identificación

Para lograr identificar los riesgos del proyecto se utilizó la técnica *brainstorming*. Esto permitió discusiones interesantes sobre los mismos. Una vez identificados los riesgos, se procede a desarrollar un análisis detallado de cada uno, de forma de que quede claro para todos los integrantes del equipo.

R1 - Difícil acceso a la información

Razones:

- Se trata de un proyecto académico, en el cual varias de las fuentes de información más importantes son expertos en el dominio, y el acceso a estas fuentes depende de la disponibilidad y voluntad de dichos expertos.
- Ninguno de las integrantes del equipo ha pasado por el proceso de obtener un diploma y presentarlo en un trabajo.
- En el caso de blockchain, se trata de una tecnología relativamente nueva sobre la cual no hay demasiado información publicada ni muchos expertos en la materia.

Formas en las que afecta:

- Puede dificultar la comprensión del dominio o visualización de la solución de forma clara.
- Puede resultar difícil definir o validar los requerimientos.
- Es difícil evaluar aspectos técnicos, por ejemplo, la arquitectura del sistema.
- Puede llevar a errores en la implementación de los requerimientos.

Etapa del proyecto que afecta:

- Principalmente en la etapa de *discovery*, ya que podría resultar en una incorrecta definición de los requerimientos.
- Etapas de validación, si no es posible validar el trabajo realizado con los interesados a tiempo esto puede resultar en retrabajo cuando finalmente se valide.
- Etapa de desarrollo, al tener que desechar trabajo realizado debido a la falta de validación y realizar re-trabajo sobre lo desechado.

R2 - Integrante del equipo abandona

Razones:

- Se trata de un equipo de 2 personas, cada uno de los integrantes es indispensable.
- Razones de fuerza mayor, oportunidad única de trabajo en el exterior, motivos familiares. Todo esto puede llevar a que uno de los integrantes decida no seguir adelante con el proyecto.

Formas en las que afecta:

- Queda sólo un integrante en el equipo. Dependiendo la etapa en que se encuentre el proyecto, puede llegar a implicar el fracaso del mismo.

Etapa del proyecto que afecta:

- Cuanto más temprano sea el abandono, más graves las consecuencias.

R3 - Escaso conocimiento de las tecnologías a utilizar

Razones:

- Se va a trabajar con una tecnología como blockchain, la experiencia de los integrantes del equipo con esta tecnología es mínima.
- Poco conocimiento de parte de los integrantes del equipo de desarrollo *front-end*.

Formas en las que afecta:

- Se deberá dedicar tiempo a investigar y entender las tecnologías que no se conocen o con las que algún integrante del equipo no se encuentre familiarizado
- Mayor probabilidad de ocurrencia de errores y, por lo tanto, de retrabajo
- Baja velocidad del equipo

Etapa del proyecto que afecta:

- Afecta la etapa de *delivery,* que es cuando comenzarán a implementarse los requerimientos con el uso de estas tecnologías

R4 - Indisponibilidad de miembros del equipo

Razones:

- Cada integrante del equipo tiene un trabajo diario y materias pendientes que cursar en la facultad

- Los integrantes del equipo se toman vacaciones en épocas diferentes

Formas en las que afecta:

- Tareas que dependan de otra tarea asignada a un integrante que no esté disponible no se podrán completar hasta que este integrante vuelva a estar disponible

- Planificar sprints acorde a la disponibilidad de cada uno

Etapa del proyecto que afecta:

- Principalmente la etapa de *delivery,* donde es habitual que algunas tareas dependan de un trabajo realizado previamente por alguno de los integrantes del equipo

7.2. Análisis cualitativo y cuantitativo

Una vez identificados los riesgos, se llevó a cabo un análisis inicial según las siguientes variables:

- **Probabilidad de ocurrencia (P)**: valor numérico que indica la probabilidad de que el riesgo se materialice. A cada riesgo se le asignó un valor entre 0 y 1, de acuerdo a la siguiente escala:

  Probabilidad:
  0.0 - No probable
  0.2 - Poco probable
  0.4 - Probable
  0.6 - Muy probable
  0.8 - Altamente probable
  1.0 - Se convierte en problema

- **Impacto sobre el proyecto (I)** - valor numérico que indica qué tan perjudicial resultaría el riesgo para el proyecto en caso de materializarse. A cada riesgo se le asignó un valor del 0 al 5, de acuerdo a la siguiente escala:

  Impacto:
  0 - Ninguno
  1 - Marginal
  2 - Poco importante
  3 - Importante (puede retrasar el proyecto)
  4 - Crítico (puede detener el proyecto)
  5 - Catastrófico (fracaso del proyecto)

Luego, a través de las variables mencionadas se calculó la magnitud de cada riesgo:

Magnitud (M) = Probabilidad de ocurrencia (P) * Impacto sobre el proyecto (I)

Para cada uno de los riesgos se definió un plan de respuesta. El mismo permitió definir qué acciones se tomarían frente al riesgo identificado.

Se intentó definir soluciones para cada uno de los riesgos identificados con la finalidad de evitar que se conviertan en problemas para el proyecto. Si no es posible encontrar una solución serán considerados problemas. Teniendo en cuenta la lista de riesgos se define un plan de mitigación y medidas de contingencia.

- **Plan de mitigación:** Tiene como finalidad reducir la probabilidad de ocurrencia del riesgo

- **Medida de contingencia:** Plantea una solución al problema generado por la materialización del riesgo

Luego del análisis inicial de los posibles riesgos identificados, se llega a la siguiente tabla:

Riesgo	Impacto (I)	Probabilidad (P)	Magnitud (I x P)	Mitigación	Contingencia
R1	4	0,4	1,6	Buscar fuentes de información alternativas. Consultar en otros institutos	Obtener la información necesaria de parte de alguno de los interesados con los que sí tengamos contacto
R2	4	0,2	0,8	Realizar un reglamento con directrices que ayuden a mantener el orden y disciplina dentro del equipo	Hablar con el integrante e intentar hacerlo cambiar de opinión. En caso de no haber vuelta atrás, redistribuir las tareas para poder cumplir con las fechas estipuladas
R3	3	0,8	2,4	Contacto con referentes de la tecnología a utilizar	Invertir tiempo en investigación y capacitación sobre la tecnología nueva
R4	3	0,5	1,5	Definir anticipación con que se debe notificar si alguno de los integrantes del equipo no va a estar disponible por un cierto período de tiempo	Replanificar las tareas a realizar durante el tiempo que alguno de los integrantes no esté disponible.

Tabla 9: evaluación inicial de riesgos

Con el resultado del análisis cuantitativo se realiza una gráfica en forma de cuadrícula con la finalidad de observar cuales son los que pueden tener un mayor impacto sobre el proyecto.

La cuadrícula se presenta en tres colores: verde, amarillo y rojo. Se le pondrá especial atención a los riesgos ubicados en el cuadrante amarillo a la hora de realizar el seguimiento de los mismos. Por otro lado, los riesgos ubicados sobre el cuadrante rojo (impacto mayor a 2,5 y probabilidad mayor a 0,5) son los que tienen más posibilidades de materializarse en un problema de mayor magnitud, por lo que cualquier riesgo que entre en este cuadrante será enfrentado con acciones de mitigación y, en caso de que estas no tengan los efectos esperados, se aplicará el plan de contingencia.

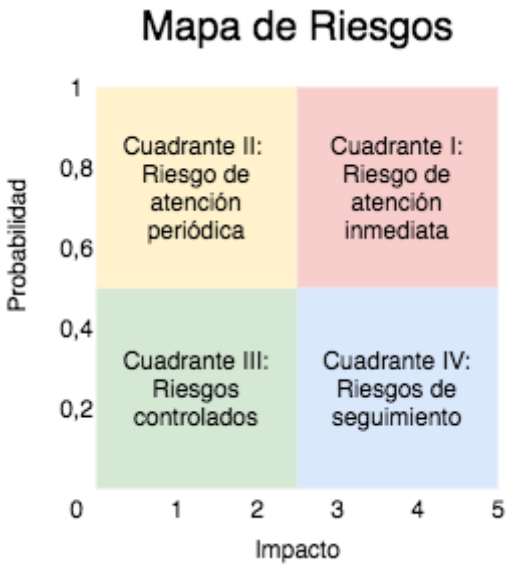


Imagen 36: mapa de riesgos

En la siguiente cuadrícula se puede observar la apreciación inicial de los distintos riesgos identificados y analizados por el equipo.

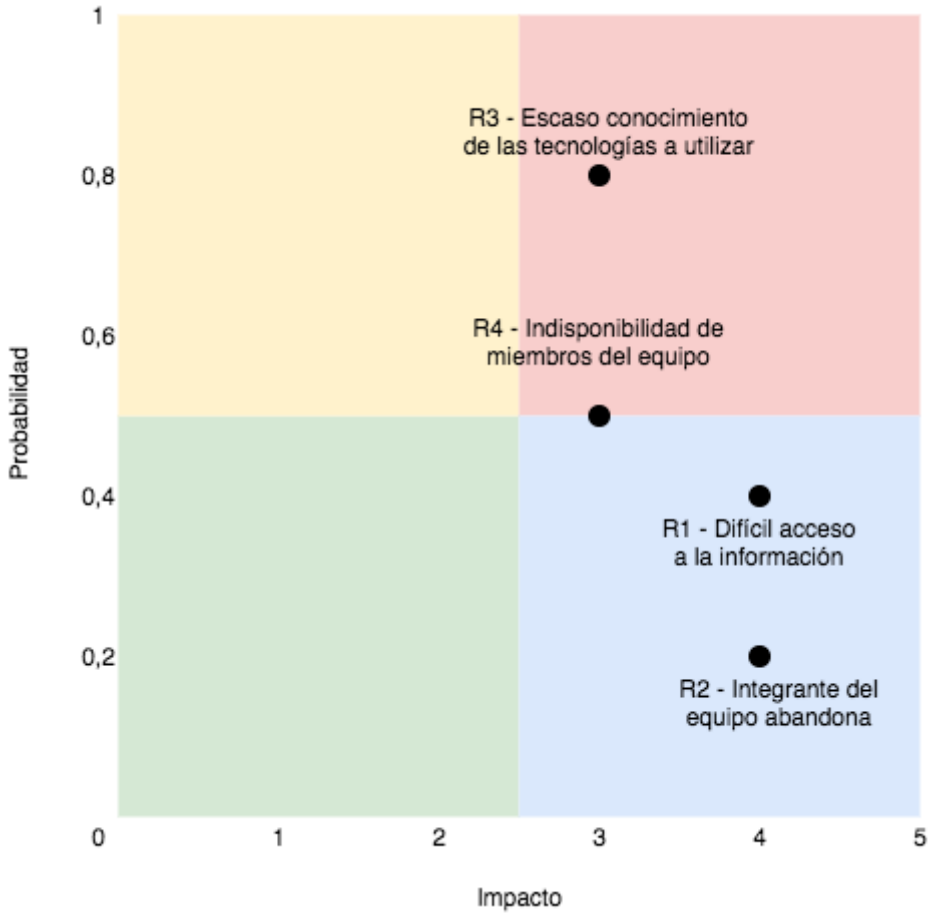


Imagen 37: riesgos por cuadrante

Según se observa en la cuadrícula, el riesgo de mayor magnitud al comienzo del proyecto es el escaso conocimientos de las tecnologías a utilizar por parte del equipo. Se pone en marcha el plan de mitigación, contactando expertos en blockchain y estudiando la tecnología de diversas fuentes. También el riesgo de indisponibilidad de alguno de los miembros del equipo aparece en la zona roja, esto debido a que ambos integrantes del equipo se encuentran trabajando y cursando materias en la universidad lo cual conlleva tiempo de estudio. Se acuerda compartir un calendario donde cada uno marque las fechas en que tiene entregas o parciales, y la cantidad de días necesarios para estudiar para esas instancias, durante los cuales no va a estar disponible para trabajar en el proyecto.

7.3. Seguimiento

Se define un plan de seguimiento y control de los riesgos, siendo este un proceso continuo a lo largo de todo el ciclo de vida del proyecto. Una vez por mes (cada dos sprints) se realizó una re-evaluación de los riesgos, evaluando si cambiaron su nivel de impacto o probabilidad de ocurrencia. En base al resultado de esta evaluación se optó por ejecutar las medidas de contingencia, si el riesgo se había materializado, o actualizar el plan de mitigación en caso de cambios significativos en el valor del impacto del riesgo.

A continuación se presenta una gráfica con la evolución de los riesgos a lo largo de todo el proyecto, y el detalle del seguimiento de cada uno, con la magnitud mes a mes.

	ABR	MAY	JUN	JUL	AGO	SET	OCT	NOV	DIC	ENE	FEB
Difícil acceso a la información	1.6	1.6	3.2	3.6	3.6	3.6	1.6	1.6	2.4	1.6	1.6
Integrante del equipo abandona	0.8	0.8	0.8	0.8	0.4	0.4	0.4	0.4	0.2	0.2	0.2
Escaso conocimiento de la tecnología	2.4	2.4	2.4	1.8	1.2	0.6	0.3	0.3	0.3	0.3	0.3
Indisponibilidad de miembros del equipo	1.2	1.2	1.2	3.2	4	2.4	0.6	0.6	2.4	2.4	0.6

Tabla 10: evolución de la magnitud de los riesgos

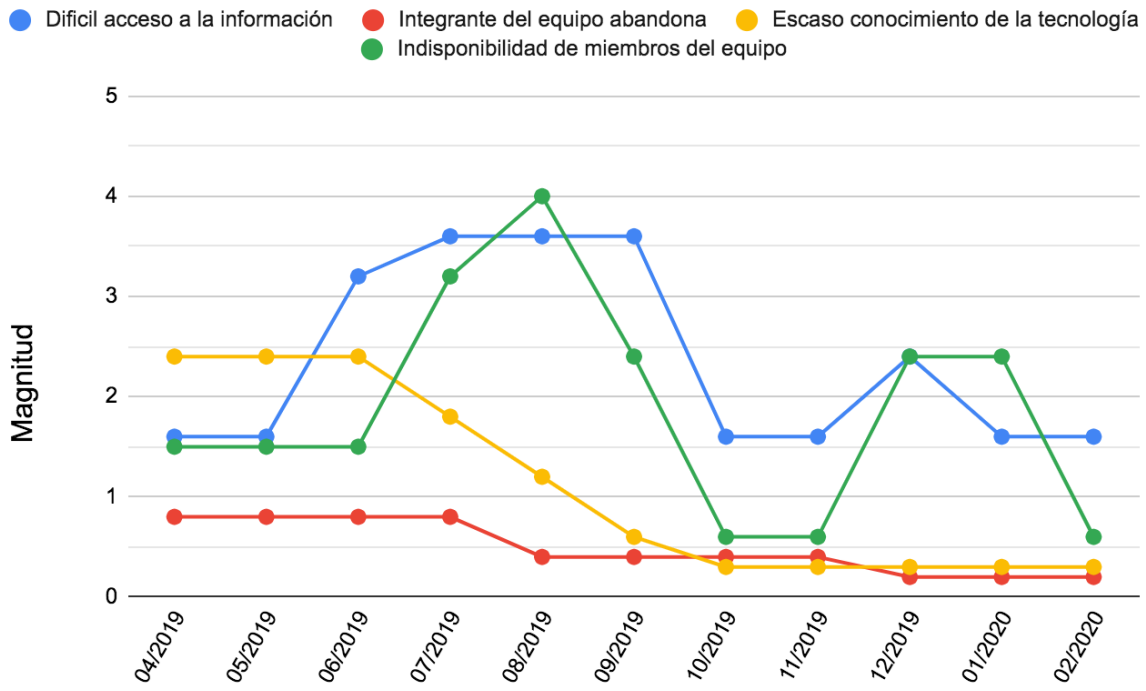


Imagen 38: evolución de la magnitud de los riesgos

7.3.1. R1 - Difícil acceso a la información

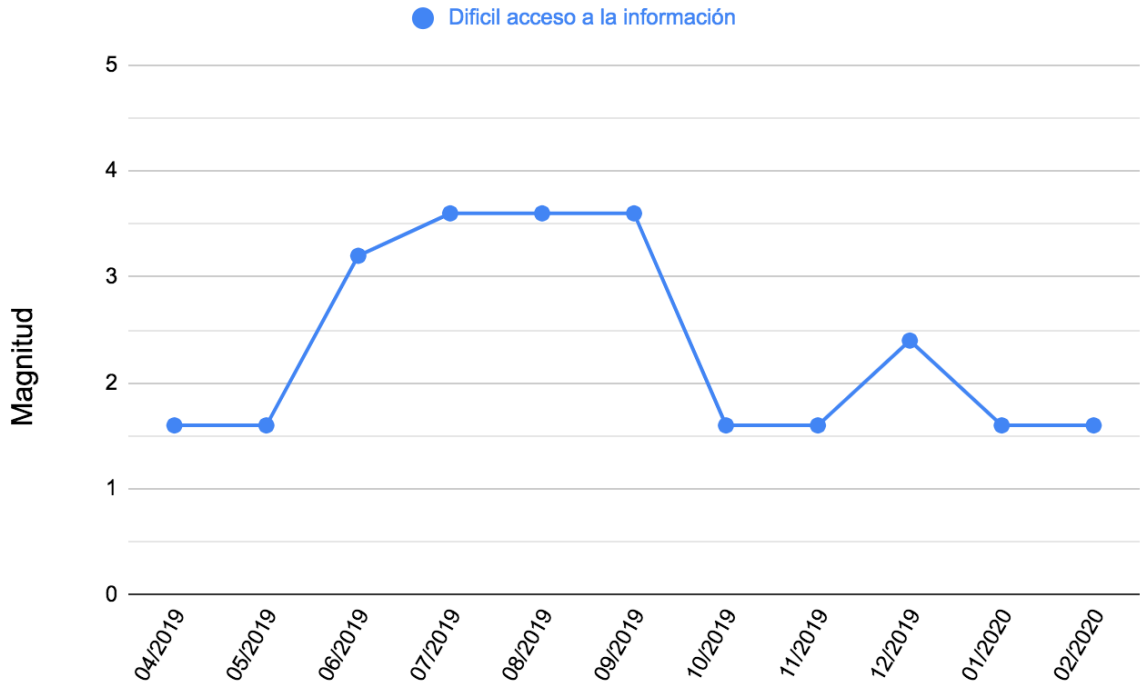


Imagen 39: evolución de la magnitud del riesgo R1

Al comienzo del proyecto, durante la etapa de *discovery*, se fue en busca de aquellas personas o fuentes de datos que pudieran proveer información sobre el proceso de emisión, registro y validación de diplomas. Una vez identificados los actores se estableció contacto y se intentó coordinar entrevistas personalmente, telefónicamente, o incluso por correo electrónico.

A medida que el equipo fue ganando un mejor entendimiento acerca del dominio del problema, quedó claro que la experta en el dominio que se había definido inicialmente no se adecuaría al dominio en cuestión. Esto se debió a que ella es integrante de *ULearn*, una academia que realiza capacitación a distancia de distintos cursos. Al investigar más acerca de esta academia se descubrió que la gran mayoría de los cursos que ofrece cuenta con certificados digitales, por lo que el sistema a desarrollar no sería necesario para esos casos.

Este descubrimiento hizo que el equipo careciera de un experto en el dominio que represente a las instituciones académicas, por lo que por un segmento del proyecto este riesgo se materializó, y el equipo no tuvo con quien validar las secciones del sistema que corresponden a instituciones académicas. Este tramo puede observarse en la gráfica, extendiéndose hasta ya finalizada la etapa de validación. Al momento de iniciar el desarrollo con el comienzo del *delivery cycle*, el equipo aún no había logrado validar dicha sección.

Con el fin de mitigar este riesgo, se optó por utilizar la estrategia de mitigación definida en la sección anterior de análisis cualitativo y cuantitativo, y se buscaron fuentes de información alternativas. Eventualmente se logró coordinar la validación de la sección del sistema faltante con Pablo San Nicolás, secretario de gestión académica de ORT, quien ya había sido previamente entrevistado. Una vez coordinada y realizada dicha validación, disminuyó el impacto del riesgo, aunque lo sucedido tuvo como consecuencia que el equipo ajustara el riesgo aumentando de la probabilidad del mismo, y por consiguiente aumentando también su magnitud. La materialización del riesgo en esta etapa del proyecto tuvo como consecuencia la

necesidad de realizar re-trabajo sobre el perfil y registro de las instituciones académicas.

También durante la etapa de *discovery* se materializa este riesgo frente a la imposibilidad de lograr una reunión con actores del Ministerio de Educación y Cultura. Nuevamente, se mitiga este riesgo recurriendo a la estrategia de mitigación definida en la sección anterior, accediendo a fuentes alternativas. En este caso, se obtiene una explicación del funcionamiento del MEC a través de Pablo San Nicolás, secretario de gestión académica y de Adriana Fernández y Victoria González, de la oficina graduados de ORT. De todos modos, con el fin de evitar que este riesgo impacte de manera negativa en el proyecto, se decide dejar la parte del proceso que involucra al ministerio por fuera del alcance del proyecto, obteniendo la información necesaria de los demás actores involucrados.

Finalmente en los últimos meses del proyecto se observa un nuevo aumento en la magnitud del riesgo. Esto se debe a que era necesario realizar una última validación del trabajo realizado con los actores involucrados y, tal vez por tratarse de una época de mucho trabajo para las instituciones educativas, se hizo difícil coordinar reuniones. Eventualmente se realizaron las validaciones correspondientes y el riesgo vuelve a bajar hacia el final del proyecto.

7.3.2. R2 - Integrante del equipo abandona

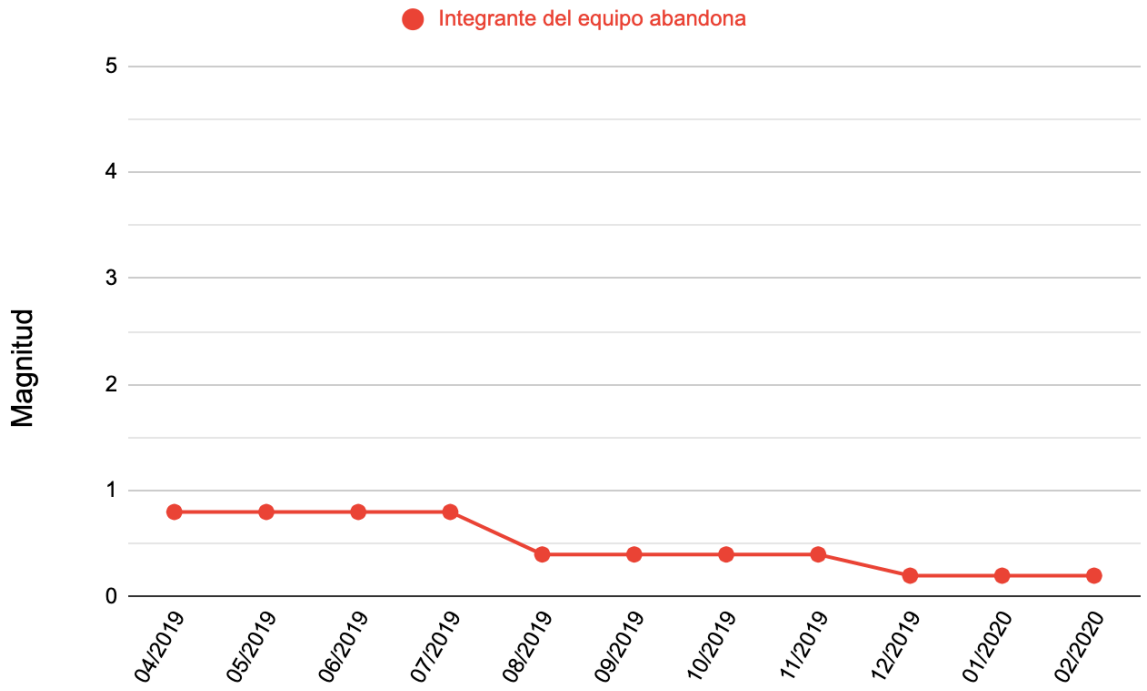


Imagen 40: evolución de la magnitud del riesgo R2

Desde un comienzo la magnitud de este riesgo fue bastante baja. Esto se explica debido a que la probabilidad de ocurrencia tuvo un valor bajo durante todo el transcurso del proyecto, a pesar que el impacto en caso de materializarse el riesgo comenzó siendo bastante alto.

Al tratarse de un equipo de sólo dos integrantes, el abandono del proyecto por parte de alguno de los miembros del equipo en los primeros meses de trabajo comprometería seriamente las posibilidades de llevar adelante el trabajo restante. A medida que transcurre el tiempo y se va avanzando en el desarrollo del proyecto, el impacto en caso de abandono de uno de los integrantes se reduce ya que se disminuye cada vez más el trabajo restante para terminar.

En cuanto a la posibilidad de ocurrencia, desde un comienzo se mantuvo en un valor bajo teniendo en cuenta los antecedentes y la situación personal de cada uno de los integrantes. Se trata de dos alumnos con un promedio académico mayor a 80 sobre 100, que han realizado la carrera en tiempo y forma hasta el momento, y

ambos se encuentran comprometidos a terminar de cursar las materias restantes para llegar a obtener el título con la entrega del proyecto. Si bien siempre cabe la posibilidad que por determinadas circunstancias inesperadas alguno deba abandonar el proyecto, a medida que pasa el tiempo y se aproxima la fecha de entrega final se reduce esta probabilidad. Cualquiera de los integrantes haría lo posible por terminar el trabajo comenzado, y no tener que volver a comenzar de cero más adelante.

7.3.3. R3 - Escaso conocimiento de las tecnologías a utilizar

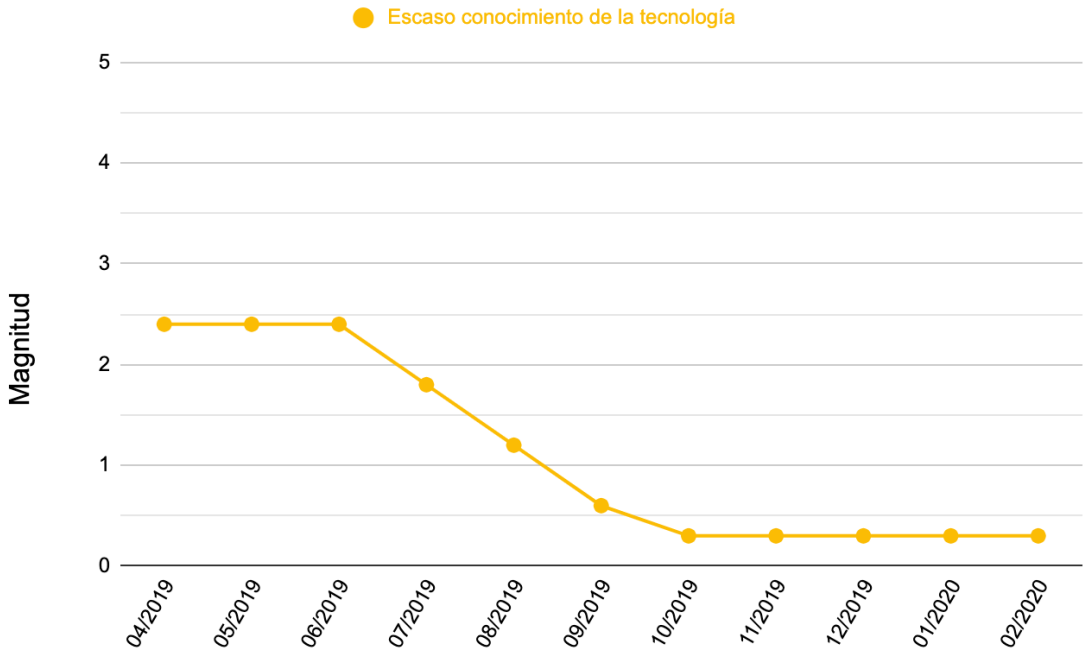


Imagen 41: evolución de la magnitud del riesgo R3

Al momento de comenzar el proyecto se identifica este riesgo como el de mayor magnitud, debido principalmente a la decisión de utilizar tecnología blockchain para el almacenamiento de la información. En primer lugar ninguno de los integrantes tiene experiencia previa ni conocimientos sobre este tipo de tecnologías, en segundo lugar se trata de un área relativamente nueva en el desarrollo de software y no abundan las fuentes de información.

Se pone en marcha el plan de mitigación del riesgo, por lo que se recurre al juicio experto, logrando tener una charla con Alejandro Narancio, CEO de Infuy, una empresa uruguaya referente en blockchain. Esta charla brinda al equipo una nueva perspectiva acerca de cómo aplicar blockchain al dominio en cuestión, aunque el equipo aún carece de conocimientos técnicos. Por esta razón, el equipo recurre al plan de contingencia, y ambos integrantes del equipo se inscriben en la materia electiva "Tecnologías blockchain para contratos inteligentes" durante el primer semestre de 2019 con el objetivo de capacitarse.

Con el correr de los meses, el equipo se va fortaleciendo en el conocimiento de blockchain por lo que la magnitud del riesgo comienza a descender gradualmente hasta salir de la zona crítica.

7.3.4. R4 - Indisponibilidad de miembros del equipo

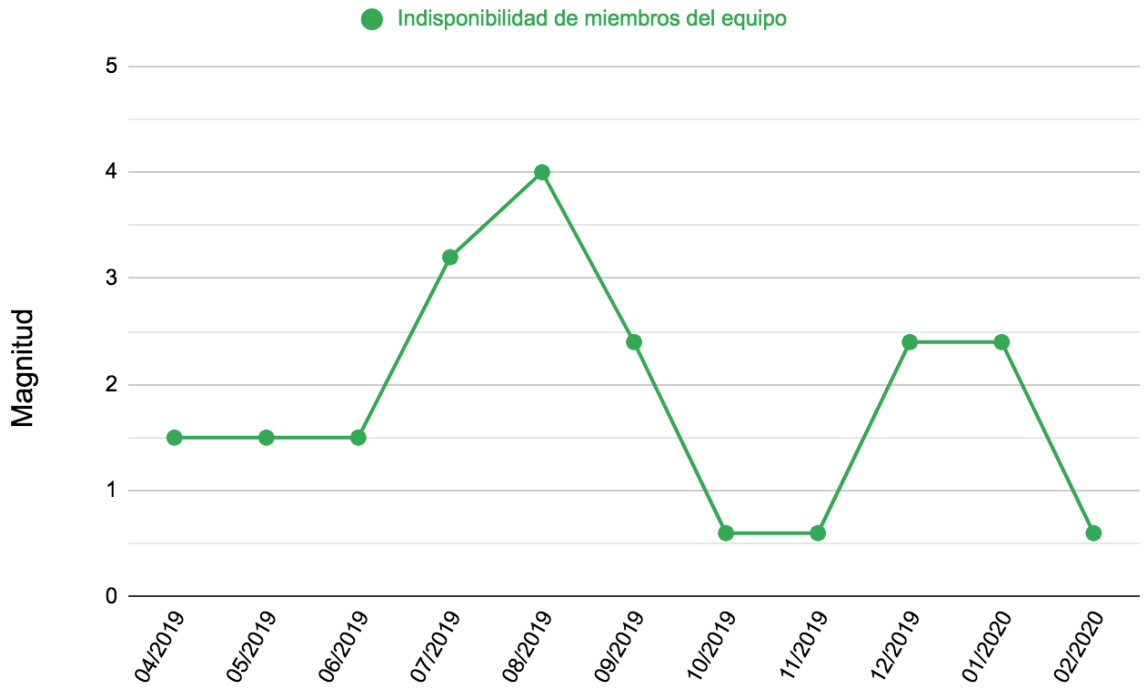


Imagen 42: evolución de la magnitud del riesgo R4

Por tratarse de un equipo de dos personas, la indisponibilidad de uno de los integrantes tiene un impacto alto ya que se pierde el 50% del esfuerzo durante ese período. Teniendo en cuenta que ambos integrantes trabajan y además deben

cursar materias en la universidad en paralelo con el desarrollo del proyecto, es esperable que existan períodos de indisponibilidad.

Para mitigar este riesgo se crea un calendario compartido en donde cada integrante registra los días durante los cuales no va a estar disponible. De esta forma se puede tener esta información en cuenta al momento de planificar los sprints.

Durante los meses de julio y agosto uno de los integrantes del equipo viajó a Europa, lo que provocó que el riesgo se materialice, dificultando la coordinación entre miembros del equipo. Durante las primeras dos semanas del viaje, estuvo de vacaciones, luego retomó el trabajo pero desde Europa. Esta situación lleva a que su disponibilidad no sea completa, ya que al estar cada integrante del equipo en un huso horario diferente pierde fluidez la comunicación. Tampoco está disponible el integrante en Europa para participar de reuniones presenciales con los interesados del proyecto.

En este caso, por haberse materializado el riesgo, se recurre al plan de contingencia. Se analiza la planificación de tareas existente y se hacen los ajustes necesarios para contemplar la ausencia de unos de los integrantes. Uno de los objetivos de esta replanificación es tener la mayor independencia posible entre las tareas que tome cada uno de los miembros del equipo, debido a la dificultad para trabajar en conjunto.

7.4. Conclusiones

En primer lugar, teniendo en cuenta lo detallado anteriormente, se puede establecer que se identificaron correctamente los riesgos del proyecto. Esto se debe a que se identificaron riesgos relevantes para el proyecto, pues en varios casos aquellos riesgos identificados se materializaron, llegando hasta a impactar en el proceso de construcción del sistema. Tal fue el caso del escaso conocimiento de las tecnologías, la indisponibilidad de un miembro del equipo, y el difícil acceso a la información, este último habiendo tenido consecuencias negativas en el transcurso del proyecto.

En segundo lugar, cabe destacar que las técnicas de mitigación de riesgos y los planes de contingencia fueron definidos y ejecutados exitosamente, logrando minimizar el impacto de los riesgos en el proyecto. En el caso del difícil acceso a la información, tener una estrategia de mitigación fue vital para evitar que una sección del sistema quede sin validar, y en el caso de la indisponibilidad del miembro del equipo, redistribuir las tareas de modo de que haya cierta independencia entre aquellas asignadas a distintos integrantes del equipo fue crucial para mantener el ritmo de desarrollo. Finalmente, la aplicación del plan de contingencia frente al riesgo de escaso conocimiento de las tecnologías permitió al equipo reducir la magnitud de un riesgo que podría haber tenido consecuencias graves en el proyecto.

En síntesis, es posible concluir que la gestión de riesgos jugó un rol clave en el éxito del proyecto, ya que mediante la identificación de riesgos e ideación de estrategias de mitigación y contingencia, el equipo logró enfrentar la materialización de los distintos riesgos de forma ordenada y sistemática.

8. Gestión de la calidad

El objetivo de esta sección es presentar todas las decisiones, acciones y procesos realizados por el equipo con el fin de asegurar la calidad tanto del producto generado, como del proceso seguido a lo largo del proyecto

8.1. Objetivos

Con el fin de asegurar la calidad, se definieron los siguientes objetivos tanto a nivel de producto como de proceso.

8.1.1. Producto

Los objetivos definidos a nivel de producto por parte del equipo son:

- Asegurar el cumplimiento de los requerimientos funcionales y no funcionales relevados.
- Asegurar que el producto final brinde soluciones y sea de valor para los usuarios.
- Validar el uso de tecnología blockchain para garantizar la integridad y validez de la información almacenada.

Como criterio de éxito, se considera tener un producto con la totalidad de los requerimientos definidos como prioritarios implementados, validado con los distintos tipos de usuario para asegurar que brinda valor, y finalmente poder elaborar un reporte sobre las fortalezas y debilidades del uso de blockchain en base a la experiencia adquirida.

8.1.2. Proceso

El principal objetivo en cuanto al proceso es que las actividades definidas contribuyan a la construcción de un producto de calidad, y a maximizar la eficiencia de los integrantes del equipo. Para lograr esto se deben respetar los estándares

definidos, mantener una velocidad de trabajo constante, identificar problemas y soluciones, y cumplir con las ceremonias establecidas en el marco metodológico.

8.2. Plan de calidad

Para poder cumplir de forma exitosa con los objetivos previamente mencionados, al comenzar el proyecto se definió un plan de calidad donde se establecieron las distintas actividades que se debían cumplir en cada fase. El objetivo de este plan es tener una hoja de ruta clara de las actividades que es necesario llevar a cabo para llegar a obtener un producto de calidad al final del proyecto, y del rol encargado de llevarlas a cabo.

Fase	Actividad	Entrada	Salida	Roles
Investigación	Capacitación en blockchain	Curso "Tecnologías blockchain para contratos inteligentes" en universidad ORT	Trabajo obligatorio realizado, donde se implementa un contrato inteligente corriendo sobre la red Ethereum	Discovery team
Ingeniería de requerimientos	Entrevistas con expertos	Reunión con expertos en el área de registro y/o validación de diplomas.	Información detallada en cuanto al proceso seguido para la emisión, registro, y validación de diplomas	Discovery team
	Realización de encuestas	Encuesta creada en un *google form*, enviada a personas que se hayan graduado de alguna carrera.	Colección de respuestas para analizar y extraer problemáticas actuales y requerimientos deseados	Discovery team

	Entrevistas con usuarios	Reunión con potenciales usuarios pertenecientes a los distintos tipos de interesados (egresados y o instituciones académicas).	Información detallada que permita comprender los distintos perfiles de cada tipo de usuario	Discovery team
	Análisis de documentos	Documentaciones de sistemas existentes, documentos legales, etc.	Documentos que sinteticen el funcionamiento de dichos sistemas, así como también los aspectos legales relevantes.	Discovery team
	Procesamiento de información relevada	Información relevada durante entrevistas con expertos, encuestas, entrevistas con usuarios, y análisis de documentos.	Perfil de cada tipo de usuario, con las tareas, dolores, y ganancias de cada uno.	Discovery team
	Sesiones de brainstorming	Perfil de cada tipo de usuario, con las tareas, dolores, y ganancias de cada uno	Mapa de valor que genere aliviadores de dolores y creadores de ganancias que encajan con aquellas tareas, dolores y ganancias de los usuarios.	Discovery team
	Prototipado con prototipos desechables	Mapa de valor generado	Funcionalidades del sistema validadas	Discovery team
	Especificación de	Funcionalidades del sistema	Backlog con historias de	Discovery team y

	requerimientos	validadas	usuario priorizadas, y *user story map* con historias separadas en *releases*.	Product Owner
Arquitectura y diseño	Análisis de requerimientos no funcionales	Requerimientos no funcionales	Escenarios de requerimientos de atributos de calidad	Tech Lead / Arq.
	Diseño de la arquitectura	Escenarios de requerimientos de atributos de calidad	Documento de arquitectura detallando tácticas y patrones arquitectónicos a utilizar.	Tech Lead / Arq.
	Elección de tecnologías	Documento de arquitectura detallando tácticas y patrones arquitectónicos a utilizar, requerimientos no funcionales.	Documento de arquitectura detallando tecnologías a utilizar	Tech Lead / Arq.
Construcción	Planificación del trabajo de desarrollo	Product backlog	Sprint backlog	Equipo Scrum
	Desarrollo de una nueva funcionalidad	Historia de usuario, documento de arquitectura	Código fuente	Equipo de desarrollo
	Revisión de código cruzada (*code review*)	Código fuente, estándares de codificación	Código fuente validado, comentarios con correcciones acerca de código o cumplimiento de estándares	Equipo de desarrollo

	Pruebas unitarias	Código fuente	Pruebas unitarias con al menos %80 de cobertura	Equipo de desarrollo
Prueba	Pruebas cruzadas de funcionalidades	Versión del producto con funcionalidad nueva desarrollada por otro integrante, criterios de aceptación en forma de escenarios BDD.	Reporte de *bugs* encontrados, con capturas de pantalla y pasos para reproducir	Equipo de desarrollo / QA
	Corrección de *bugs*	Reporte de *bug* encontrado, con capturas de pantalla y pasos para reproducir	Versión del producto con *bug* corregido	Equipo de desarrollo
	Verificación de corrección de bug	Versión del producto con *bug* corregido	Versión del producto con corrección de bug verificada por otro integrante	Equipo de desarrollo / QA
	Prueba de regresión	Versión del producto con nuevas funcionalidades para ser liberadas en un *release*	Versión del producto validada, pronta para ser liberada en un *release*	Equipo de desarrollo / QA
	Prueba con usuarios	Versión del producto funcional	Reportes de pruebas con usuarios, encuestas de satisfacción.	Equipo de desarrollo

Tabla 11: plan de calidad

- **Investigación:** la actividad definida en esta sección surge como respuesta al riesgo definido en el capítulo de 7. Gestión de riesgos como R3 - Escaso conocimiento de las tecnologías a utilizar. En dicho capítulo es posible

encontrar información acerca de la actividad en mayor detalle.

- **Ingeniería de requerimientos:** se realizó este proceso que comprende todas las actividades mencionadas con el objetivo de obtener conocimiento en profundidad del dominio del problema, y al mismo tiempo generar documentación que permita guiar al equipo en etapas de desarrollo. El capítulo 4. Ingeniería de requerimientos explica estas actividades en profundidad.

- **Arquitectura y diseño:** una vez definidos los requerimientos no funcionales, se procedió a definir los escenarios de requerimientos de atributos de calidad, de los cuales se desprendieron las tácticas y patrones arquitectónicos a utilizar. Con estas decisiones en mente, se pasó a seleccionar las tecnologías a utilizar y a la instanciación de dichas tácticas y patrones. El capítulo 5. Arquitectura y desarrollo explica estas actividades en mayor detalle.

- **Construcción:** durante el desarrollo, se planifica el trabajo a realizar en cada iteración siguiente, al mismo tiempo que se evalúa el trabajo realizado en la iteración anterior. El capítulo 6. Gestión del proyecto explica cómo fue encarado el proceso de construcción en mayor profundidad.

- **Prueba:** durante esta fase fue ejecutado el plan de pruebas basándose tanto en requerimientos funcionales como en requerimientos no funcionales. De este modo, se aseguró que cada nueva funcionalidad desarrollada cumpla no sólo con los requerimientos, sino también con los escenarios definidos en las historias de usuarios. El objetivo de esto fue poder integrar nuevas funcionalidades con el código existente sin la presencia de efectos secundarios. Es posible encontrar el plan de prueba en mayor detalle en la sección 8.4. Pruebas.

8.3. Aseguramiento de la calidad

Con el objetivo de alcanzar los objetivos establecidos y de esta forma asegurar un nivel apropiado de calidad aplicado al producto y al proceso, se establecieron y

ejecutaron acciones de carácter preventivo y de evaluación a través de la confección de un plan de aseguramiento de la calidad. Este fue definido en etapas iniciales del proyecto, y a medida que el mismo transcurrió, se le fueron efectuando los ajustes que se vieron necesarios. A continuación se explicarán las distintas metodologías y técnicas utilizadas.

8.3.1. Especificación de estándares

En primer lugar, con el objetivo de asegurar la calidad tanto del código desarrollado como de los documentos producidos durante el transcurso del proyecto, se definieron determinados estándares aplicables a ellos. De este modo, se aseguró que tanto el código como los documentos cumplan con determinado formato que facilite su lectura.

Estándares de codificación

Seguir un estándar de codificación es de suma importancia. Se debe elegir un conjunto simple de reglas que definan el formato del código y se deben cumplir consistentemente. Si el trabajo es en equipo, entonces el equipo entero debe acordar en el conjunto de reglas a seguir.

Según el libro *Clean code*, el propósito de seguir un estándar que determine el formato del código se fundamenta de la siguiente manera: "*El formato del código es importante. La funcionalidad también es importante, pero es propensa a cambiar a lo largo de la producción del código. Si se respeta un formato, el código es más legible, lo que facilita los cambios que se harán en el mismo, aumentando su mantenibilidad*." [37].

Se optaron por los siguientes estándares de codificación:

- React (Front-end): Airbnb React/JSX Style Guide. [38]
- Node: JavaScript Standard Style (adaptado a *Typescript*) [39]
- Solidity: Official Solidity style guide. [40]

Estándares de documentación

Tal como lo dicta el documento titulado “Guía para la entrega final de proyectos de grado” de ORTsf, la documentación a entregar deberá cumplir con los siguientes documentos de estándares publicados por la facultad:

- Normas específicas para la presentación de trabajos finales de carrera (Documento N.° 302 - FI) [41]
- Hoja de verificación de formato (Documento N.° 303 - FI) [42]
- Orientación para títulos, resúmenes o abstracts e informes de corrección de trabajos finales de carrera (Documento N.° 306) [43]

8.3.2. Verificación

Durante el transcurso de todo el proyecto, se verificó que cada actividad realizada produzca la salida definida en el plan de calidad. Esta verificación fue necesaria debido a que en varios casos, la salida de una actividad era la entrada de la siguiente.

Además de la verificación de las salidas producidas, también se verificó que dichas salidas respeten los estándares establecidos. En el caso de los documentos, esta verificación se dio de forma manual, mientras que en el caso del código fuente se utilizaron herramientas automatizadas como por ejemplo *plugins* de ESLint.

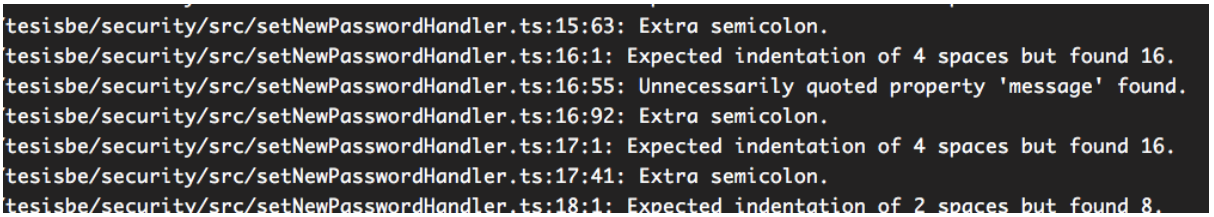


Imagen 43: salida de ESLint

Finalmente, durante la etapa de desarrollo se realizaron revisiones de código cruzadas. Estas tuvieron como principal finalidad tanto la detección de errores o posibles puntos de mejora, como además el cumplimiento con algunos aspectos de

la *definition of done* especificada en la sección 6.2.1.3. Definition of done. Puntualmente, los puntos referentes a la presencia de pruebas unitarias y a la misma realización de la *peer code review*.

8.3.3. Validación

La validación fue variando a lo largo del transcurso del proyecto, tanto en cuanto a técnicas de validación como en cuanto a sus objetivos. En etapas iniciales, la validación fue realizada en dos formas. En principio, se realizó una validación con expertos del dominio y los distintos interesados, la cual se centró en la validación la de existencia de una problemática. Esta validación tuvo un objetivo más genérico, orientado a averiguar si existía un problema a solucionar. Esta validación se detalla en la sección 2.3. Justificación del problema.

Una vez que fue validada la idea general, y fue relevada información más detallada con respecto al problema, se ideó una solución. En ese momento ocurrió la segunda validación, en la cual se utilizaron prototipos desechables con el objetivo de corroborar que la solución ideada se adapte a las necesidades de los distintos interesados. Este proceso se profundiza en la sección 4.2.1. Validación.

Durante etapas iniciales se realizaron además validaciones de la arquitectura del sistema con Ing. Gastón Mousques, integrante de la cátedra de ingeniería de software de universidad ORT. También se realizaron validaciones tanto de la arquitectura como de los requerimientos a desarrollar con integrantes de la empresa cliente, Effectus.

Imagen 44: validación con el cliente

En etapas posteriores, durante el desarrollo del proyecto, con cada *release* que se hacía, se realizaban validaciones con los tipos de usuarios correspondientes para asegurar que las funcionalidades desarrolladas tengan valor para los clientes y se adapten a sus necesidades. En una ocasión, durante estas validaciones surgieron cambios en los requerimientos, los cuales se tradujeron en la necesidad de realizar re-trabajo. Esta ocurrencia se detalla en la sección 7.3.1. R1 - Difícil acceso a la información. Con cada *release* que se hacía también se realizaron además validaciones con el cliente, con el objetivo de mostrar los avance realizados y evaluar su conformidad con los distintos aspectos del proyecto y el producto. Estas se detallan en la sección 8.4.6. Evaluación con el cliente.

Finalmente, se realizaron pruebas con potenciales usuarios con el fin de evaluar el sistema desarrollado, su usabilidad, y su utilidad para los usuarios. Estas pruebas con usuarios serán cubiertas más en detalle en la sección 8.4.4. Pruebas con usuarios.

8.4. Pruebas

En la siguiente sección se detalla el proceso de prueba del software en todos sus distintos niveles, comenzando en el más bajo nivel como lo son las pruebas unitarias, y finalizando en las pruebas de aceptación con usuarios. Se utilizó el modelo V, propuesto por Paul Rook en la década de 1980 [44]. El modelo V

demuestra las relaciones entre cada fase del ciclo de vida del desarrollo y su fase asociada de prueba.

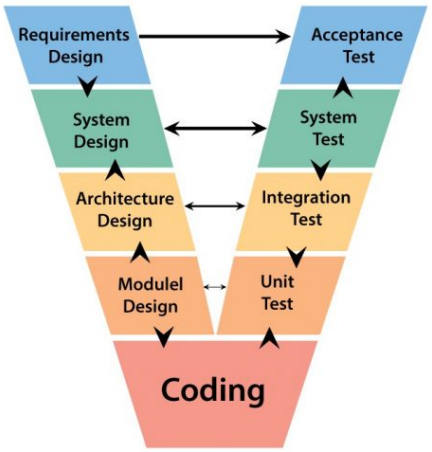


Imagen 45: modelo V

- **Prueba unitaria:** se enfoca en cada componente individualmente, asegurando que funcione correctamente como una unidad. Utiliza en gran medida las técnicas de prueba de caja blanca, ejerciendo rutas específicas en la estructura de control de un módulo para garantizar una cobertura completa y la máxima detección de errores.

- **Pruebas de integración:** aborda el ensamblaje y la integración de componentes para formar un paquete de software completo. Utiliza técnicas de prueba de caja negra para abordar problemas relacionados con la construcción de programas.

- **Pruebas del sistema:** pruebas realizadas en un sistema completo e integrado para evaluar el cumplimiento del sistema con los requisitos especificados. No requiere conocimiento del diseño interno del código o la lógica.

- **Pruebas de aceptación:** las pruebas para verificar que un producto cumpla con los requisitos especificados por el cliente o usuarios. Un cliente o usuario generalmente realiza este tipo de pruebas en un producto [45].

El equipo realizó algunas adaptaciones a este modelo. La primera de ellas se hizo de forma tal de simplificar el proceso de prueba, ya que al tratarse de un equipo de dos integrantes se debió hacer que dicho proceso sea lo más eficiente posible. En lugar de realizar pruebas unitarias y pruebas de integración de forma separada, se codificaron pruebas unitarias que abarquen también las pruebas de integración, utilizando *mocks* para probar la comunicación entre los distintos componentes. Otra adaptación que se hizo fue la de utilizar también pruebas de regresión previo a cada *release*.

8.4.1. Pruebas unitarias e integración

Como fue mencionado en la sección 6.2.1.3. Definition of done, para que una historia se considere como completa, su lógica de *back-end* debe estar cubierta por pruebas unitarias. Dichas pruebas abarcaron desde métodos específicos hasta la ejecución completa de la lógica de un *endpoint*. En primera instancia, estas pruebas fueron implementadas y ejecutadas por el desarrollador responsable de cada historia. Además, cada vez que se subía código nuevo a los repositorios, se ejecutaban las pruebas unitarias asociadas al microservicio modificado.

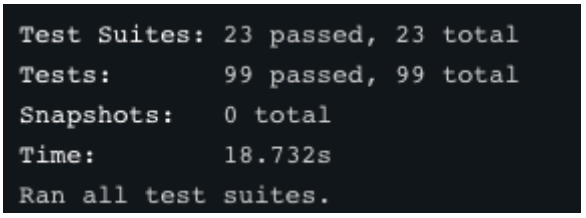


Imagen 46: resultado de la ejecución de pruebas unitarias

```
test("Encryption Test 1 - plain object", () => {
    const object = {
        a: "1",
        b: "2"
    };
    const encripted = encriptObject(object);
    const decripted = decriptBody(encripted);

    expect(decripted).toEqual(object);
})
```

Imagen 47: ejemplo de una prueba unitaria

8.4.2. Pruebas de sistema

Como fue mencionado anteriormente, en lo que a pruebas de sistema concierne, se utilizaron los criterios de aceptación de cada historia. Como fue mencionado en la sección 6.2.1.2. Criterios de aceptación, se utilizó el método BDD para definir los criterios de aceptación en modalidad de escenarios de prueba utilizando el estilo *Given - When - Then* (Dado - Cuando - Entonces). Esta modalidad de definir los escenarios facilitó el proceso de prueba, pues para cada escenario se detalla la precondición, el comportamiento esperado, y el estado final del sistema. Esto fue realmente útil ya que, como fue detallado en el plan de calidad, las pruebas de sistema se realizaban de forma cruzada, por lo que el responsable de las pruebas no estaba tan interiorizado con la historia como el desarrollador, y los escenarios facilitaban su entendimiento.

Es posible ver cada historia con sus criterios de aceptación en el anexo 12.12. Historias de usuario.

8.4.3. Pruebas de regresión

El propósito de las pruebas de regresión es garantizar que las correcciones de errores y las nuevas funcionalidades introducidas en una nueva versión de un software no afecten negativamente la funcionalidad correcta heredada de la versión anterior.

Las pruebas de regresión fueron efectuadas previo a la publicación de cada *release*. En todas las ocasiones se utilizó el enfoque de “regresión parcial”. La prueba de regresión parcial se centra en las partes modificadas de un producto y las áreas adyacentes que podrían haber sido afectadas, seleccionando aquellos casos de prueba relevantes. Se utilizó este enfoque ya que, como fue mencionado previamente, se intentó hacer que el proceso de prueba sea lo más eficiente posible, y este enfoque es más eficiente en términos de tiempo que el enfoque alternativo, la regresión completa.

Como primera instancia de prueba de regresión se ejecutaron la totalidad de las pruebas unitarias automatizadas. Además, también se utilizaron las pruebas de regresión manuales. La prueba de regresión manual es el método básico para la prueba de regresión para cada producto. Este tipo de prueba siempre precede a la automatización, en algunos casos es incluso más eficiente que esta última. Por ejemplo, es imposible escribir scripts de prueba para probar áreas de aplicación adyacentes a la parte del código que se modificó. Se utilizó este enfoque debido a que al realizar únicamente tres *releases*, no representó una inversión de tiempo y esfuerzo lo suficientemente grande como para considerar automatizar el proceso.

A través de pruebas de regresión fueron encontrados un total de 8 defectos, los cuales fueron priorizados y agregados al *backlog*. Estos defectos se distribuyeron de la siguiente manera:

1. Primer release: 3 defectos
2. Segundo release: 3 defectos
3. Tercer release: 2 defectos

4	No se muestra mensaje de error para imágenes de perfil de tamaño mayor a 1 MB	D-01024	Low
5	Una institución puede agregar títulos con el mismo nombre	D-01025	Low
6	Al seleccionar una institución, sigue estando disponible para ser seleccionada de nuevo	D-01026	Low

Imagen 48: defectos encontrados durante la primera prueba de regresión

8.4.4. Pruebas con usuarios

Se realizaron pruebas con usuarios con el fin de asegurar el cumplimiento de los objetivos definidos. De este modo, se corroboró que se desarrolló un producto que no sólo cumpla con los requerimientos funcionales y no funcionales, sino que también brinde soluciones y sea de valor para los usuarios, y que al mismo tiempo sea de fácil uso y aprendizaje.

Imagen 49: pruebas con usuarios

Las pruebas con usuarios fueron prepararon de una forma definida y estructurada, con el objetivo de contar con un proceso definido que produzca resultados medibles y comparables.

En primer lugar, se establecieron criterios para los estudiantes o egresados con quienes se realizarían las pruebas. El principal requisito que debieron cumplir fue estar cursando o haber finalizado una carrera universitaria. En el caso de los empleadores e instituciones académicas, se realizaron pruebas con aquellos que habían sido entrevistados o con quienes ya se habían validado prototipos, de modo que estos puedan ver la evolución del producto y si el mismo cumplió con sus expectativas iniciales. Se hicieron pruebas con Josefina Vidal (supervisora en CPA Ferrere), y Pablo San Nicolás (secretario de gestión académica en ORT), ambos entrevistados durante la etapa de relevamiento de requerimientos.

Luego, se definieron los roles. Es conveniente que en cada prueba con usuarios haya una persona dedicada a guiar la prueba y garantizar que todo funcione sin

problemas y otra u otras dedicadas a observar y tomar notas. Esto se respetó cuando ambos integrantes estaban disponibles, como fue el caso de la prueba con San Nicolás. Sin embargo, al tratarse de un equipo de sólo dos integrantes, en varias ocasiones estas tareas se concentraron en una única persona con el objetivo de poder realizar pruebas aunque no estén disponibles ambos integrantes.

Una vez definidos los roles, se debió establecer un plan que detalle los distintos aspectos de las pruebas. Dicho plan puede ser visto en detalle en el anexo 12.21. Pruebas con usuarios.

A los usuarios que participaron en las pruebas realizadas luego del tercer release, se les envió una encuesta de satisfacción con el objetivo de medir qué tan a gusto quedaron los usuarios con el sistema desarrollado. La misma puede ser vista en el anexo 12.22. Encuestas de satisfacción.

8.4.5. Evaluación heurística

Además de las pruebas con usuarios, se realizó una evaluación utilizando las heurísticas definidas por Jacob Nielsen, con el objetivo de mejorar la usabilidad del sistema. Luego de realizado el análisis y las correcciones pertinentes, se concluyó en que el sistema cumple con las siguientes heurísticas:

- Visibilidad del estado del sistema
- Coincidencia entre el sistema y el mundo real.
- Control del usuario y libertad
- Consistencia y estándares
- Prevención de errores
- Reconocimiento en lugar de recordar
- Flexibilidad y eficiencia de uso

- Ayudar a los usuarios a reconocer, diagnosticar y recuperarse de errores
- Ayuda y documentación

Es posible encontrar el análisis más detallado con capturas de pantalla de la interfaz y algunas correcciones realizadas en el anexo 12.23. Análisis según heurísticas de Nielsen.

8.4.6. Evaluación con el cliente

Tras la liberación de cada *release*, se mantuvieron reuniones con el cliente con la finalidad de realizar una puesta a punto del estado del proyecto, discutir los avances realizados desde el último *release*, y realizar una breve demostración de la versión del sistema. Además de esto, en cada reunión también se discutieron los planes para los próximos pasos, y se le pidió al cliente que rellenara una encuesta para evaluar el grado de satisfacción del cliente con el equipo y el producto desarrollado.

Imagen 50: evaluación con el cliente

En la primera reunión, se explicó el proceso de ingeniería de requerimientos en mayor profundidad, pues estos ya habían sido detallados durante la validación de requerimientos. Se mostró además la versión inicial luego del primer *release* con las funcionalidades asociadas al perfil de las instituciones académicas. En la segunda reunión, además de exponer la versión del producto tras el segundo *release* y discutir los avances realizados, se discutió el cambio en los requerimientos surgido a raíz de la validación tardía. Se mencionó que se incorporarían los cambios en el próximo *release*, algo con lo que el cliente estuvo de acuerdo. Finalmente, en la

tercera reunión, además de realizar la demostración de producto final, se entregó el documento con la transferencia de conocimientos.

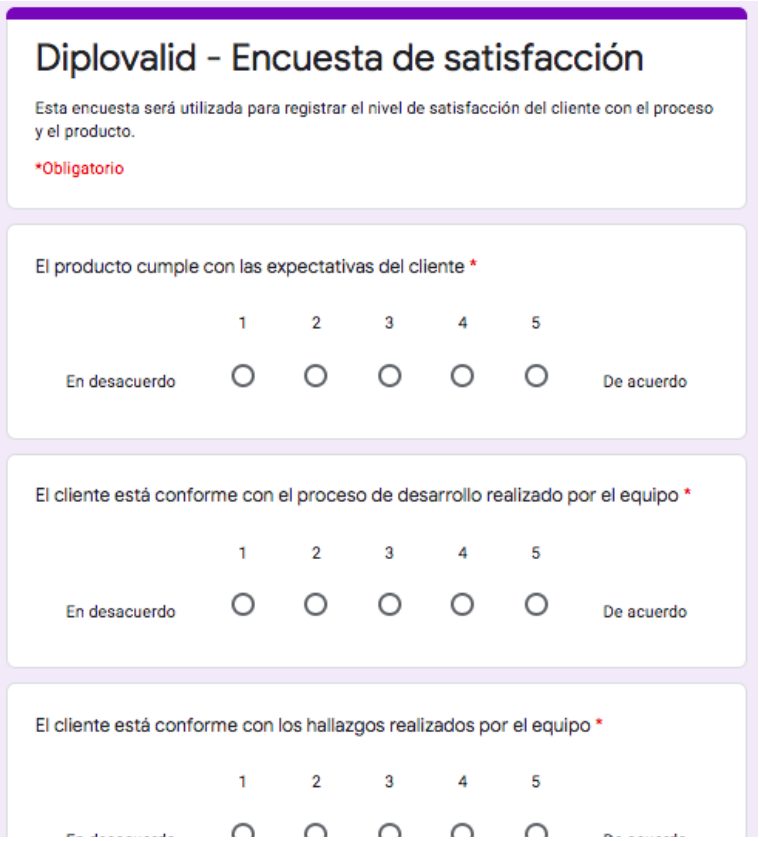


Imagen 51: Encuesta de satisfacción del cliente

8.5. Resultados de ejecución de pruebas

En esta sección se detallarán los resultados de la ejecución de las diversas fases de prueba realizadas a lo largo del proyecto.

8.5.1. Pruebas unitarias

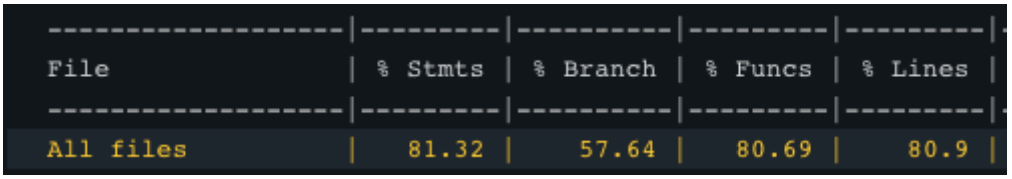
En primer lugar, en lo que a las pruebas unitarias respecta, no se cuenta con resultados registrados en cuanto a la cantidad de defectos que fueron corregidos a raíz de ellas. Esto se debió a que las pruebas unitarias proveen una forma rápida y eficaz de corregir defectos, y cada vez que se detectaba un defecto con la ejecución de una prueba unitaria, el mismo era corregido inmediatamente, sin la necesidad de crear un reporte de defecto. Sin embargo, se puede inferir que gracias a las pruebas

unitarias, el número de defectos reportados a raíz de las pruebas de sistema no fue tan elevado como lo hubiera sido sin ellas.

El resultado que es posible extraer de la ejecución de las pruebas unitarias es la cobertura del código, los cuales marcan que se ha cubierto un porcentaje mayor al 80% de todas las declaraciones y líneas.

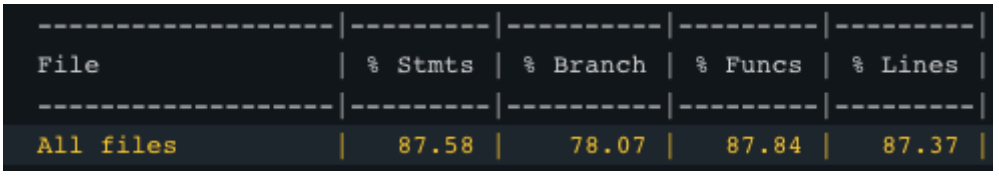
File	% Stmts	% Branch	% Funcs	% Lines
All files	83.53	61.96	79.8	83.02

Imagen 52: cobertura del código del microservicio *qualifications*



File	% Stmts	% Branch	% Funcs	% Lines
All files	81.32	57.64	80.69	80.9

Imagen 53: cobertura del código del microservicio *security*



File	% Stmts	% Branch	% Funcs	% Lines
All files	87.58	78.07	87.84	87.37

Imagen 54: cobertura del código del microservicio *blockchain manager*

8.5.2. Pruebas de sistema y regresión

En cuanto a los resultados de ejecución de pruebas de sistema y regresión, se encontraron un total de 30 defectos, 8 de los cuales surgieron a raíz de pruebas de regresión. Cada defecto encontrado fue categorizado y priorizado. Además de esto, cada reporte de defecto incluyó capturas de pantalla y pasos para reproducir. En cuanto a las categorías utilizadas, las mismas fueron las siguientes:

- **Usabilidad:** relacionados con aspectos que puedan ser perjudiciales para la usabilidad del sistema. Ejemplos:

    - En el perfil de las instituciones, el botón para eliminar títulos y el botón para eliminar requisitos tenían formatos diferentes, mientras deberían tener el mismo formato al tratarse de una acción similar para mantener la consistencia.
    - Al intentar iniciar sesión con una cuenta que aún no ha sido verificada, el mensaje de error no es descriptivo.

- **Performance:** aunque no fue un aspecto central del sistema, existieron defectos relacionados con la rapidez con la cual el usuario recibía *feedback* por sus acciones. Se creó esta categoría porque había ciertos comportamientos del sistema que no eran funcionalmente incorrectos, pero requerían un tiempo de ejecución demasiado largo. Podrían ser considerados de usabilidad, ya que también dañan la usabilidad del sistema, por lo que se podrían ver como un subconjunto más específico de los defectos de usabilidad. Ejemplos:
    - La acción de registro de usuario solía tardar hasta 30 segundos antes de mostrar en pantalla un mensaje de confirmación.
    - La acción de subir archivos asociados a calificaciones solía tardar hasta 30 segundos antes de finalizarse, si no fallaba la operación por *timeout* dentro del *back-end*.
- **Seguridad:** relacionados con la seguridad de la aplicación, principalmente con la autorización de los usuarios. Ejemplo:
    - Las instituciones académicas podían acceder a perfiles de egresados que no habían ingresado calificaciones para su institución.
- **Funcionalidad:** relacionados con las funcionalidades desarrolladas. Generalmente surgían a raíz de criterios de aceptación que no eran satisfechos. Ejemplos:
    - No se mostraba mensaje de error para imágenes de perfil de tamaño mayor a 1 MB.

    - Cuando un egresado seleccionaba una institución, seguía estando disponible para ser seleccionada de nuevo por el egresado.

- **Interfaz:** estrictamente relacionados con aspectos de interfaz gráfica, ya sea tamaño de botones, textos, etc. Ejemplos:

    - En la pantalla de inicio de sesión, un botón tenía el texto “Log In” en lugar de “Ingresar”.

En cuanto a la prioridad, se utilizaron los siguientes criterios para asignarle una prioridad a cada defecto:

- **Alta**: el defecto no permite la ejecución correcta de la funcionalidad desarrollada, daña la usabilidad del sistema de forma considerable, o provoca un incumplimiento de algún requerimiento no funcional de seguridad.
- **Media**: el defecto permite la ejecución correcta de la funcionalidad desarrollada aunque daña la usabilidad del sistema de una forma que no es considerada severa aunque tampoco leve.
- **Baja**: el defecto permite la ejecución correcta de la funcionalidad desarrollada aunque daña levemente la usabilidad del sistema.

Se registraron las siguientes métricas en cuanto a la distribución de cantidad de defecto reportados y resueltos por sprint:

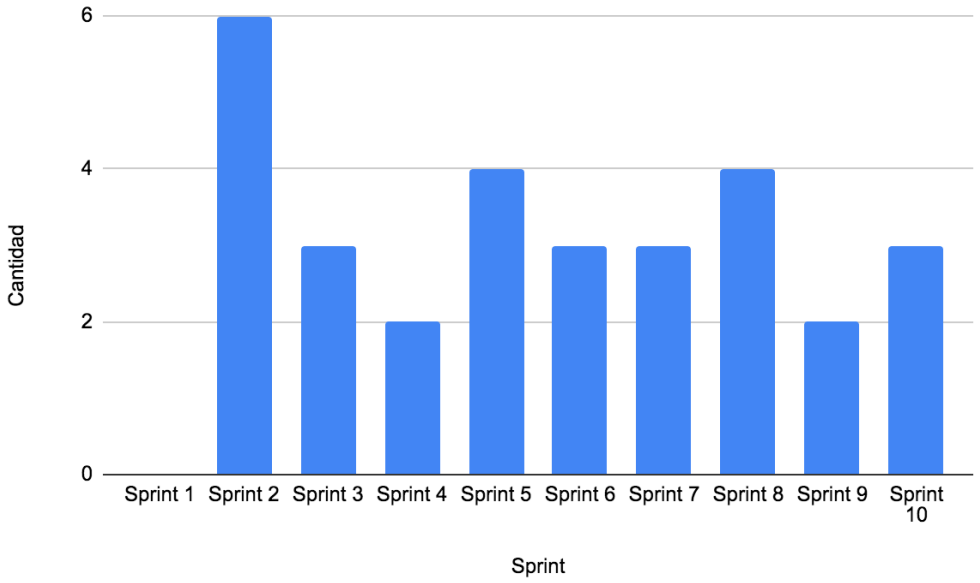


Imagen 55: cantidad de defectos hallados por sprint

A los defectos reportados se les fue asignada la siguiente prioridad:

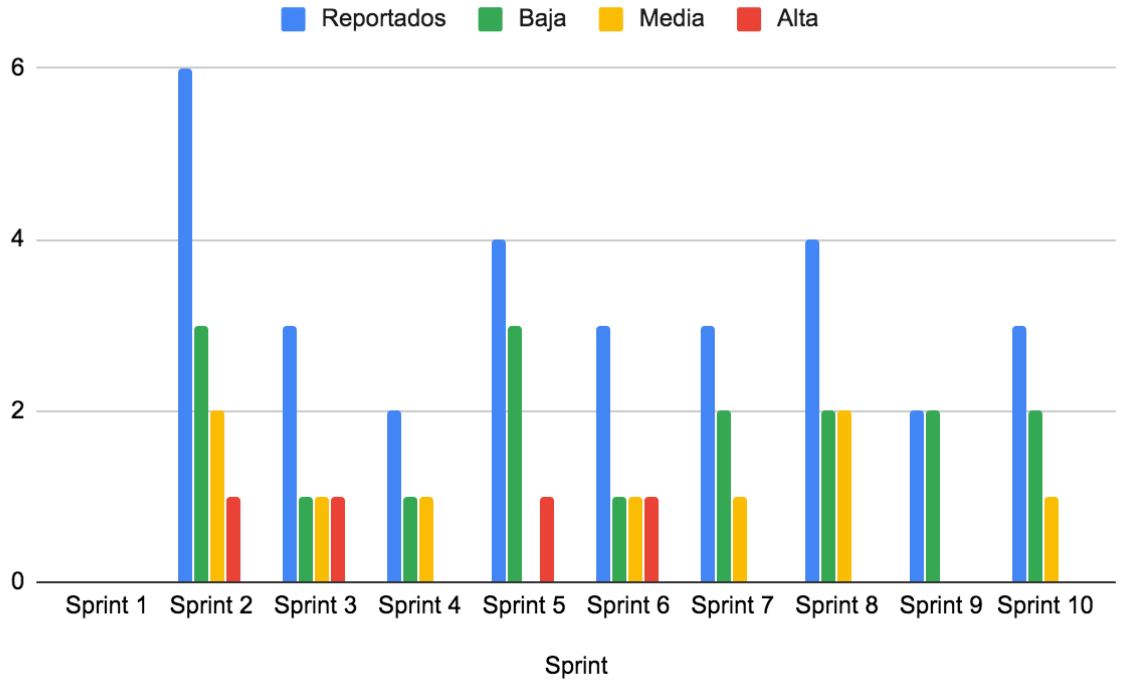


Imagen 56: cantidad de defectos hallados por sprint y por prioridad

Finalmente, la distribución por categorías fue la siguiente:

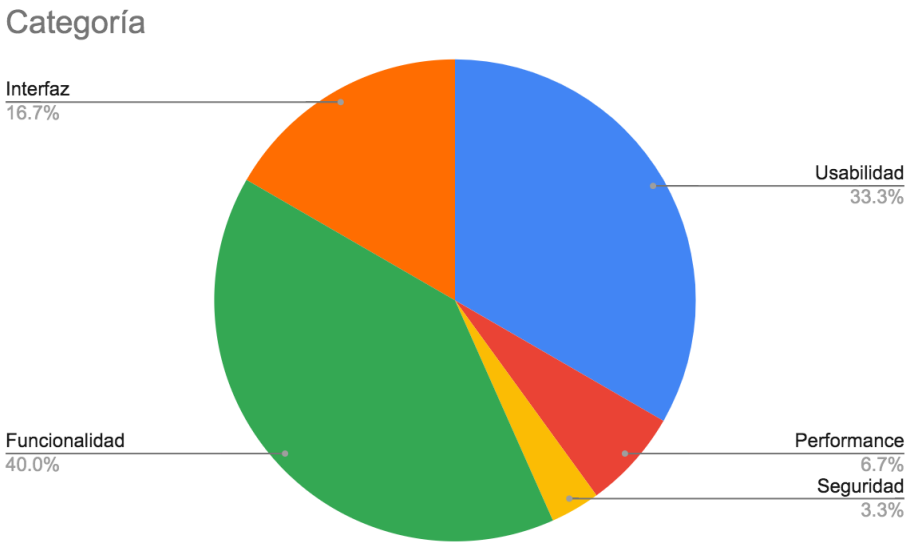


Imagen 57: porcentaje de defectos hallados por categoría

8.5.3. Pruebas con usuarios

A continuación se detallarán los resultados de las pruebas con los distintos tipos de usuarios.

En primer lugar, se resumirán los resultados de las pruebas realizadas con un integrante de la institución educativa. Como fue visto anteriormente, la primera prueba realizada con San Nicolás derivó en nuevos requerimientos los cuales llevaron a la necesidad de realizar re-trabajo. Aunque este no fue el resultado esperado, fue de gran utilidad para desarrollar un producto que se adapte a las necesidades reales de las instituciones académicas, y de esta forma lograr el cumplimiento de los objetivos propuestos. Una vez incorporados al proyecto y desarrollados los nuevos requerimientos, se validó el sistema nuevamente con San Nicolás. En esta ocasión, se obtuvo la aprobación del mismo sin la necesidad de realizar re-trabajo.

En segundo lugar, la validación realizada con Josefina Vidal dio los resultados esperados, mostrándose satisfecha con el producto desarrollado. Este resultado fue esperable ya que en etapas de relevamiento de requerimientos, ella fue una de las

personas con quienes se validaron los prototipos, y se considera que durante la etapa de desarrollo se respetó lo conversado con los distintos interesados.

Finalmente, las pruebas con alumnos y egresados brindaron la posibilidad de realizar estadísticas debido al uso del *template* establecido y la cantidad de participantes con los cuales se pudo realizar. Los resultados fueron los siguientes:

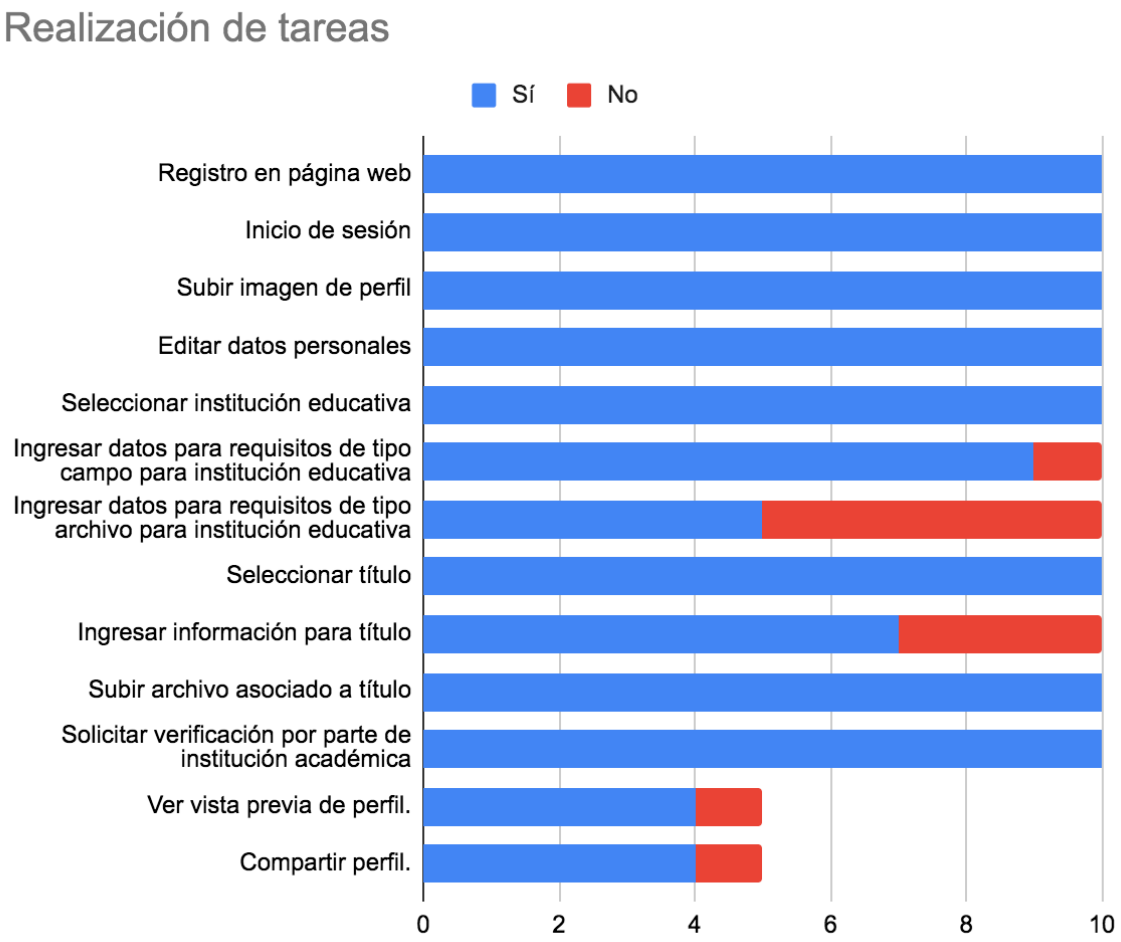


Imagen 58: resultado de las pruebas con usuarios

Cabe aclarar que las últimas dos tareas fueron probadas con menos usuarios debido a que corresponden a funcionalidades desarrolladas en el último *release*, por lo que estuvieron presentes durante una única instancia de prueba, mientras que el resto estuvo presente en ambas instancias de prueba.

En lo que a resultados concierne, las tareas que menos se pudieron completar fueron las de ingreso de datos para requisito de tipo archivo y la de ingreso de

información adicional para títulos. En la primera instancia de prueba, solamente 1 de 5 pudo completar la primera, y 2 de 5 la segunda. Por esta razón, se decidió incluir textos de ayuda para indicarle a los egresados cuál es la función de los elementos que pueden ver en pantalla. Los cambios introducidos fueron los siguientes:

Imagen 59: inclusión de textos de ayuda en la aplicación

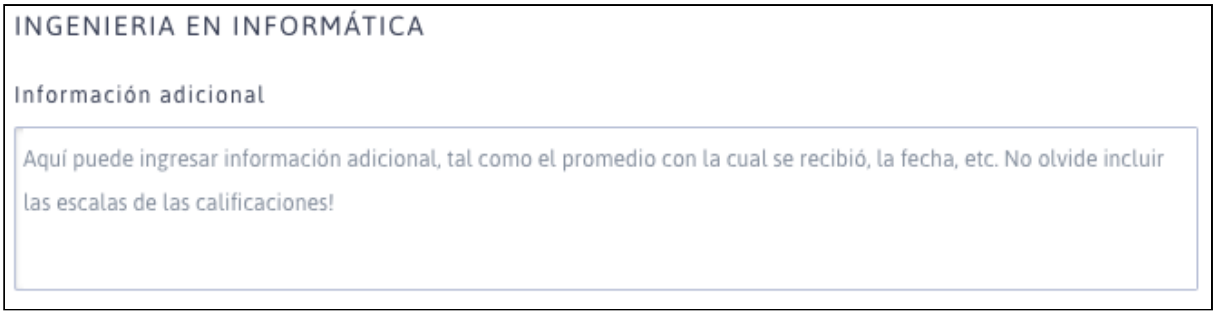


Imagen 60: *placeholder* en el campo de texto para ayudar a los usuarios

Durante la segunda instancia, los resultados para ambas tareas mejoraron. En este caso, 4 de 5 pudieron completar la primera tarea, y también 4 de 5 pudieron completar la segunda.

Como fue mencionado anteriormente, luego de las pruebas realizadas al finalizar el tercer *release* se les envió a los participantes una encuesta de satisfacción. Los resultados fueron los siguientes:

Afirmación	Promedio
La aplicación es fácil de usar	4,1

La aplicación es de gran utilidad	4,3
La aplicación es segura y confiable	4,2
La aplicación es agradable de usar	3,7
Puntaje general de la aplicación	4,1

Tabla 12: resultado de encuesta de satisfacción a usuarios

En general, los resultados de las encuestas de satisfacción fueron considerados más que aceptables por el equipo. Como fue mencionado en la sección 1.4.3. Objetivos del producto, uno de los objetivos en cuanto al producto era lograr la satisfacción del cliente y de los usuarios desarrollando un producto útil y de calidad. Los puntajes obtenidos en relación a la utilidad de la aplicación desarrollada, y el puntaje general de la misma fueron satisfactorios para el equipo. Esto permitió al equipo considerar como cumplido dicho objetivo en lo que a los usuarios respecta.

8.5.4. Evaluación con el cliente

Las encuestas de satisfacción con el cliente dieron buenos resultados a lo largo del proyecto. En las tres reuniones (una por cada *release*), el cliente pareció estar a gusto con los distintos aspectos evaluados en las encuestas.

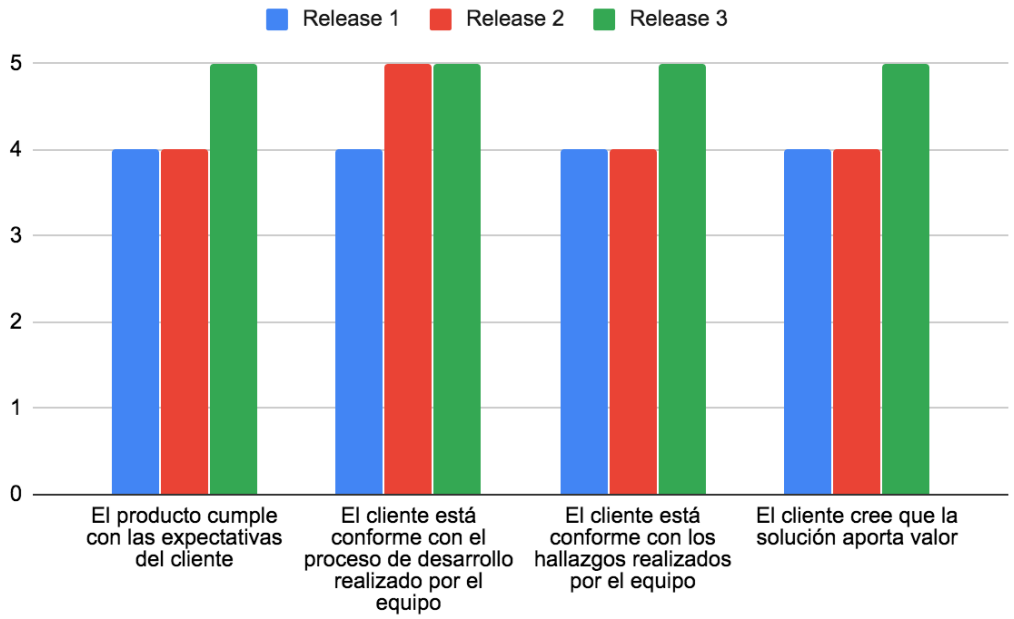


Imagen 61: resultado de encuestas de satisfacción al cliente por release

8.6. Acciones correctivas

El equipo utilizó la herramienta VersionOne para el reporte, seguimiento, y resolución de los distintos defectos que surgieron durante la etapa de desarrollo. Como fue mencionado anteriormente, para facilitar la reproducción y resolución de cada defecto, cada uno de ellos incluyó una captura de pantalla representativa del defecto, y los pasos necesarios para reproducirlo. Estos fueron incluidos en el reporte de defectos dentro de VersionOne.

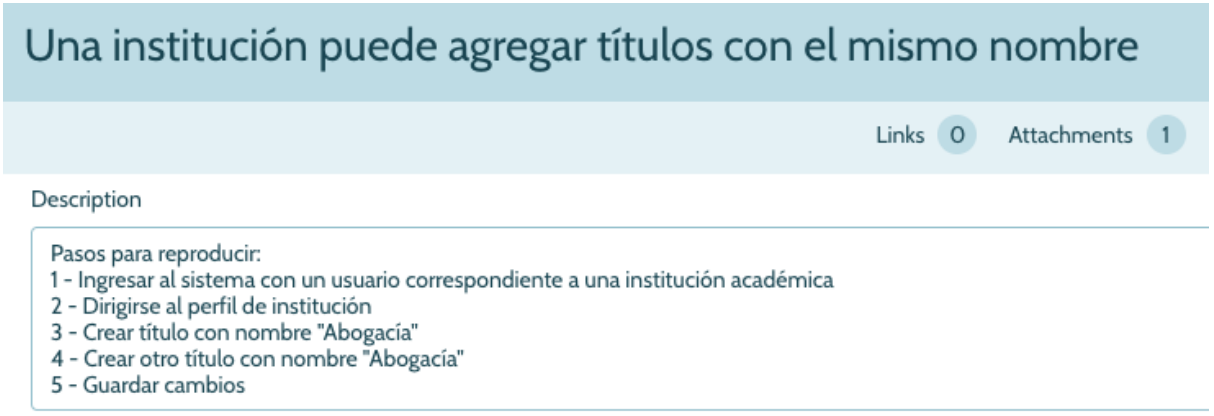


Imagen 62: ejemplo de defecto registrado en VersionOne

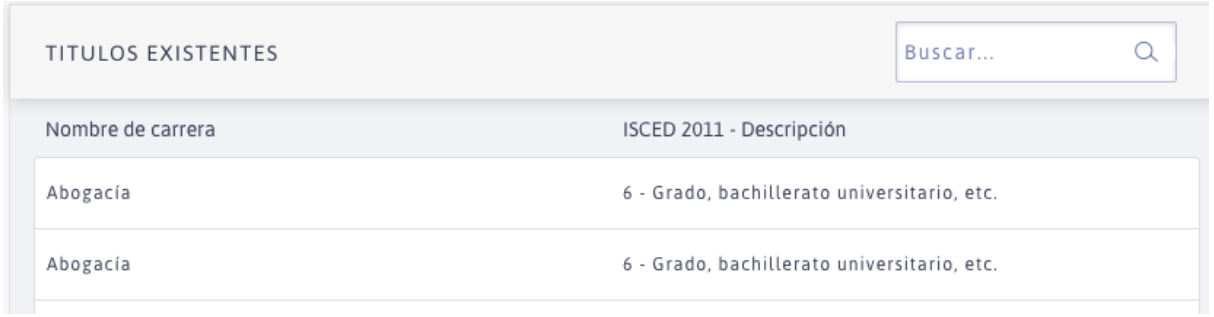


Imagen 63: captura de pantalla adjunta al defecto

En cuanto a la resolución de los defectos encontrados, se dio de la siguiente forma. Cada vez que se reportaban defectos asociados a una historia, dicha historia no podía considerarse como finalizada hasta que fueran resueltos todos los defectos asociados a ella. Esta fue una de las razones por las cuales muchas historias fueron finalizadas en *sprints* siguientes al planificado inicialmente. Cuando un *sprint* finalizaba y aún quedaban defectos pendientes asociados a una historia de usuario, la misma era pasada al siguiente *sprint* junto con sus defectos. Luego, durante las

reuniones de *sprint planning*, el equipo debía tener en cuenta estas historias sin terminar, generalmente dándoles una prioridad mayor que aquellas historias aún no comenzadas.

La resolución de defectos a lo largo de los sprints se dio de la siguiente forma:

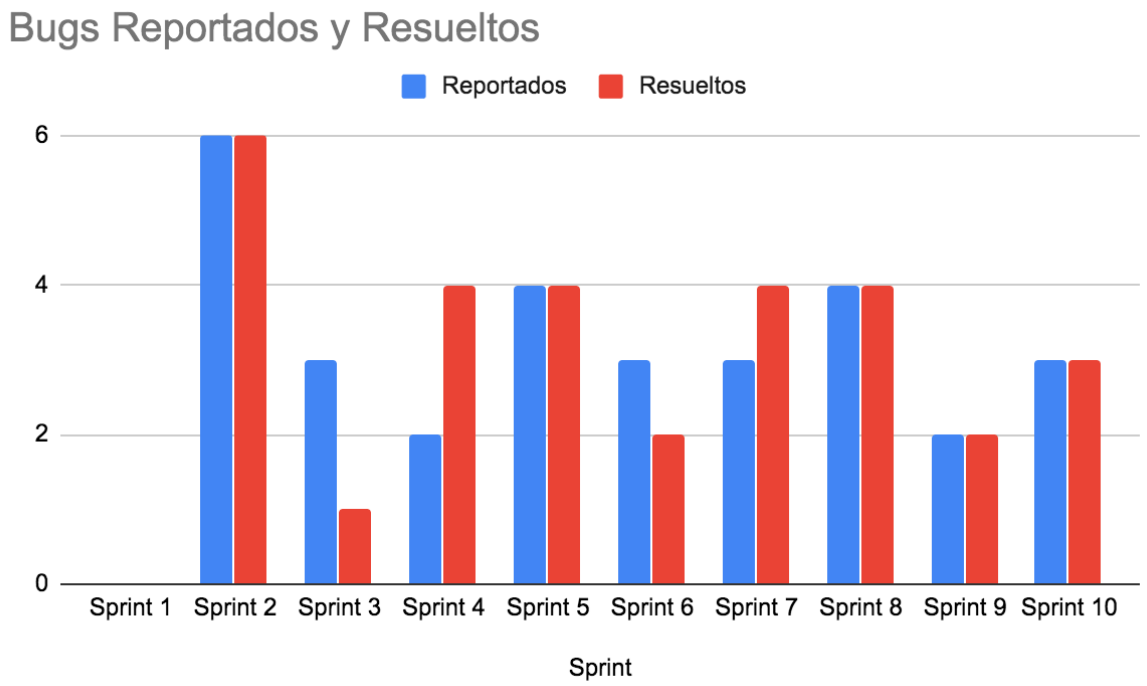


Imagen 64: cantidad de defectos encontrados y resueltos por sprint

De esta gráfica es posible extraer algunas conclusiones relevantes. En primer lugar, teniendo en cuenta las fechas en las cuales se realizaron los *releases* (final de *sprint 5*, final de *sprint* 7, y final del *sprint 10*), se puede concluir que no se liberó ninguna versión del producto que contuviera defectos reportados sin resolver. Esto se debió a que en aquellos *sprints* previos a realizar *releases*, se le asignó una prioridad mayor a la resolución de defectos.

Adicionalmente, es posible observar que en los *sprints* finales todos los defectos reportados fueron resueltos en el mismo *sprint* en el cual fueron reportados. Esto se debió que durante la *sprint retrospective* al finalizar el séptimo *sprint*, uno de los asuntos que surgió que tenían al equipo descontento fue que algunos defectos estaban impidiendo que las historias se finalizaran a tiempo, afectando la velocidad del equipo. A raíz de esta situación se designó una acción a llevar a cabo para

corregir esta ocurrencia, a través de la cual se determinó que se le daría mayor prioridad a la resolución de defectos dentro del *sprint* en el cual fueron reportados. Observando la gráfica se puede concluir que esta acción fue ejecutada de forma exitosa.

8.7. Conclusiones

Con el proyecto finalizado, y luego de haber implementado mecanismos de aseguramiento de la calidad a lo largo de todo el proyecto, mirando los resultados obtenidos es posible concluir que el aseguramiento de calidad, como el nombre lo sugiere, juega un rol vital en el desarrollo de un producto de calidad.

Comenzando con las pruebas unitarias, estas tienen beneficios que parecen imperceptibles debido a que no producen reportes de defectos. En lugar de ello, las pruebas unitarias funcionan como mecanismo de prevención de defectos, evitando que estos deban ser reportados. Esto tiene un gran valor para el proyecto, sobre todo teniendo en cuenta que uno de los temas vistos en una retrospectiva fue la lenta resolución de defectos por parte del equipo. En síntesis, las pruebas unitarias permitieron un gran ahorro de tiempo y esfuerzo en corrección de defectos.

Siguiendo con las pruebas de sistema y regresión, el beneficio de ellas sí es medible en cantidad de defectos reportados. En caso de no haber realizado pruebas de sistema o de regresión, una gran cantidad de defectos - 30 para ser exactos - se hubieran filtrado a los *releases*, disminuyendo la calidad del producto percibida por los usuarios durante las pruebas, y posiblemente perjudicando las encuestas de satisfacción.

Finalmente, las pruebas con usuarios fueron vitales para asegurar que no existan tareas demasiado complejas para el usuario promedio. Generalmente, es difícil para aquellos que desarrollaron un sistema discernir cuando una tarea se ha tornado demasiado trabajosa para los usuarios. A través de estas pruebas se detectó cuáles tareas lo eran, y se evitó que en la versión final del producto existan tareas que no son realizables por más de la mitad de los usuarios.

9. Gestión de la configuración

En este capítulo se detalla la estrategia que el equipo definió para llevar adelante la gestión de la configuración del software. Se especifican los elementos de configuración de software (ECS) identificados, las herramientas utilizadas para gestionarlos, la estrategia para el manejo de versiones, y la forma en la que se controlan los cambios en el proyecto.

Tener control de versiones y de cambios es de suma importancia para asegurar la calidad del producto. Permite asegurar que todos los miembros del equipo se encuentran trabajando la misma versión del código fuente o de un documento.

9.1. Identificación de los elementos de la configuración

Se consideró un elemento de configuración de software (ECS) a todo elementos generado durante el desarrollo del proyecto, relacionados al código o a la documentación, que fuera necesario gestionar e identificar en sus distintas versiones.

9.1.1. Elementos de código fuente

Elemento	Lenguaje	Descripción
Front end	React JS	Capa de presentación de la aplicación. Contiene la interfaz gráfica con la que interactúa el usuario.
Blockchain manager	Node JS	Servicio encargado de encapsular la interacción con el contrato inteligente alojado en la blockchain.
Notifications	Node JS	Servicio en el cual se centraliza el envío de notificaciones por email.

Security	Node JS	Servicio encargado de la creación y gestión usuarios dentro del sistema, así como también de las de sesiones de los usuarios y sus permisos asociados.
Qualifications	Node JS	Servicio encargado del manejo de la información de instituciones académicas y egresados, solicitudes de verificación por parte de egresados, y el estado de cada calificación.
Contrato inteligente	Solidity	Contrato inteligente encargado de registrar información de usuarios y diplomas en la blockchain.

Tabla 13: elementos de configuración de código fuente

9.1.2. Elementos de documentación

Elemento	Descripción
Actas de reunión con el tutor	Actas con un resumen de lo conversado en cada reunión con el tutor, y acciones a seguir.
Backlog	Registro de historias de usuario, realizadas y pendientes, priorizadas y con distinto nivel de detalle.
Bibliografía consultada	Links a sitios webs consultados, y referencias a libros de donde se obtuvo información para la realización del proyecto.
Bocetos del sistema	Bocetos creados en papel para presentar a los posibles usuarios del sistema y validar las distintas funcionalidades.
Documento de arquitectura	Documento donde se detalla la arquitectura diseñada para el sistema junto a su debida

	justificación. Incluye también la elección de los atributos de calidad priorizados y tácticas utilizadas.
Encuestas de satisfacción	Encuestas realizadas al cliente y miembros del equipo con el objetivo de medir el nivel de satisfacción tanto con el producto obtenido como con el proceso realizado.
Entrega final	Documento entregado como resultado del proyecto realizado, para su evaluación por parte de la universidad.
Estándares de codificación	Conjunto de reglas y buenas prácticas a seguir al momento de escribir código en los distintos lenguajes de programación. Aseguran uniformidad en el código escrito y fácil comprensión por parte de cualquier desarrollador.
Feedback de revisiones	Registro del feedback obtenido por los distintos docentes en cada una de las revisiones realizadas durante el transcurso del proyecto.
Gestión de incidentes	Registro de los errores encontrados durante las etapas de testing, clasificación y priorización de los mismos.
Gestión de la calidad	Objetivos de calidad, plan de calidad y de aseguramiento de la calidad y métricas de calidad.
Gestión de la configuración	Registro de la estrategia definida para mantener
Gestión de las comunicaciones	Documento que describe los mecanismos y herramientas utilizados para la comunicación entre los interesados del proyecto.
Gestión de riesgos	Documento que describe los riesgos existentes del proyecto, su magnitud, como prevenirlos y mitigarlos.

Informes de avance	Informes de avance relacionados al proyecto, solicitados por la Universidad ORT.
Ingeniería de requerimientos	Documento donde se registraron las técnicas utilizadas para relevar requerimientos, y los resultados obtenidos en cada etapa del proceso.
Resultado de encuestas	Resultado de encuestas realizadas a distintos actores involucrados en el proyecto con el objetivo de obtener respuestas de un gran número de personas al mismo tiempo.
Registro de entrevistas	Apuntes tomados durante entrevistas realizadas a los distintos involucrados en el proyecto. Incluye también transcripción de entrevistas grabadas en audio.
Resultados de investigación	Informes escritos al finalizar una investigación sobre una nueva tecnología o herramienta, para evaluar su utilidad.
Resultado de reuniones retrospectivas	Registro de lo charlado en las reuniones retrospectivas del equipo, planes de mejora y acciones a tomar.
Reglamento del equipo	Documento creado al inicio del proyecto para fijar reglas de trabajo para el equipo, con el fin de unificar y establecer el compromiso necesario para llevar adelante el proyecto.

Tabla 14: elementos de configuración de documentación

9.2. Elección de herramientas

En esta sección se presentan las herramientas utilizadas a lo largo del proyecto para la gestión, seguimiento, versionado y almacenamiento de los elementos de configuración. Se especifican también los procesos de la gestión de la configuración seguidos.

9.2.1. Software

Para el versionado del código fuente de los distintos componentes de la aplicación se optó por un modelo distribuido. Este tipo de modelo promueve el uso de ramas (*branches*) para el desarrollo de las distintas funcionalidades de un sistema, sin afectar a otros desarrollos que se estén haciendo en simultáneo sobre el mismo sistema. Una vez concluido el desarrollo de una funcionalidad, estos cambios pueden combinarse con otras ramas para mantener todas las funcionalidades incluidas en una misma base de código.

La herramienta seleccionada para el control de versiones del código fuente fue *Git*. Se trata de un sistema de control de versiones gratuito y de código libre, pensando en la eficiencia y la confiabilidad del mantenimiento de versiones de aplicaciones cuando éstas tienen un gran número de archivos de código fuente [46].

La elección del cliente *Git* usado para la comunicación con el repositorio remoto quedó a criterio personal de cada miembro del equipo, ya que no es algo que afecte el resultado obtenido. En algunos casos se usó *GitKraken*, en otros se trabajó directamente desde una consola o con *git bash*.

9.2.2. Documentación

Para la gestión de todos los documentos relacionados al proyecto se utilizó *Google drive* como repositorio de datos. La elección de esta herramienta se justifica por ser gratuita, ofrecer gran capacidad de espacio de almacenamiento, disponer de una interfaz de usuario intuitiva y fácil de usar, y principalmente porque permite editar documentos en forma concurrente, agregar comentarios y sugerir cambios. Ofrece además la posibilidad de revisar el histórico de los cambios realizados, lo que permite ir hacia atrás en caso de realizar algún cambio no intencionado.

9.3. Organización de repositorio

Se describe aquí la estrategia utilizada para la organización de los distintos repositorios, tanto de código como de documentación.

9.3.1. Software

Como proveedor de repositorios se utilizó Bitbucket, esto fue una decisión del cliente quien nos proveyó de 2 repositorios privados para el frontend y el backend respectivamente.

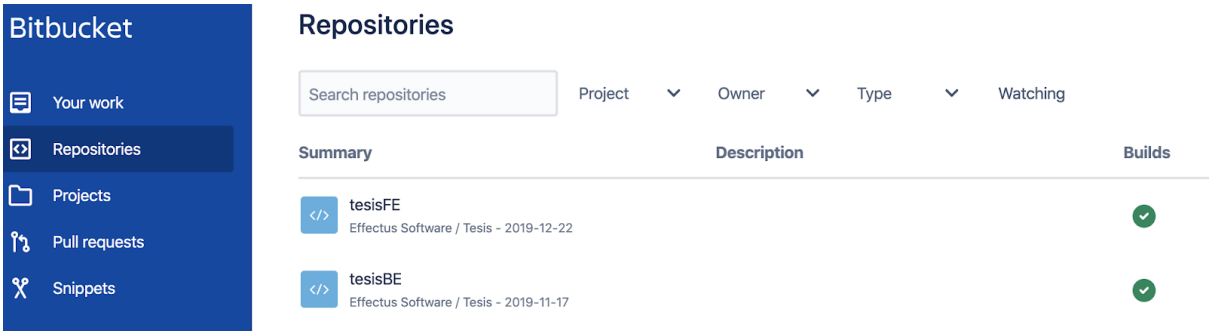


Imagen 65: repositorios de BitBucket

El repositorio "tesisBE" contiene una carpeta por cada servicio que forma parte del *back-end*.

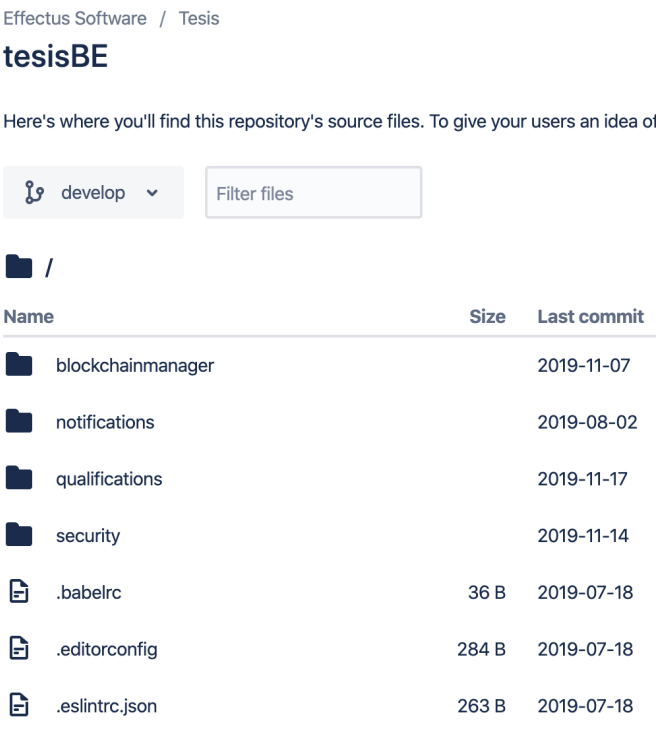


Imagen 66: contenido del repositorio "tesisBE"

9.3.1.1. Política de branching

Para organizar el trabajo se siguió el flujo de trabajo *GitFlow* [47] en ambos repositorios. Se trabajó con las siguientes ramas:

- **beta**: rama principal, contiene la última versión estable y funcional del sistema. Cada vez que se realizó un *release*, se actualizó el código de esta rama, lo cual desencadenó un despliegue automático.
- **develop**: rama donde se van integrando las últimas funcionalidades desarrolladas.
- **features**: cada vez que se comienza a trabajar en una nueva funcionalidad, se crea una nueva rama *feature* a partir de *develop*. Una vez concluido el desarrollo se crea una solicitud para integrar estos cambios en *develop*. Dichas solicitudes tomaron la forma de *pull requests*, las cuales debieron ser revisadas por el otro integrante como parte de la revisión cruzada.

9.3.2. Documentación

Se organizaron los documentos en una estructura de carpetas, como muestra la siguiente imagen:

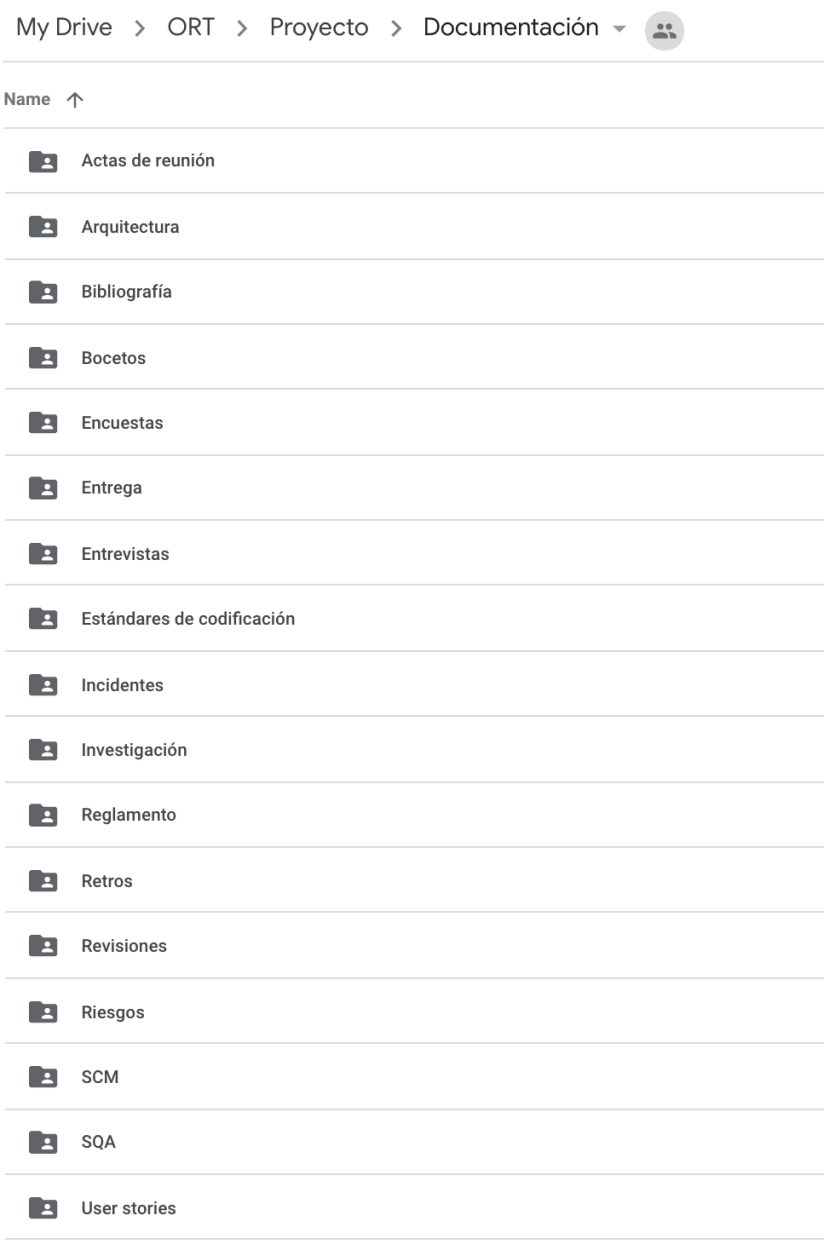


Imagen 67: repositorio de documentos

9.4. Despliegue

Como fue detallado anteriormente, como repositorio de código y herramienta de versionado se utilizó *Bitbucket*, ya que fue la herramienta provista por *Effectus*. Se cuenta con dos repositorios, uno que almacena el código del *back-end*, y otro que almacena el código del *front-end*. Tanto para el *back-end* como para el *front-end* se utilizó *Codeship* como herramienta de despliegue e integración continua.

En el caso del *front-end*, cada vez que se hace sube nuevo código al repositorio de *bitbucket*, este interactúa con el proyecto en codeship a través de *Web Hooks*, desencadenando un build. Cuando esto ocurre, *Codeship* ejecuta una serie de comandos para hacer el *build* del proyecto, y luego sube el resultado de dicho *build* a un *S3 Bucket*, en donde se hostea la web app. El ambiente al cual se realiza el deploy depende del nombre de la *branch* a la cual se acaba de realizar el *push*.

El ambiente de desarrollo para el *front end* es el servidor local *localhost*.

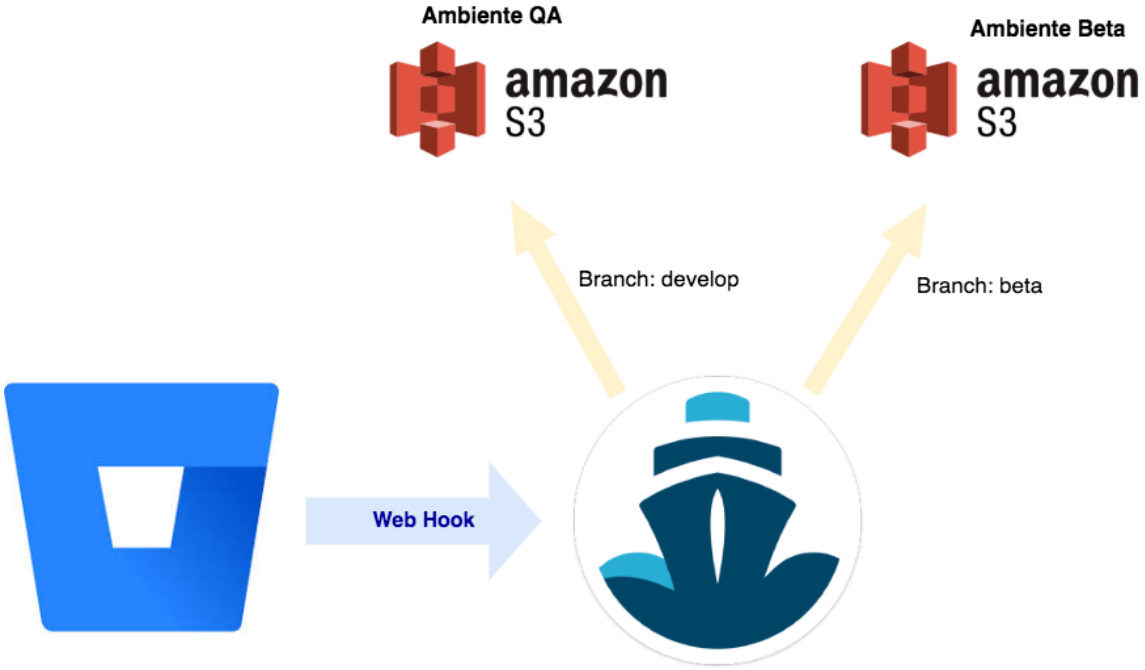


Imagen 68: diagrama de despliegue del *front-end*

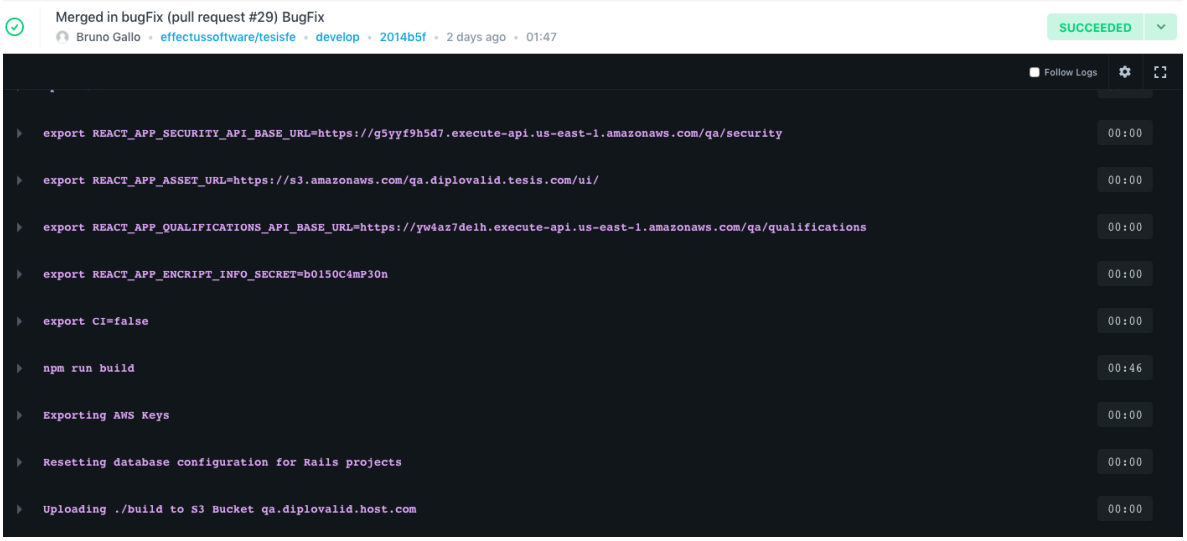


Imagen 69: salida de consola de *CodeShip* durante el despliegue

En el caso del *back-end*, el funcionamiento es similar al anterior, con la excepción de que los *builds* de codeship utilizan la herramienta *serverless* para desplegar el nuevo código de los lambdas.

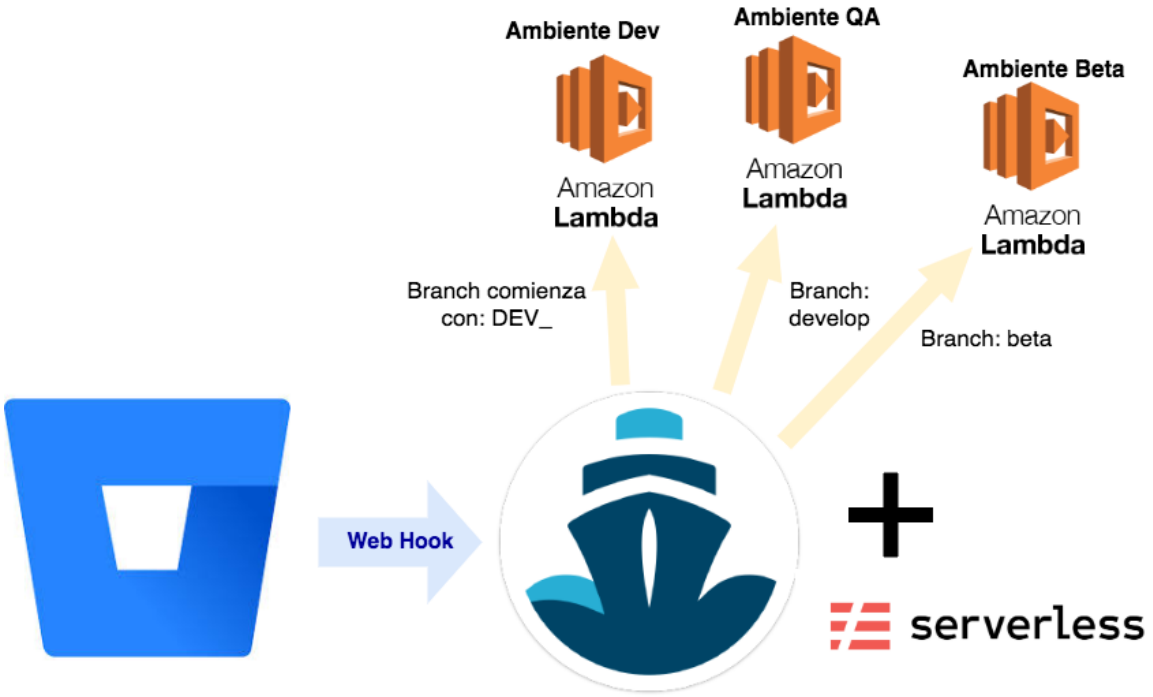


Imagen 70: diagrama de despliegue del *back-end*

9.5. Manejo de dependencias

Una de las tareas de la gestión de la configuración es la de administrar las dependencias de los distintos proyectos. De este modo, se pudo controlar cuáles eran las versiones de las distintas librerías utilizadas por cada componente. Esta tarea fue de gran importancia para facilitar el armado del ambiente de desarrollo de cada integrante del equipo, y asegurar que ambos estén utilizando las mismas versiones de las dependencias.

Tanto en el *front-end* como en el *back-end*, el manejo de dependencias fue realizado con NPM. El mismo utiliza un archivo llamado *package.json*, en el cual se indican las distintas dependencias y su respectiva versión, tanto de desarrollo (aquellas necesarias para realizar tareas de desarrollo, como por ejemplo ejecutar pruebas),

como aquellas de producción, es decir, necesarias para el correcto funcionamiento del sistema en tiempo de ejecución. En ambos casos, las dependencias fueron excluidas de los repositorios, por lo que la herramienta NPM fue particularmente útil ya que permite instalar todas las dependencias en un nuevo equipo con tan sólo un comando.

Como fue mencionado en el capítulo 5. Arquitectura y desarrollo, se implementó el patrón de microservicios para el *back-end*. La ventaja de este patrón en cuanto al manejo de dependencias es que cada microservicio cuenta con su propio archivo *package.json*, el cual determina únicamente las dependencias del microservicio al cual pertenece. De este modo, los distintos microservicios presentan sólo las dependencias necesarias para su ejecución como unidad independiente, lo cual aceleró el proceso de despliegue de cada microservicio de forma independiente.

9.5. Manejo de respaldos

En cuanto al manejo de respaldos, este no se hizo de forma manual, sino que fue hecho por las distintas herramientas seleccionadas. En el caso de los documentos, los mismos fueron almacenados en *Google Drive*. Para respaldar estos archivos, se utilizó la aplicación de escritorio de *Google* para mantener una copia local de los mismos que se mantenga actualizada automáticamente.

En el caso de los repositorios de código, los mismos fueron almacenados en repositorios privados de *Bitbucket*. Con el fin de poder desarrollar el código, ambos integrante del equipo contaron en todo momento con una copia de los repositorios en sus ambientes locales. Además de esto, se utilizó la herramienta *Bitbucket server backup client*. Esta herramienta contiene dos programas de línea de comandos que se pueden usar para hacer una copia de seguridad y restaurar un servidor *Bitbucket*.

9.7. Conclusiones

En conclusión, la gestión de la configuración tuvo como resultado la facilitación de varios aspectos en el proyecto. Esto se debió en gran parte a la amplia disponibilidad de herramientas con las que se cuenta hoy en día. Mediante el uso de

Google Drive, el equipo contó en todo momento con las versiones más actualizadas de todos los documentos relacionados al proyecto. Además, con el uso de *BitBucket*, el equipo también pudo contar con todas las versiones del código con la ejecución de simples comandos en la consola. Finalmente, combinando el uso de herramientas como *Codeship*, *serverless*, y varios servicios provistos por *Amazon*, se logró generar *pipelines* que realicen despliegues de forma automática y continua cada vez que se actualiza el código de los distintos repositorios. Gracias a ellos, no transcurría más de diez minutos entre que se actualizaba el código de los distintos repositorios hasta que se contaba con el código desplegado en la página web.

Finalmente, el correcto manejo de las dependencias fue esencial para completar rápidamente las historias de usuario que fueron más técnicas, como por ejemplo el despliegue de la aplicación y las instalaciones de los ambientes de desarrollo.

Muchas veces todas estas herramientas son pasadas por alto, aunque cuando se ejecuta un proyecto en su totalidad de inicio a fin es posible apreciar su importancia y la forma en que facilitan las tareas de todos los miembros del equipo de desarrollo.

10. Conclusiones

A modo de conclusión se evaluará el cumplimiento de los objetivos establecidos al comienzo del proyecto. Se incluye además un resumen de las lecciones aprendidas de mayor importancia, y se definen los próximos pasos a seguir.

10.1. Evaluación de objetivos

A continuación se realizará un análisis del estado de los objetivos definidos inicialmente al momento de finalizar el proyecto.

10.1.1. Objetivos académicos

En cuanto a los objetivos definidos para el ámbito académico, se puede concluir que los mismos fueron cumplidos al momento de realizar este análisis.

En primer lugar, aunque aún no se conozca el puntaje obtenido por el equipo, el mismo aplicó los conocimientos técnicos y de gestión de proyectos incorporados durante el transcurso de la carrera. Se logró desarrollar un sistema en su totalidad, partiendo de repositorios de código vacíos. El equipo de sólo dos integrantes logró autogestionarse, llevando a cabo el desarrollo del sistema de forma sistemática, adaptándose a los cambios que surgieron, al mismo tiempo que se adaptaba el marco de trabajo utilizado.

En segundo lugar, es posible concluir que al desarrollar el sistema en cuestión, ambos integrantes del equipo adquirieron conocimientos en tecnologías de blockchain, tanto a partir del desarrollo como de la materia cursada como actividad de capacitación. Además de esto, uno de los integrantes no estaba familiarizado con los lenguajes utilizados para desarrollar el *front-end* o el *back-end*, por lo que su aprendizaje fue aún más extenso. El aprendizaje no se limitó a las tecnologías utilizadas para el desarrollo, ya que en el transcurso del proyecto se utilizaron varias

tecnologías y herramientas para apoyar el desarrollo en las cuales el equipo carecía de experiencia, como el caso de *VersionOne*, *Retrium*, *Codeship*, entre otras.

Finalmente, el equipo logró gestionar un proyecto de software real, aplicando todo el proceso de ingeniería de software fuera del ambiente académico. Se realizó el desarrollo de un sistema completo comenzando con conocimientos del dominio casi nulos. Se siguieron varios marcos de trabajo como el *dual track agile* y *Scrum*, realizando un proceso exhaustivo de ingeniería en requerimientos, para luego pasar a la gestión de proyecto que guió el proceso de desarrollo. Esto se vio reflejado en las encuestas de satisfacción realizadas a los usuarios, los cuales demostraron estar a gusto con el producto final desarrollado. Dichos resultados pueden ser vistos en la sección 8.5.3. Pruebas con usuarios.

10.1.2. Objetivos del equipo

En cuanto a los objetivos definidos para el equipo, es posible establecer que los mismos fueron cumplidos.

Por un lado, se mejoró y afianzó la relación entre ambos compañeros del equipo. Durante todo el transcurso del proyecto no hubo ningún altercado entre los mismos, y siempre hubo lugar para el diálogo y la discusión, lo cual se vio reflejado en la encuesta realizada acerca del clima de trabajo en el equipo, en la cual se obtuvieron resultados completamente satisfactorios. Es posible ver un mayor detalle de dicha encuesta en el anexo 12.24. Encuesta de clima de trabajo en el equipo.

Por otro lado, se mejoró la comunicación y el trabajo en equipo. Esto presentó un desafío en repetidas ocasiones durante el proyecto, ya que existieron tramos enteros en los cuales se debió trabajar a la distancia, existiendo cinco horas de diferencia. Sin embargo, aunque esto afectó la comunicación de a momentos, en ningún momento un integrante del equipo se apartó del reglamento del equipo definido.

10.1.3. Objetivos del producto

Finalmente, también es posible concluir que los objetivos relacionados al producto fueron cumplidos.

En primer lugar, se logró la satisfacción del cliente y los usuarios desarrollando un producto útil y de calidad, como lo marcó el resultado de las encuestas finales de satisfacción del cliente y usuarios. Esto fue el reflejo de un proceso exhaustivo de ingeniería en requerimientos, hecho con la consigna de lograr un conjunto de requerimientos que se adapte a las necesidades de los distintos tipos de usuarios que interactuarían con el sistema.

También se logró implementar el 100% de los requerimientos definidos como prioritarios entre el cliente y el equipo. Cuando surgieron cambios en los requerimientos, se realizó una re-priorización de los mismos, con el objetivo de que siempre queden dentro del alcance del proyecto aquellos de mayor prioridad.

Finalmente, se realizó la transferencia de conocimientos a Effectus de forma exitosa, a través de la generación de un documento que detalle las lecciones aprendidas, ventajas y desventajas de la tecnología Blockchain. Además del documento de transferencia, se les puso a disposición este documento con el fin de que puedan informarse más en profundidad del detalle de cómo fue todo el proceso del proyecto.

10.2. Lecciones aprendidas

A continuación, se listan las lecciones aprendidas más relevantes que le dejó al equipo realizar un proyecto completo de software de estas características, en el mismo orden en el cual está estructurado este documento.

Marco metodológico

Esta sección del proyecto le dejó al equipo como lección la importancia de elegir marcos metodológicos y de trabajo que se adapten a las necesidades de las distintas etapas del proyecto, y que no tiene por qué usarse un único marco de trabajo para cada proyecto. En el caso de este, se combinaron varios marcos de

trabajo como el *dual track agile*, *kanban*, y Scrum, donde cada uno de ellos le brindó un conjunto diferente de ventajas al equipo. Por un lado, el *dual track agile* permitió que el equipo defina dos etapas dentro del proyecto con objetivos diferenciados, una etapa teniendo como objetivo el relevamiento y especificación de requerimientos en un dominio del cual se carecía de conocimiento, y la otra teniendo como fin el desarrollo de las funcionalidades especificadas durante la primera etapa.

El equipo también se lleva como lección la importancia de realizar adaptaciones a los marcos de trabajo elegidos, los cuales no tiene por qué ser respetarlos al pie de la letra. Tal fue el caso del *dual track agile*, en el cual los dos ciclos no fueron concurrentes, sino que se definieron etapas del proyecto diferentes en los cuales la presencia de cada ciclo sería más marcada, con el *discovery cycle* en la etapa inicial y el *delivery cycle* en la etapa siguiente. También fueron importantes las adaptaciones realizadas al marco de trabajo Scrum, ya que al ser sólo dos integrantes, no fue necesario invertir tanto esfuerzo en la coordinación del equipo.

Ingeniería de requerimientos

Entre los principales aprendizajes que esta sección le dejó al equipo está la importancia de validar los requerimientos con los distintos usuarios antes de comenzar el desarrollo. Son los mismos usuarios quienes utilizarán el software, y si el mismo no se adapta a las necesidades de ellos, carece de valor. Si una validación con usuarios deja como resultado que cierta funcionalidad no es valiosa para los usuarios, la misma debe ser cambiada sin importar la etapa en la que el proyecto se encuentre. En el caso de este proyecto, dicho cambio surgió luego de desarrollada las funcionalidades, por lo que fue necesario realizar re-trabajo. Por esta razón, para evitar el re-trabajo, siempre es necesario realizar las validaciones con los grupos correspondientes en etapas iniciales, ya que esto podría haber sido evitado si se hubieran logrado validar las funcionalidades en cuestión con instituciones académicas previo a su desarrollo.

Además de esto, el equipo también aprendió la importancia de adaptarse a los cambios en los requerimientos, incorporándolos al alcance del proyecto según su

prioridad. No tiene ningún sentido resistirse a los cambios siempre y cuando estos surjan de *feedback* de clientes o usuarios. Fue necesario para el equipo comprender que los cambios que surgieron fueron debido a fallas del propio equipo durante el proceso de relevamiento de requerimientos.

Arquitectura y desarrollo

El cuanto a la arquitectura y desarrollo del sistema, esta le dejó varios aprendizajes al equipo. En primer lugar, una gran enseñanza fue la de llevar un proceso que inició en la ingeniería en requerimientos, con la definición de requerimientos no funcionales. A partir de estos, el equipo debió definir atributos de calidad, y luego determinar qué tácticas y patrones de arquitectura podrían satisfacerlos de la mejor manera. Realizar todo este proceso de principio a fin le dio al equipo una nueva perspectiva de cómo se especifica y desarrolla el software, ya que ninguno de los dos integrantes había participado en un proceso similar previo a la realización del proyecto.

Adicionalmente, el equipo ganó experiencia en la utilización de tecnologías de vanguardia como es el caso de *blockchain*, desarrollando una solución que utiliza un contrato inteligente alojado en una *blockchain* para proveer un mecanismo de verificación de integridad. Esto fue particularmente interesante, ya que por el tipo de datos sensibles que se trataban en el sistema, se tuvo que tener en cuenta ciertos aspectos legales al momento de tomar determinadas decisiones, como lo fue determinar qué información almacenar en la *blockchain* y de qué manera. Uno de los aspectos más positivos de este aprendizaje es que el mecanismo utilizado para verificar la integridad no es específico del dominio en el cual se trabajó, pues este puede ser utilizado para verificar la integridad de cualquier elemento al que se le pueda aplicar la función *hash*.

En resumen, desde el punto de vista de la arquitectura y desarrollo, el equipo aprendió acerca del proceso de diseño de una arquitectura a partir de requerimientos no funcionales definidos, y se interiorizó con el uso de nuevas tecnologías emergentes.

Gestión del proyecto

El principal aprendizaje que la gestión del proyecto le dejó al equipo fue el de la importancia de respetar el marco de trabajo definido durante el desarrollo. Una vez que se contó con los requerimientos definidos, el primer impulso del equipo, presionado por las fechas de entrega de los distintos *releases*, fue el de comenzar el desarrollo lo más rápido posible, sin prestar demasiada atención al proceso de gestión del proyecto. Afortunadamente no se actuó de acuerdo a este impulso, y se mantuvieron reuniones con el objetivo de planificar y coordinar los esfuerzos a realizar durante los *sprints*, y también otras con el objetivo de reflexionar acerca del proceso, buscando puntos de mejora. Probablemente, apegarse a las pautas definidas por el marco de trabajo Scrum fue lo que le permitió al equipo mantener un ritmo de trabajo sostenido y un nivel de coordinación aceptable mientras sus integrantes se encontraban en países con husos horarios diferentes.

Gestión de riesgos

El equipo aprendió la importancia de realizar una gestión de riesgos bien meditada que se adecue al proyecto. En varias ocasiones durante el proyecto, los riesgos definidos se materializaron, teniendo consecuencias negativas en el progreso hacia los objetivos del proyecto. En estas ocasiones, fue de gran ayuda tener procesos preparados, con técnicas de mitigación de riesgos y planes de contingencia. De esta forma, se logró limitar el impacto que estos tuvieron en el proyecto, y se logró enfrentar los riesgos de forma ordenada. Es altamente probable que en caso de no haber realizado una gestión de riesgos adecuada, el impacto de los riesgos hubiera sido mucho mayor.

Gestión de calidad

La principal enseñanza que la gestión de calidad le dejó al equipo fue la de cuán importante es la planificación y ejecución de las pruebas del software. Todas las etapas del proceso de prueba definido en la adaptación del modelo V tuvieron su impacto en el aumento de la calidad del producto final. Comenzando con las pruebas unitarias, las cuales no sólo evitan la aparición de defectos durante el

desarrollo, sino que además también facilitan la introducción de cambios y el refactoreo del código al contar con pruebas que comprueban su correcto funcionamiento. Luego, las pruebas de sistema y regresión permitieron la detección de defectos presentes en el código para su posterior corrección, evitando que estos fallos pasen a formar parte de los *releases*. Finalmente, a través las pruebas con usuarios el equipo logró diferenciar aquellas tareas que eran demasiado difíciles como para ser completadas por la mayoría de los usuarios. A raíz de esto, se realizaron cambios para simplificarlas y se obtuvieron mejores resultados en pruebas con usuarios posteriores.

En caso de no haber contado con este proceso, es altamente probable que no se hubieran podido cumplir algunos objetivos asociados a la calidad del producto, por lo que se puede concluir que la gestión de calidad fue vital para el éxito del proyecto y el cumplimiento de sus objetivos.

Gestión de la configuración

La gestión de la configuración dejó como principal enseñanza al equipo la importancia del uso de herramientas de automatización con el fin de facilitar varios procesos asociados al desarrollo y la gestión, ahorrando de este modo tiempo y esfuerzo valiosos. La principal ventaja que el equipo obtuvo a partir de la correcta gestión de la configuración fue la de contar con un proceso de despliegue continuo y automático, desencadenado cada vez que nuevo código era subido a los repositorios. Esto sumado al manejo de las dependencias permitió que el equipo ahorra tiempo en configuración y despliegue, pudiendo dedicarle este tiempo a procesos de mayor prioridad.

10.3. Estado actual del sistema

Actualmente, la aplicación web se encuentran en una versión beta. Se implementaron aquellas funcionalidades acordadas de mayor prioridad, pero aún hay historias pendientes en el *backlog* del producto. De momento, el sistema almacena el *hash* de toda información asociada a calificaciones y de todos los archivos almacenados en una blockchain de prueba, a través de un contrato

inteligente desplegado en la misma. Los *front-ends* de los distintos ambientes se encuentran alojados en *buckets* de S3.

Durante la defensa del proyecto, se realizará una demostración del funcionamiento de la aplicación web en el ambiente *beta*.

10.4. Próximos pasos

Hasta el momento aún es incierto si *Effectus* deseará continuar con el proyecto o si tomará las lecciones aprendidas en el mismo para aplicar la tecnología de blockchain en alguna otra problemática. En caso de desear seguir con el proyecto en cuestión, algunos de los pasos que podría tomar *Effectus* serían los siguientes:

Culminar el desarrollo de las funcionalidades pendientes

Como fue mencionado en la sección anterior, algunas funcionalidades aún quedan pendientes para ser desarrolladas. Si bien estas son de baja prioridad, entre ellas hay algunas que aumentarían la usabilidad del sistema, reforzando la prevención de errores, como por ejemplo la de hacer posible que las instituciones académicas incluyan restricciones en los requisitos de los estudiantes (por ejemplo que un campo sea de un largo fijo, o numérico).

Avance en blockchain a utilizar

Se debe decidir si se implementará la idea de levantar una *blockchain* privada en la que cada institución académica cuente con un nodo, y de esta forma todas participen de la red, aumentando su confianza en la misma. En caso de no desear seguir adelante con esta idea, de todos modos deberán desplegar el contrato inteligente en una red que no sea de prueba.

Modificar nombre de dominio de la página web

Hoy en día, las aplicaciones web de los distintos ambientes se encuentran alojadas en *buckets* de S3. Aún no cuentan con un nombre de dominio simple y descriptivo, sino que su nombre es el asignado automáticamente por Amazon. En caso de hacer

pública la aplicación web, sería necesario elegir un nombre de dominio más simple. A modo de ejemplo, el dominio de la aplicación web en el ambiente de QA es el siguiente http://qa.diplovalid.host.com.s3-website-us-east-1.amazonaws.com.

Monetización del producto

Al tratarse de un MVP, el equipo aún no ha considerado cómo se sacaría rédito del sistema desarrollado. Este aspecto escapa del alcance del proyecto, aunque el equipo cree que existen posibilidades de monetizar el producto si se elige la estrategia correcta.

Mejorar estética de la aplicación

Como fue visto en la sección 8.5.3. Pruebas con usuarios, El puntaje más bajo obtenido en la encuesta de satisfacción fue en la pregunta relacionada con qué tan agradable era de utilizar la aplicación. A raíz de esto, uno de los posibles próximos pasos podría ser mejorar la estética de la aplicación, quizás involucrando a diseñadores gráficos de interfaz de usuario en el proceso de construcción.

11. Referencias bibliográficas

[1] The Economist, "The great chain of being sure about things", ~~The Economist,~~ 2020. [Online~~En línea~~]. Available~~Disponible en~~: https://www.economist.com/briefing/2015/10/31/the-great-chain-of-being-sure-about-things. [Accessed~~Accedido~~: 04- Mar- 2020].

[2] Doran, G. T. (1981). "There's a S.M.A.R.T. way to write management's goals and objectives". *Management Review*. 70 (11): 35–36.

[3] Graduados.ort.edu.uy, "Misión y objetivos - Graduados - Universidad ORT Uruguay", ~~Graduados.ort.edu.uy,~~ 2020. [Online~~En línea~~]. Available~~Disponible en~~: https://graduados.ort.edu.uy/index.php?id=AAAAAA. [Accessed~~Accedido~~: 04- Mar- 2020].

[4] Ministerio de Educación y Cultura, "Cometidos", ~~Ministerio de Educación y Cultura,~~ 2020. [Online~~En línea~~]. Available~~Disponible en~~: https://www.gub.uy/ministerio-educacion-cultura/institucional/cometidos. [Accessed~~Accedido~~: 04- Mar- 2020].

[5] Impo.com.uy, "Ley Nº 18331", ~~Impo.com.uy,~~ 2020. [Online~~En línea~~]. Available~~Disponible en~~: https://www.impo.com.uy/bases/leyes/18331-2008. [Accessed~~Accedido~~: 04- Mar- 2020].

[6] EL PAÍS, ~~E. País,~~ "Reportaje | Los títulos falsos proliferan en la Red", ~~EL PAÍS,~~ 2020. [Online~~En línea~~]. Available~~Disponible en~~: https://elpais.com/diario/2007/11/26/sociedad/1196031603_850215.html. [Accessed~~Accedido~~: 04- Mar- 2020].

[7] Atlassian, "Kanban vs Scrum | Atlassian", ~~Atlassian,~~ 2020. [Online~~En línea~~]. Available~~Disponible en~~: https://www.atlassian.com/agile/kanban/kanban-vs-scrum. [Accessed~~Accesido~~: 04- Mar- 2020].

[8] Vertical Distinct, L. Wolf, "The 10 Benefits of Kanban - Vertical Distinct", Vertical Distinct, 2020. [Online En línea]. Available Disponible en: https://verticaldistinct.com/the-10-benefits-of-kanban/. [Accessed Accedido: 04- Mar- 2020].

[9] Scrumguides.org, 2020. [Online En línea]. Available Disponible en: https://www.scrumguides.org/docs/scrumguide/v2017/2017-Scrum-Guide-Spanish-SouthAmerican.pdf. [Accessed Accedido:: 04- Mar- 2020].

[10] toolshero, "What is the Value Proposition Canvas? Summary + template | ToolsHero", toolshero, 2020. [Online En línea]. Available Disponible en: https://www.toolshero.com/marketing/value-proposition-canvas/. [Accessed Accedido: 04- Mar- 2020].

[11] Checkdiploma.org, "Diploma-Blockchain: Diploma-Blockchain", Checkdiploma.org, 2020. [Online En línea]. Available Disponible en: https://checkdiploma.org/works#technical. [Accessed Accedido: 04- Mar- 2020].

[12] Blockcerts, "Blockchain Credentials", Blockcerts, 2020. [Online En línea]. Available Disponible en: https://www.blockcerts.org/guide/. [Accessed Accedido: 04- Mar- 2020].

[13] Hinge Marketing, "In-Depth Interview vs Online Survey: A Marketing Director's Guide to Brand Research Data Collection | Hinge Marketing", Hinge Marketing, 2020. [Online En línea]. Available Disponible en: https://hingemarketing.com/blog/story/in-depth-interviews-vs-online-surveys-a-marketing-directors-guide-for-brand. [Accessed Accedido: 04- Mar- 2020].

[14] Agencia de Gobierno Electrónico y Sociedad de la Información y del Conocimiento, "Qué es la Firma Digital", Agencia de Gobierno Electrónico y Sociedad de la Información y del Conocimiento, 2020. [Online En línea]. Available Disponible en:

https://www.gub.uy/agencia-gobierno-electronico-sociedad-informacion-conocimiento/que-es-la-firma-digital. [AccessedAccedido: 04- Mar- 2020].

[15] Gub.uy, 2020. [OnlineEn línea]. AvailableDisponible en: https://www.gub.uy/ministerio-salud-publica/sites/ministerio-salud-publica/files/2019-01/manual_firma_electronica_v3.pdf. [AccessedAccedido: 04- Mar- 2020].

[16] Iddigital.com.uy, 2020. [OnlineEn línea]. AvailableDisponible en: https://www.iddigital.com.uy/es/ayuda-y-soporte/preguntas-frecuentes/. [AccessedAccedido: 04- Mar- 2020].

[17] Ronjeffries.com, "Stories and the Three C's - Revisited Yet Again", Ronjeffries.com, 2020. [OnlineEn línea]. AvailableDisponible en: https://ronjeffries.com/articles/019-01ff/stories-3cs-yet-again/. [AccessedAccedido: 04- Mar- 2020].

[18] Innolution.com, "Chapter 5 of "Essential Scrum": Requirements, User Stories | Innolution", Innolution.com, 2020. [OnlineEn línea]. AvailableDisponible en: https://innolution.com/essential-scrum/table-of-contents/chapter-5-requirements-and-user-stories. [AccessedAccedido: 04- Mar- 2020].

[19] Agilebusiness.org, "Chapter 10: MoSCoW Prioritisation", Agilebusiness.org, 2020. [OnlineEn línea]. AvailableDisponible en: https://www.agilebusiness.org/page/ProjectFramework_10_MoSCoWPrioritisation. [AccessedAccedido: 04- Mar- 2020].

[20] Aha.io, "What is user story mapping? | Aha!", Aha.io, 2020. [Online]. Available: https://www.aha.io/roadmapping/guide/release-management/what-is-user-story-mapping. [Accessed: 04- Mar- 2020].

[21] L. Bass, P. Clements and R. Kazman, *Software architecture in practice*. Upper Saddle River, NJ: Addison-Wesley, 2015.

[22] the morning paper, "A concurrent perspective on smart contracts", the morning paper, 2020. [OnlineEn línea]. AvailableDisponible: https://blog.acolyer.org/2017/08/30/a-concurrent-perspective-on-smart-contracts/. [AccessedAccedido: 04- Mar- 2020].

[23] Google Docs, "RSK Educate - Executive Course", Google Docs, 2020. [OnlineEn línea]. AvailableDisponible en: https://drive.google.com/file/d/1nATjtMEEINSbYEaAceNEXno-2nLr2MG0/view. [AccessedAccedido: 04- Mar- 2020].

[24] BitInfoCharts, A. chart, "Ethereum Avg. Transaction Fee chart", BitInfoCharts, 2020. [OnlineEn línea]. AvailableDisponible en: https://bitinfocharts.com/comparison/ethereum-transactionfees.html. [AccessedAccedido: 04- Mar- 2020].

[25] BitInfoCharts, A. chart, "Bitcoin Avg. Transaction Fee chart", BitInfoCharts, 2020. [OnlineEn línea]. AvailableDisponible en: https://bitinfocharts.com/comparison/bitcoin-transactionfees.html. [AccessedAccedido: 04- Mar- 2020].

[26] Kilthub.cmu.edu, 2020. [OnlineEn línea]. AvailableDisponible en: https://kilthub.cmu.edu/articles/Modifiability_Tactics/6575696/files/12062342.pdf. [AccessedAccedido: 04- Mar- 2020].

[27] Softeng.nju.edu.cn, 2020. [OnlineEn línea]. AvailableDisponible en: http://softeng.nju.edu.cn/tech_reports/TR-18-002-MSQ.pdf. [AccessedAccedido: 04- Mar- 2020].

[28] Tools.ietf.org, "RFC 7519 - JSON Web Token (JWT)", Tools.ietf.org, 2020. [OnlineEn línea]. AvailableDisponible en: https://tools.ietf.org/html/rfc7519. [AccessedAccedido: 04- Mar- 2020].

[29] Reactjs.org, "React – A JavaScript library for building user interfaces", Reactjs.org, 2020. [OnlineEn línea]. AvailableDisponible en: https://reactjs.org/. [AccessedAccedido: 04- Mar- 2020].

[30] Blockgeeks, "Smart Contract Platforms Comparison: RSK vs Ethereum vs EOS vs Cardano", Blockgeeks, 2020. [OnlineEn línea]. AvailableDisponible en: https://blockgeeks.com/guides/smart-contract-platforms-comparison-rsk-vs-ethereum-vs-eos-vs-cardano/. [AccessedAccedido: 04- Mar- 2020].

[31] Stateofthedapps.com, "State of the DApps — DApp Statistics", Stateofthedapps.com, 2020. [OnlineEn línea]. AvailableDisponible en: https://www.stateofthedapps.com/stats. [AccessedAccedido: 04- Mar- 2020].

[32] Dan North & Associates, "What's in a Story?", Dan North & Associates, 2020. [OnlineEn línea]. AvailableDisponible en: https://dannorth.net/whats-in-a-story/. [AccessedAccedido: 04- Mar- 2020].

[33] martinfowler.com, M. Fowler, "bliki: GivenWhenThen", martinfowler.com, 2020. [OnlineEn línea]. AvailableDisponible en: https://martinfowler.com/bliki/GivenWhenThen.html. [AccessedAccedido: 04- Mar- 2020].

[34] Scrum.org, "What is Sprint Planning?", Scrum.org, 2020. [OnlineEn línea]. AvailableDisponible en: https://www.scrum.org/resources/what-is-sprint-planning. [AccessedAccedido: 04- Mar- 2020].

[35] M. Cohn, *Agile estimating and planning*. Upper Saddle River, NJ: Prentice Hall PTR, 2012.

[36] La guía PMBOK, "8. Gestión de los Riesgos del Proyecto", La guia PMBOK, 2020. [OnlineEn línea]. AvailableDisponible en: http://uacm123.weebly.com/8-gestioacuten-de-los-riesgos-del-proyecto.html. [AccessedAccedido: 04- Mar- 2020].

[37] R. Martin, *Clean code*, 11th ed. Prentice Hall, 2008.

[38] GitHub, "airbnb/javascript", GitHub, 2020. [OnlineEn línea]. AvailableDisponible en: https://github.com/airbnb/javascript/tree/master/react. [AccessedAccedido: 04- Mar- 2020].

[39] Standardjs.com, "JavaScript Standard Style", Standardjs.com, 2020. [OnlineEn línea]. AvailableDisponible en: https://standardjs.com/#can-i-use-a-javascript-language-variant-like-flow-or-typescript. [AccessedAccedido: 04- Mar- 2020].

[40] Solidity.readthedocs.io, "Style Guide — Solidity 0.6.4 documentation", Solidity.readthedocs.io, 2020. [OnlineEn línea]. AvailableDisponible en: https://solidity.readthedocs.io/en/latest/style-guide.html. [AccessedAccedido: 04- Mar- 2020].

[41] Ort.edu.uy, 2020. [OnlineEn línea]. AvailableDisponible en: http://www.ort.edu.uy/reglamentos/normas-especificas-para-la-presentacion-de-trabajos-finales-de-carrera-facultad-de-ingenieria-excepto-biotecnologia__documento-302.pdf. [AccessedAccedido: 04- Mar- 2020].

[42] Ort.edu.uy, 2020. [OnlineEn línea]. AvailableDisponible en: http://www.ort.edu.uy/reglamentos/hoja-de-verificacion-de-pautas-de-presentacion-de-trabajos-finales-de-carreras-excepto-biotecnologia__documento-303.pdf. [AccessedAccedido: 04- Mar- 2020].

[43] Ort.edu.uy, 2020. [OnlineEn línea]. AvailableDisponible en: https://www.ort.edu.uy/varios/pdf/documento306.pdf. [AccessedAccedido: 04- Mar- 2020].

[44] Testbytes, "What is V-model and W-model in Software Testing [What are They]", Testbytes, 2020. [OnlineEn línea]. AvailableDisponible en: https://www.testbytes.net/blog/v-model-and-w-model-software-testing/. [AccessedAccedido: 04- Mar- 2020].

[45] Citeseerx.ist.psu.edu, 2020. [OnlineEn línea]. AvailableDisponible en: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.206.3052&rep=rep1&type=pdf. [AccessedAccedido: 04- Mar- 2020].

[46] Git-scm.com, "About - Git", Git-scm.com, 2020. [OnlineEn línea]. AvailableDisponible en: https://git-scm.com/about. [AccessedAccedido: 04- Mar- 2020].

[47] Atlassian, "Gitflow Workflow | Atlassian Git Tutorial", Atlassian, 2020. [OnlineEn línea]. AvailableDisponible en: https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow. [AccessedAccedido: 04- Mar- 2020].

12. Anexos

12.1. Documento de transferencia de conocimiento a Effectus

Effectus Software

Análisis y resultados obtenidos del uso de tecnología blockchain en un proyecto real

Autores:

Bruno Gallo

Marcos Irisarri

Diciembre 2019

Introducción

El presente documento es el resultado de la construcción de una plataforma web de registro de diplomas académicos utilizando *blockchain* como tecnología de almacenamiento. Se incluye un análisis del problema a solucionar, la solución planteada, ventajas y desventajas del uso de la tecnología, comparación con otras alternativas de almacenamiento de datos, y las conclusiones obtenidas.

Acerca del proyecto

Sólo en EEUU se venden más de 100.000 diplomas falsos cada año. Por tan sólo U$S650 es posible conseguir un diploma de Harvard, incluso hay sitios web donde uno puede comparar precios.

Hoy en día, una nueva tecnología de base de datos permite a los usuarios insertar información la cual se asegura inmutable en el tiempo.

Se creó una herramienta que registra la información de diplomas y títulos en una *blockchain*. Mediante el uso de esta tecnología además se brinda fácilmente la posibilidad de que cualquier tercero pueda consultar la validez de un diploma a través de un simple vínculo. No siendo posible alterar registros existentes se asegura que una vez que un diploma fue corroborado por una institución, este no podrá ser cambiado ni falsificado.

De esta forma cualquier persona puede compartir sus logros y títulos en linkedin, curriculums, o con potenciales empleadores mediante un simple vínculo.

Hoy en día la única forma de verificar la veracidad de un diploma es contactando a la institución que lo emitió. El objetivo a largo plazo es que este sistema funcione como una alternativa viable a los métodos ya existentes.

A modo de ejemplo podrían registrarse:

- Títulos universitarios, o académicos de cualquier tipo
- Certificados de idiomas
- Certificados de logros deportivos
- Participaciones en talleres, charlas o eventos

En principio, la forma de lograr esto sería mediante un sistema en el cual puedan registrarse tanto instituciones académicas como egresados de las mismas. Los egresados registrarían sus calificaciones y solicitarían la verificación a las instituciones correspondientes, quienes ingresando al perfil de los egresados

verificarían o no las calificaciones dependiendo de la veracidad de los datos ingresados. Luego, con las calificaciones verificadas, el egresado podrá generar un vínculo que podrá ser utilizado por terceros para ingresar a su perfil y ver sus calificaciones verificadas.

Comparativa de soluciones

Base de datos tradicional

La solución “tradicional” a este problema implicaría la creación de una plataforma con las mismas funcionalidades que las planteadas donde los datos de las certificaciones, usuarios, e instituciones se guarden en una base de datos centralizada. Para asegurar la integridad y autenticidad de los diplomas se podría hacer uso de documentos firmados digitalmente mediante el uso de claves públicas y privadas.

Ese tipo de arquitectura se la conoce como cliente - servidor, donde un usuario (el cliente) puede modificar la información almacenada en una base de datos centralizada alojada en un servidor. El control de la base de datos recae sobre un administrador encargado además de brindar permisos a los distintos usuarios para que realicen distintas operaciones (lectura, escritura).

Ventajas

- Datos estructurados y consultas rápidas, lo que impactaría de forma positiva la performance del sistema.
- Escalabilidad: es posible manejar miles de registros manteniendo la velocidad de respuesta.

Desventajas

- Por tratarse de un sistema centralizado, hay un único punto de falla. En caso de caer el servidor o ser hackeado, caería toda la aplicación. Además un único grupo de personas tiene el control sobre la información lo que la hace

vulnerable a las acciones de terceros. Dada la sensibilidad de los datos almacenados, esta desventaja tiene un peso considerable.

- Se depende exclusivamente de un administrador de base de datos. En caso de perder las credenciales o renuncia del DBA se vuelve un proceso tedioso el cambio de contraseñas para mantener la seguridad. Este proceso sería necesario, teniendo en cuenta la importancia de los datos, y el daño que podría ocasionar un tercero en caso de obtener las credenciales del administrador.
- Seguridad. En caso de no tener el software actualizado hackers podrìan aprovechar alguna vulnerabilidad conocida para lograr acceder a la base y modificar la data. La modificación de la misma, al ser un sistema centralizado, es de mayor facilidad.

Blockchain

Otra opción es el uso de blockchain para el registro de diplomas. En este caso la arquitectura es distribuida entre varios nodos en la red donde cada uno tiene una copia completa de toda la blockchain. Antes de ingresar un nuevo registro es necesario que el 50% + 1 de los nodos de la red lo aprueben. Además, en este caso las únicas operaciones permitidas son escritura y consulta, no es posible modificar ni eliminar registros.

Ventajas

- Descentralización. La información se guarda en múltiples nodos independientes conectados entre sí.
- Alta tolerancia a fallas: si falla un nodo hay muchos otros brindando el servicio, aumentando el *uptime* de la aplicación. No hay un punto único de falla.
- Seguridad, al tener la información replicada en cada nodo es casi imposible alterar un registro. Se debería violar más del 50% de la red. Esto disminuye considerablemente la probabilidad de que un atacante pueda registrar diplomas de forma ilícita.

- Inmutabilidad. Una vez registrado un bloque ya no puede cambiarse. En caso de cambiarse algún valor es posible rastrear hacia atrás sus valores anteriores. Esto significa que una vez que un diploma quede validado, dicha data sería imposible de modificar.
- Transparencia, es una plataforma abierta en donde cualquiera puede consultar el historial de registros lo que brinda mayor confianza, algo necesario en una plataforma de registro y consulta de diplomas.

Desventajas

- Consumo energético. Debido al protocolo definido para escribir nuevos bloques se requiere mucha capacidad de cómputo y esto genera un alto consumo de energía eléctrica. Esta es una desventaja en algunas *blockchains* en sí, aunque no afectaría el funcionamiento de la aplicación.
- Escalabilidad, ya que los bloques en general son de un tamaño pequeño puede volverse lento realizar consultas a la blockchain cuando se tienen varios miles de registros, lo cual afectaría la performance de la aplicación en forma negativa.
- Tamaño, es necesario que cada nodo tenga capacidad de almacenar varios *gigabytes* de información ya que se va guardando un registro de cada cambio hecho en la blockchain en cada nodo. Al igual que el consumo energético, esta es una desventaja que no afectaría el funcionamiento de la aplicación.
- Interoperabilidad, hoy en día no es sencillo interconectar distintas blockchain entre sí. Cada una es más bien su propio ecosistema. Esto puede dificultar migrar la aplicación de una blockchain a otra.

Elección de solución

Las base de datos tradicionales son mejores para sistemas empresariales gracias a su estabilidad y capacidad de manejar miles de registros a la vez sin sacrificar performance. Es la mejor opción para sistemas que necesitan guardar grandes volúmenes de datos sin encriptar y realizar distintas consultas y análisis sobre estos

datos. Ej: sistemas de e-commerce, análisis de compra y venta de acciones en la bolsa, datos estadísticos.

Una *blockchain* es mejor para el caso donde se prioriza la confianza y la transparencia. Es particularmente útil en casos donde se quiere garantizar que no hay fraude y llevar un registro transparente de cada movimiento, tal como es el caso de registro y consulta de diplomas.

Creemos que las características de la aplicación a desarrollar coinciden claramente con el caso en que se recomienda usar blockchain. Si bien utilizar las *blockchains* presentan algunas desventajas considerables, la aplicación en cuestión se basa en la seguridad y la confianza en que los certificados presentados son efectivamente reales, y no es la principal prioridad tener la capacidad de procesar miles de transacciones por segundo en tiempos reducidos.

Solución desarrollada

A continuación, se detallará el funcionamiento de la aplicación desarrollada, con sus distintos tipos de usuario y las tareas que puede realizar cada uno dentro del sistema. La aplicación cuenta con tres tipos de usuarios: administradores, instituciones académicas, y egresados.

Se comenzará con el registro de las instituciones. Este sucedería luego de una reunión entre un administrador del sistema y un representante de dicha institución. En esta reunión, se explicaría el funcionamiento de la aplicación, y se crearía el perfil de la institución. Es el administrador quien crea dicho perfil, y al momento de hacerlo, también determina las credenciales con las cuales un representante de la institución académica ingresará al sistema. Tanto el representante de las instituciones académicas como los administradores pueden editar el perfil de las instituciones académicas, agregando, modificando, o eliminando requisitos y títulos.

Los requisitos determinan información que los egresados deberán ingresar con la finalidad de verificar su identidad. Los mismos pueden tomar la forma de campos o de archivos. Un ejemplo de un requisito de tipo “campo” puede ser un número de

estudiante, o un número de documento. Un ejemplo de un requisito de tipo archivo (imagen o PDF) puede ser el consentimiento del egresado, que es una carta firmada por el egresado dándole autorización a la institución a verificar las calificaciones ingresadas. Esto es necesario para que las instituciones puedan verificar las calificaciones ya sea a través de un sistema, o de una llamada telefónica por parte de un posible empleador.

Los títulos de las instituciones académicas representan todos las titulaciones ofrecidas por la institución. Estas podrán ser seleccionadas por los egresados para ingresar calificaciones. Como las instituciones pueden llegar a ofrecer cientos de títulos, se agregó la posibilidad de cargarlos a partir de un archivo CSV.

Por otro lado, los egresados pueden registrarse libremente en el sistema. Cuando ingresan al mismo, pueden editar sus datos personales y elegir una imagen de perfil. Además de esto, pueden seleccionar instituciones de entre las registradas en el sistema. Una vez que seleccionan una institución, pueden seleccionar de entre los títulos ingresados por la misma, ingresar información asociada al título, y adjuntar archivos al mismo, tales como imágenes del diploma obtenido. Previo a solicitar la verificación de la calificación por parte de la institución, los egresados deben ingresar información para todos los requisitos ingresados por dicha institución. Una vez hecho esto, podrán solicitar la verificación. En este momento, la calificación queda en estado “pendiente”.

Todos los datos asociados a calificaciones ingresadas por el egresado es asegurada con un mecanismo de verificación de integridad. Este es utilizado tanto para la información ingresada (promedio o fecha de recibimiento por ejemplo), como para los archivos asociados a la misma. El mecanismo consiste en el almacenamiento de un *hash* de los datos en una *blockchain*, de modo de asegurar su inmutabilidad. Cada vez que los datos son recuperados, se verifica que el *hash* de los mismos coincida con el almacenado en la *blockchain*, y de este modo se asegura que estos no hayan sido modificados de forma ilícita.

Las instituciones académicas, además de poder acceder a su perfil, pueden acceder a la lista de solicitudes de verificación pendientes. Los administradores no pueden acceder a estas listas. A través de dicha lista pueden acceder al perfil de los egresados que realizaron estas solicitudes. Dentro del perfil de los egresados, las instituciones solamente pueden ver la información ingresada por el egresado asociada su propia institución. También pueden descargar los archivos asociados a requerimientos o calificaciones. Se verifica la integridad de toda la información a la que acceden utilizando el mecanismo mencionado anteriormente.

En caso de que la información ingresada por el egresado sea veraz, pueden aprobar la solicitud de verificación. En caso de que la información no sea veraz, pero solamente requiera un pequeño cambio, pueden solicitar un cambio en la misma, y junto con la solicitud incluir un mensaje de qué debería cambiar. Finalmente, en caso de que la información no sea veraz y se sospeche que existe intención de realizar fraude, la institución académica puede rechazar la solicitud, en cuyo caso automáticamente se bloquea el usuario del egresado.

El sistema almacena un historial de estados para cada calificación. Para cada estado además se almacena una *snapshot* de la calificación en el momento que pasó por ese estado. En este historial queda registrado el momento en el que el egresado solicita la verificación (junto al estado de la calificación al momento de solicitarla), y el momento en el que la institución solicita cambios, aprueba, o rechaza la solicitud de verificación, junto con el estado de la calificación al momento de procesar la solicitud de verificación. También se protege la integridad de este historial utilizando el mecanismo detallado anteriormente.

Una vez que un egresado cuenta con al menos una calificación verificada por una institución académica, este puede generar un vínculo de duración variable que puede ser utilizado para ingresar al perfil del egresado. Aquel que utilice el vínculo podrá sólo ver las calificaciones verificadas, pudiendo también descargar los archivos asociados. Toda la información que es exhibida está respaldada por el mecanismo de verificación de integridad, por lo que se asegura que no ha sido

modificada de forma no autorizada. En caso de que la información haya sido modificada, la misma no es mostrada.

Elección de blockchain

Tipo de blockchain - pública vs privada

En cuanto a las elecciones relacionadas con blockchain, el equipo debió primero decidir qué tipo de blockchain utilizar. Se debió considerar tanto las blockchain públicas abiertas como las privadas cerradas.

En las blockchain públicas, todos pueden formar parte de ellas y explorarlas. Estas son sus 3 características principales:

1. Cualquiera puede descargar el código y comenzar a ejecutar un nodo público en su dispositivo local, validando las transacciones en la red, participando así en el proceso de consenso, el proceso para determinar qué bloques se agregan a la *blockchain* y cuál es el estado actual.
2. Cualquier persona en el mundo puede enviar transacciones a través de la red y esperar verlas incluidas en la *blockchain* de bloques si son válidas.
3. Cualquiera puede leer la transacción en el explorador de bloques público.

Tienen los siguientes efectos:

- Potencial para alterar los modelos comerciales actuales a través de la desintermediación
- Sin costos de infraestructura para los usuarios (el único costo es la tarifa de cada transacción). No es necesario mantener servidores o administradores de sistemas. Esto reduce radicalmente los costos de crear y ejecutar aplicaciones descentralizadas (dApps).

En las *blockchains* privadas, los permisos de escritura se mantienen centralizados para una organización o un conjunto específico de participantes (partes confiables). Los permisos de lectura pueden ser públicos o restringidos de forma arbitraria. Las

aplicaciones probables incluyen administración de bases de datos, auditorías y otras que son internas a una sola compañía, por lo que la legibilidad pública puede no ser necesaria en absoluto. Las *blockchains* privadas son una forma de aprovechar la tecnología *blockchain* mediante la creación de grupos y participantes que pueden verificar las transacciones internamente. Las *blockchains* privadas tienen su caso de uso, especialmente cuando se trata de escalabilidad y cumplimiento del estado de las reglas de privacidad de datos y otros problemas regulatorios.

Tienen los siguientes efectos:

- Reducción de los costos de transacción y la redundancia de datos, y reemplazamiento de los sistemas heredados (*legacy*), simplificando el manejo de documentos y eliminando los mecanismos de cumplimiento semi manuales.
- Reduce los costos, pero no es disruptivo. [1]

	Públicas sin permisos	Privadas con permisos
Acceso	Lectura y escritura, públicos para cualquiera	Lectura y escritura, sólo con invitación
Actores de la red	No se conocen entre ellos	Se conocen entre ellos
Velocidad	Lenta	Rápida

Tabla 15: Comparación de blockchains

En una reunión con Alejandro Narancio, CEO de *Infuy*, surgió la idea de utilizar una blockchain privada y cerrada, en la cual cada institución académica pueda formar parte de la blockchain, participando de la blockchain como un nodo. De este modo, formando parte de la red, las instituciones se asegurarían de que están contribuyendo a la seguridad de la red, participando en el proceso de consenso.

Teniendo en cuenta ambas opciones analizadas y la sugerencia del experto en el dominio, se decidió utilizar una blockchain privada con permisos. En primer lugar,

esto se debió a que no se desea que cualquiera pueda escribir o leer en la *blockchain*, sino que solamente aquellos autorizados para hacerlo, que en este caso sería el *back-end* de la aplicación.

Esta idea fue posteriormente validada con Pablo San Nicolás, secretario de gestión académica de ORT, quien agregó que aunque le parecía una buena idea, “todo el asunto está todavía muy en pañales” como para saber si esa sería la forma definitiva. Por esta razón, aunque se continuó con la idea de la blockchain privada, se tuvo en cuenta que esta no era una decisión particularmente estable.

Además, como la idea propuesta por un experto en el dominio fue que cada institución académica en lo posible tenga un nodo, se dividirían los costos de infraestructura, quedando solamente el costo de cada transacción a realizar. Con las blockchains públicas, se debería cubrir el costo de cada transacción. Tomando en cuenta la cantidad de egresados y el costo promedio de una transacción, la suma de los costos de todas las transacciones podría tornarse considerable. A modo de ejemplo, a principios de julio del año 2019, la transacción promedio en Ethereum valía U$S 0,24 [2], y en bitcoin valía U$S 4,04 [3]. En uruguay, en el año 2016 egresaron 12.665 alumnos. Viendo estos números, es posible llegar a la conclusión de que los costos de las transacciones podrían dispararse en un futuro.

Plataforma de *blockchain*

Al momento de decidir qué plataforma de blockchain utilizar, se debieron tomar en cuenta las necesidades del sistema. El principal requerimiento que el sistema tiene en cuanto a la tecnología de *blockchain* a utilizar es que esta debe permitir el desarrollo y ejecución de contratos inteligentes. Esto hizo que no se tuviera en cuenta la blockchain más popular, Bitcoin. El otro requerimiento principal es que la tecnología a utilizar permita la fácil migración a una *blockchain* pública, puesto que según lo discutido con San Nicolás, la decisión de utilizar una *blockchain* privada no es definitiva. Por esta razón, también se descartó la posibilidad de usar *Hyperledger Fabric*, un *framework* de código abierto para implementar *blockchains* privadas.

Se consideraron varias opciones en relación a qué plataforma utilizar. Finalmente, la decisión se redujo a Ethereum, EOS y RSK.

Ethereum

En 2013, un programador ruso-canadiense, Vitalik Buterin, lanzó el informe técnico de Ethereum y presentó al mundo las plataformas de contratos inteligentes. Desde entonces, Ethereum se ha fortalecido y se ha reunido para construir una de las comunidades de desarrolladores más saludables en el espacio criptográfico.

Si bien Buterin estaba intrigado por bitcoin y su potencial utilidad, sintió que la tecnología *blockchain* tenía mucho más que ofrecer que sólo ser un sistema de pago. Imaginó un futuro en el que los desarrolladores pudieran crear sus aplicaciones en la *blockchain*. Ethereum iba a ser una supercomputadora global descentralizada que alquilaría poder de cómputo a los desarrolladores para crear sus aplicaciones descentralizadas o dApps.

El desarrollo de Ethereum comenzó a principios de 2014 a través de una empresa suiza, Ethereum Switzerland GmbH (EthSuisse).

Ethereum utiliza una variante del consenso de Nakamoto, que se basa en la prueba de trabajo, generalmente se le conoce como PoW (*proof of work*). En ese sentido, Ethereum es relativamente similar a Bitcoin. Sin embargo, durante su etapa final, se actualizará a un nuevo protocolo de consenso llamado Casper FFG, que generalmente se conoce como *proof of stake* o PoS.

Desarrollo en Ethereum

Para comprender cómo funcionan los contratos inteligentes además de Ethereum, necesitamos comprender el concepto de gas.

Los contratos inteligentes en Ethereum están codificados en el lenguaje de programación Solidity. Solidity es bastante similar a JavaScript y es Turing completo. Un lenguaje es Turing completo si teóricamente puede resolver cualquier problema siempre que tenga suficiente tiempo y recursos. Es por eso que fue fundamental

integrar un mecanismo de detención dentro del sistema, que detendrá el contrato inteligente cuando sea necesario. Ethereum utiliza la "tarifas de gas" como medidor de recursos. El gas es una unidad que mide la cantidad de esfuerzo computacional que se necesitará para ejecutar ciertas operaciones. En pocas palabras, cada paso de ejecución en el contrato cuesta gas y una vez que el costo del gas excede la tarifa prepaga, la ejecución se cancela.

Todas las operaciones que los usuarios desean ejecutar en ethereum deben proporcionar gas tanto para cubrir sus datos (también conocido como gas intrínseco) como para cubrir todo el cómputo realizado [4].

Entre las principales ventajas con las que cuenta Ethereum se encuentra que, al ser la primera plataforma para contratos inteligentes, aún es la plataforma que cuenta con mayor cantidad de aplicaciones distribuidas corriendo, con 2518 aplicaciones distribuidas en julio del 2019. Para poner este número en contexto, la plataforma que le sigue contaba con 285 en la misma fecha [5]. Además, probablemente también sea la plataforma más cercana a ganar la aceptación general, dado que Microsoft y AWS ofrecen Ethereum *blockchain-as-a-service*. Por estas razones previamente mencionadas, se podría decir que una gran ventaja de Ethereum es la comunidad y soporte disponible. Este no es un aspecto menor, ya que al tratarse de una tecnología nueva, el apoyo de la comunidad y material disponible son de gran importancia.

EOS

EOS se basa en el software EOSIO creado por block.one. Block.one está encabezado por Brendan Blumer y Dan Larimer. Larimer es el creador del *proof-of-stake* delegado y las organizaciones autónomas descentralizadas, también conocidas como DAO.

Si bien Ethereum marcó el camino para las plataformas de contrato inteligentes, aún es una plataforma muy lenta. Debido a su diseño, solo puede administrar 15-20

transacciones por segundo. Una plataforma con una latencia tan alta no puede soportar dApps modernas.

Eos tiene como objetivo apoyar aplicaciones descentralizadas a escala industrial. La forma en hacen es abordando los contratos inteligentes desde un ángulo diferente al de Ethereum. En lugar de ser una supercomputadora descentralizada, Eos planea ser un sistema operativo descentralizado. Los usuarios de Eos poseerán recursos a cambio de su participación. Entonces, si se posee 1/1000 de la participación en EOS, entonces se poseerá 1/1000 de la potencia computacional y los recursos totales en EOS.

EOS delega la validación de la *blockchain* a 21 miembros de un comité de consenso, bajo un mecanismo de consenso de *proof-of-stake* delegado (DPoS).

En DPoS, cualquiera que tenga tokens en una *blockchain* integrada en el software EOS puede seleccionar a los productores de bloques a través de un sistema de votación de aprobación continua. Cualquiera puede participar en la elección del productor de bloques y se les dará la oportunidad de producir bloques proporcionales al total de votos que reciban en relación con todos los demás productores.

De este mecanismo se desprende la principal desventaja de EOS. Dado que EOS fue diseñado para depender solo de 21 productores de bloques para confirmar todas las transacciones, algunos han argumentado que la red está más centralizada que Ethereum y algunas otras criptomonedas. Mientras que EOS depende de la votación de los dueños de *tokens*, la baja participación de votantes podría conducir a una mayor centralización. Además, a algunos les preocupa el hecho de que los usuarios habituales no pueden auditar la red a menos que ejecuten personalmente un nodo completo.

Desarrollo en EOS

Como fue mencionado, EOS utiliza un modelo de gobernanza. Siempre que se cuente con tokens EOS, se tiene derecho a recursos propios como RAM, ancho de

banda de red y ancho de banda de CPU a cambio. Dicho esto, estos recursos son muy escasos y es por eso que solo se puede conservar los tokens EOS, sin usarlos, durante un período de 3 años. A los titulares que no usen sus tokens por ese período se les cancelará su cuenta.

EOS utiliza *WebAssembly* (WASM) para desarrollar contratos inteligentes. Si bien WASM no es un lenguaje de programación, le da a los desarrolladores la posibilidad de codificar en el idioma que deseen, para luego compilarlo a un código de bytes que pueda ejecutarse en un navegador compatible.

Las razones por las que EOS eligió WASM son las siguientes:

- Flexibilidad: los desarrolladores pueden codificar en el idioma de su elección.
- Velocidad y eficiencia: *WebAssembly* se ejecuta a velocidad nativa aprovechando las capacidades comunes de hardware disponibles en una amplia gama de plataformas.
- Abierto y depurable: está diseñado para ser impreso en un formato de texto legible para depurar, probar, experimentar, optimizar, aprender, enseñar y escribir programas a mano.
- Seguro: *WebAssembly* describe un entorno de ejecución de espacio aislado seguro para la memoria que incluso puede implementarse dentro de máquinas virtuales JavaScript existentes.

RSK

Rootstock (RSK) es una plataforma de contratos inteligentes que está conectada a la blockchain de Bitcoin a través de tecnología de *sidechain*. Rootstock nació para ser compatible con las aplicaciones de Ethereum (el modelo web3 / EVM / Solidity) pero usando bitcoin como la criptomoneda subyacente. La idea detrás de la creación de RSK era dar las funcionalidades de contratos inteligentes de blockchain a Bitcoin.

La *blockchain* de Bitcoin tiene varias ventajas. Es relativamente antigua con seguridad comprobada, amplia distribución y popularidad. Además, también tiene

una comunidad saludable con un fuerte poder de cómputo. RSK quiere que sus usuarios disfruten de los beneficios de Bitcoin como una reserva de valor y que, al mismo tiempo, puedan contar con la funcionalidad de contratos inteligentes y una mayor escalabilidad.

Desarrollo en RSK

Los contratos inteligentes en RSK se ejecutan dentro de la máquina virtual RSK (RVM). Las características principales de RVM son las siguientes:

- A nivel de código de operación, la RVM es compatible con la EVM (máquina virtual Ethereum), lo que significa que RSK puede ejecutar contratos de Ethereum.
- Los usuarios podrán ejecutar Ethereum DApps con la seguridad de la blockchain de Bitcoin. Esencialmente disfrutando lo mejor de ambos mundos.
- La comunidad RSK constantemente sugiere mejoras para la performance, las cuales están documentadas en numerosos RSKIPs (propuestas de mejora de RSK).

Este es un enfoque ingenioso que ha sido adoptado por RSK. En lugar de crear su propio lenguaje y obligar a los desarrolladores a trabajar de cierta manera, les permitirán usar el lenguaje de contrato inteligente más popular (Solidity) para crear dApps.

Blockchain a utilizar

Tomando en cuenta las ventajas de cada una de las plataformas vistas anteriormente, el foco que el proyecto tiene en la seguridad, y lo hablado con Alejandro Narancio y Pablo San Nicolás, se decidió utilizar la plataforma RSK. Aunque es cierto que en un principio se utilizará una blockchain privada, también es cierto que, como fue mencionado previamente, esta decisión no es definitiva. Por

ello, en caso de migrar el proyecto a una pública en un futuro, RSK cuenta con varias ventajas.

Adicionalmente RSK es compatible con Ethereum, por lo que si esta decisión cambiara podría fácilmente usarse el mismo contrato inteligente en la red Ethereum sin hacer ningún cambio en el mismo.

RSK es una plataforma de contratos inteligentes asegurada por la red Bitcoin. Al igual que los desarrolladores pueden desarrollar dApps para Ethereum y proyectos similares, RSK les permite crear contratos inteligentes utilizando la seguridad de la red de Bitcoin y al mismo tiempo utilizando las herramientas de Ethereum y el código estándar probado y revisado por la comunidad.

A pesar de lo previamente mencionado, el sistema no utiliza una blockchain privada para funcionar. Esto se debe a que no se pudo contar con una arquitectura *on premise*. Respetando dicho requerimiento, la alternativa a esto hubiera sido utilizar *Amazon Web Services*, aunque esta alternativa también se debió descartar debido a los costos.

Finalmente, con el objetivo de agilizar el proceso de desarrollo, se optó por utilizar una red de prueba. Estas cuentan con la ventaja de que las transacciones no tienen costo.

Decisiones de diseño relacionadas a *blockchain*

Como fue mencionado anteriormente, en el sistema desarrollado se utilizó una *blockchain* como parte central de un mecanismo de verificación de integridad de datos, con el objetivo de detectar modificaciones no autorizadas. Al idear dicho mecanismo, el equipo debió tomar decisiones con respecto a tres aspectos centrales:

1. Qué almacenar en la blockchain
2. En qué momento hacerlo
3. Manejo de concurrencia

Qué almacenar en la blockchain

En cuanto a esta disyuntiva, el enfoque inicial fue de almacenar toda la información en la blockchain, tanto data en formato JSON como archivos. Sin embargo, fue la misma inmutabilidad que caracteriza a las *blockchains* que llevó al equipo a tomar otro camino debido a los siguientes artículos de la ley número 18.331 (ley de protección de datos personales):

- **Artículo 8:** Los datos deberán ser eliminados cuando hayan dejado de ser necesarios o pertinentes a los fines para los cuales hubieren sido recolectados.

- **Artículo 15**: Derecho de rectificación, actualización, inclusión o supresión.- Toda persona física o jurídica tendrá derecho a solicitar la rectificación, actualización, inclusión o supresión de los datos personales que le corresponda incluidos en una base de datos, al constatarse error o falsedad o exclusión en la información de la que es titular.

Estos artículos dictan que los datos deben ser eliminados en determinadas circunstancias, lo cual no sería posible en caso de ser almacenados en una blockchain. Esto obligó al equipo a idear una solución que permita almacenar una representación de la data que no presente problemas al permanecer en la blockchain.
Finalmente, se decidió no almacenar la data en su totalidad, y en su lugar se adoptó la táctica de verificar integridad de mensaje, la cual se basa en el empleo de tácticas como *checksums* o *hashes* para verificar la integridad de datos. Puntualmente, se optó por almacenar un *hash* de la información en la blockchain, mientras se almacena la información no hasheada en una base de datos tradicional. Almacenar hashes en la *blockchain* no presenta un problema dado que estos son irreversibles. Además, al ser libres de colisiones, en caso de que se realice una modificación no autorizada en la base de datos tradicional, el *hash* almacenado en la blockchain

dejaría de coincidir, evidenciando dicha modificación. De esta forma se logra asegurar la integridad de la información almacenada en la base de datos tradicional. Finalmente, sí es posible remover la data de las bases de datos tradicionales en caso de que un usuario así lo solicite.

En qué momento almacenar información en la blockchain

En cuanto a la segunda disyuntiva, la primer alternativa que se evaluó fue la de utilizar *triggers* de bases de datos. Estos desencadenan la ejecución de código cada vez que se crea, actualiza, o elimina un registro. Sin embargo, este enfoque haría que todas las modificaciones, ya sean legítimas o no, se propaguen a la *blockchain*, haciendo que deje de cumplir su propósito. Por esta razón, se optó por actualizarla "manualmente": cada vez que se crea, actualiza, o elimina un registro, se hace realizando un llamado a un microservicio del *back-end*. Este, además de actualizar la base de datos, realiza un llamado al servicio encargado de interactuar con el contrato inteligente alojado en la *blockchain* con el fin de propagar el cambio a la información almacenada en la *blockchain*.

Manejo de concurrencia

Finalmente, en cuanto a la tercer disyuntiva, el manejo de concurrencia, se debió investigar el funcionamiento de la concurrencia para los contratos inteligentes. Tras investigar, se encontró que el orden de las transacciones incluidas en un bloque no está determinado por el momento de ejecución de las transacciones. Los resultados pueden variar dependiendo de cómo se ordenan las mismas. Además, también existe el conflicto entre transacciones. Dos transacciones están en conflicto si acceden a la misma ubicación de memoria, y al menos una de ellas es una escritura. Por cada par de transacciones en conflicto, una es descartada, y la otra es aceptada. Este conocimiento obligó al equipo a implementar un mecanismo de manejo de concurrencia dentro de nuestro sistema, con el objetivo de evitar conflictos en las transacciones asociadas a nuestro contrato inteligente.

A continuación se puede observar un diagrama de secuencia de cómo se realiza la persistencia en la blockchain del *hash* de la información de un estudiante, utilizando un mecanismo de manejo de concurrencia.

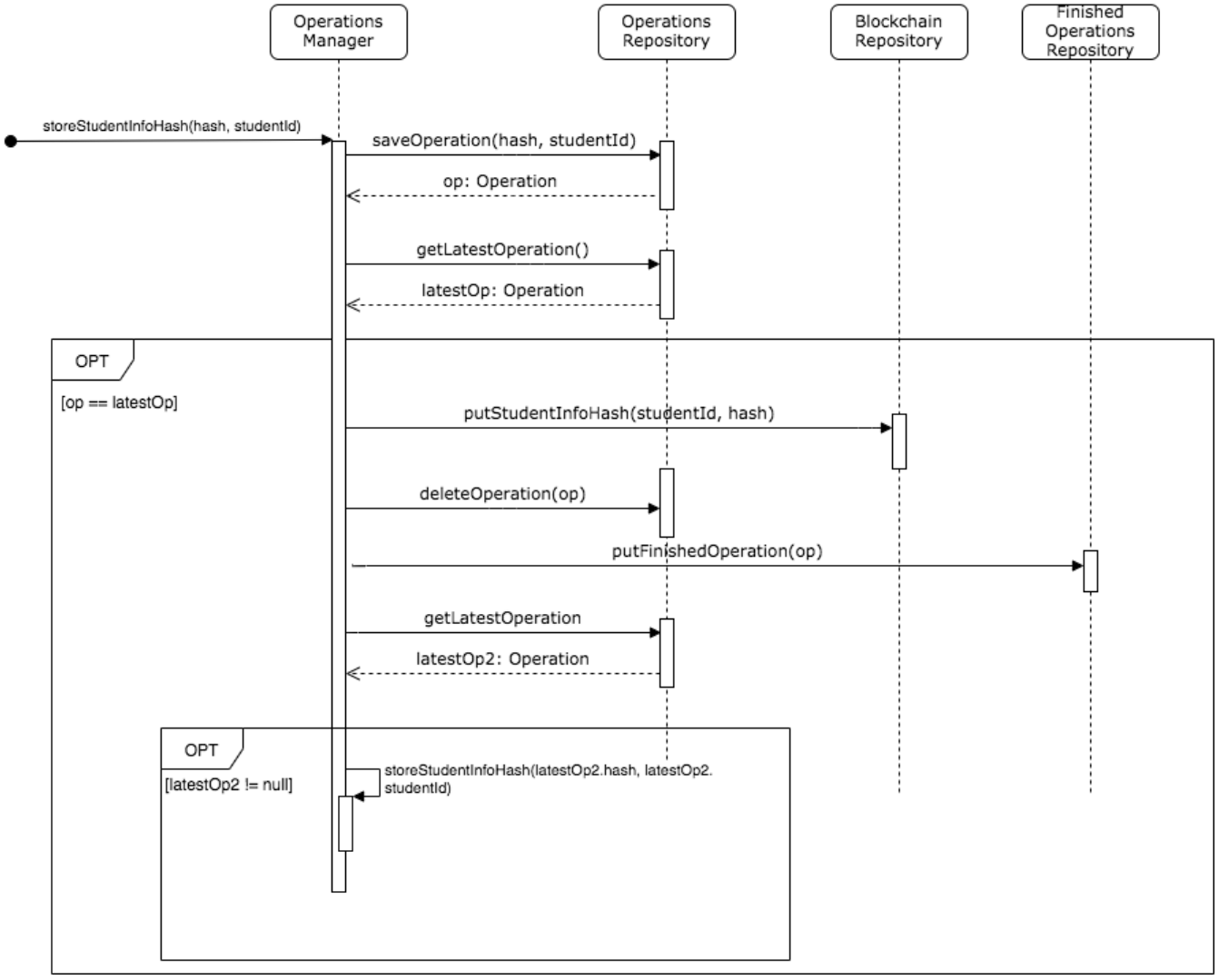


Imagen 71: Diagrama de secuencia de la escritura en *blockchain*

Cabe aclarar que la decisión de utilizar una blockchain para preservar la integridad afecta negativamente a la performance del sistema, ya que las actualizaciones y consultas a la misma no tienen la misma velocidad que aquellas realizadas a las bases de datos tradicionales. Se realizó un *trade-off* entre seguridad y performance, debido a que durante el relevamiento de requerimientos, los distintos potenciales usuarios dieron mayor prioridad a la seguridad que a la performance.

Lecciones aprendidas

A continuación, se detallarán las principales lecciones aprendidas a raíz de la realización de un proyecto que utiliza blockchain como tecnología central del mecanismo de verificación de integridad de datos.

Escalabilidad

Aunque este no fue un aspecto central del proyecto, durante las etapas de investigación se aprendió que en varias plataformas de *blockchain* existen problemas de escalabilidad. A modo de ejemplo, Bitcoin, en su forma actual, puede manejar alrededor de 7 transacciones por segundo (TPS). Ethereum, en el que las empresas a menudo están más interesadas ya que puede ejecutar contratos inteligentes, puede soportar aproximadamente 20 TPS. Este aspecto debe ser tenido en cuenta al momento de decidir qué tipo de *blockchain* utilizar, o si *blockchain* es una solución adecuada.

Costo de transacciones

Al momento de decidir qué tipo de *blockchain* utilizar para resolver cierta problemática, si una pública o una privada, se debe tener en cuenta que en varias *blockchains* las transacciones tienen un costo asociado. Por esta razón, se debe considerar quién cubriría el costo, y tener en cuenta una aproximación de la cantidad de usuarios esperados y el costo por transacción, con el fin de poder llegar a un estimativo del costo total.

Inmutabilidad de datos

Al momento de decidir qué almacenar dentro de la *blockchain* se debe tener en cuenta que aquello almacenado en una *blockchain* es inmutable. Según el tipo de datos que se esté tratando, pueden existir leyes (como fue el caso de el proyecto realizado) que establezcan que los usuarios pueden solicitar la eliminación de los

datos almacenados. Fue por esta razón que el equipo decidió almacenar en la *blockchain* una representación irreversible de los datos, llamada *hash*.

Trade-off con la performance

Al momento de decidir si la utilización de una *blockchain* es adecuada para cierto problema, se debe tener en cuenta que esta daña considerablemente la performance del sistema. En el caso del proyecto desarrollado, la seguridad tuvo una prioridad mayor que la performance, pero en caso de que existan requerimientos prioritarios en cuanto a la performance, es posible que las tecnologías de *blockchain* no sean la decisión adecuada.

Manejo de concurrencia

Como fue visto anteriormente, la concurrencia en cuanto a la actualización de datos debe ser un aspecto a tener en cuenta al momento de utilizar una *blockchain*. Previo a implementar una solución, el funcionamiento de la concurrencia dentro de la tecnología de *blockchain* electa debe ser investigado, pues en esta pueden existir distintos mecanismos de resolución de conflicto entre transacciones.

Blockchain como mecanismo de control de integridad

El proyecto realizado produjo resultados satisfactorios en cuanto a la utilización de un mecanismo basado en tecnología de *blockchain* de control de integridad de los datos. Uno de los aspectos más positivos de dicho mecanismo es que este se relativamente genérico, por lo que se puede aplicar a cualquier dominio. Existe un estándar llamado *blockcerts* basado en un concepto similar al aplicado en el proyecto. En este, se almacena el *hash* de los certificados en una *blockchain*, el cual es posteriormente utilizado para verificar la validez de dichos certificados. El equipo cree que, en caso de que la integridad de la data juegue un rol clave en los requerimientos de un proyecto, *blockchain* puede ser una solución adecuada, aunque se deben tener en cuenta los otros aspectos mencionados en este documento.

12.2. Reglamento del equipo

1. A partir del comienzo el proyecto hasta la fecha de entrega se mantendrán reuniones de 1 hora de duración todas las semanas entre los miembros del equipo y el tutor. Sólo se suspenderán en caso que no pueda asistir el tutor o la totalidad de los integrantes del equipo.
2. Al menos una vez por semana durante toda la duración del proyecto habrá reuniones entre los integrantes del equipo para una puesta a punto de la situación del proyecto, de cada integrante, planificación de los siguientes pasos, y discusión de problemas que hayan surgido.
3. Los días y horarios de las reuniones pueden ir cambiando a lo largo del año estando todas las partes de acuerdo.
4. En caso de no poder asistir a una reunión se debe notificar con anticipación no menor a tres horas antes del comienzo de la misma.
5. En caso de reuniones con interesados alcanza con que asista uno de los integrantes del equipo. Se prioriza el tiempo de la persona que nos recibe.
6. Llevar registro de reuniones con los distintos interesados. Fecha y hora, grabación en audio y tomar una fotografía en lo posible para registrar evidencia.
7. Cada integrante del equipo se compromete a dedicar entre 15 y 20 horas semanales al proyecto en promedio.
8. Se contemplarán 30 días de vacaciones para cada uno de los integrantes a lo largo de todo el proyecto sin necesidad de recuperar las horas no trabajadas.

12.3. Descripción del Dual Track Agile

El proceso de dual track agile es una forma de organizar y planificar el trabajo de un equipo ágil que permite solucionar los siguientes problemas:

1. La participación tardía del equipo de desarrollo en el proceso de planificación.
2. Las historias incompletas que llegan al *sprint planning*.

Este proceso se vale de los siguientes principios:

1. El *feedback loop*

Comenzando por el *delivery team*. Su función es entregar una solución para los problemas del cliente. Está conformado por los desarrolladores, el *tech lead* (líder técnico), y el *scrum master*, cuya función es asegurarse de que el proceso se lleve a cabo con la menor cantidad de contratiempos posibles.

Cada incremento de valor generado por el *delivery team* entregado al cliente es usado por el mismo. El cliente luego responde con feedback en la forma de ideas: reportes de bugs o peticiones de *features.* Estas ideas son procesadas por el *delivery team* para producir un nuevo incremento de valor.

Mientras más rápido se entregue el valor al cliente, más rápido se obtiene *feedback*, lo cual incrementa el valor del software para el cliente.

2. Relación entre la realidad y el *feedback loop*

El *feedback loop* descrito anteriormente resalta la importancia de obtener *feedback* de forma veloz. Desafortunadamente, la realidad de construir software es un poco más complicada.

En primer lugar, algunas ideas son imposibles de implementar (no son feasibles). Dichas ideas son descartadas por el *delivery team*.

En segundo lugar, es inevitable que el tiempo no sea suficiente para implementar todas las ideas, por lo que estas se acumulan y se convierten en un *backlog*.

Con el tiempo, dicho backlog puede volverse más y más grande, hasta que el *delivery team* sea incapaz de responder rápidamente a los clientes. Una forma de lidiar con este problema es con un ***product owner*.**

3. El rol del *product owner*

El *product owner* ayuda a mantener el *backlog* bajo control. Este no es un miembro del *delivery team*, sino que forma parte de un equipo en paralelo que, junto con el *delivery team*, son los encargados de crear valor para los clientes.

El *product owner* es el responsable de ponderar si cada idea posee valor. Si la respuesta es “no”, dicha idea se descarta. Si la respuesta es “sí”, este toma la idea y crea una historia de usuario - una declaración clara del problema a resolver - y la prioriza en un *product backlog*. Cuando el *delivery team* esté listo para más trabajo, tomarán la idea de mayor prioridad en el *product backlog*, y trabajarán para convertirla en valor.

4. Mejorar el manejo de ideas

Hasta ahora, el mayor problema con este modelo es que asume que se crea software puramente de forma reactiva. Es necesario asegurarse de que se está progresando hacia un propósito que determina qué debería ser el producto. Este propósito es descrito por la **visión** y los **objetivos** del producto.

La visión y los objetivos se convierten en fuentes de ideas, y también ayudan a determinar el valor de las ideas provenientes de los clientes. Si alguna de ellas no encajan en la visión y objetivos, puede ser descartada.

5. El rol del tech lead

El *tech lead* debe trabajar junto con el *product owner*, ayudándolo a identificar cuáles ideas no son feasibles. Junto con este, el *teach lead* forma el *discovery team***.**

De esta forma, cuando el *delivery team* toma una idea del *product backlog*, se sabe con certeza que la misma es feasible y que agrega valor al software.

6. Acelerar la obtención de *feedback*

Para obtener *feedback* de forma más rápida, se debe identificar cuál es la parte más lenta del *loop*.

En la gran mayoría de los casos, construir el software es lo que lleva más tiempo, y acelerar esta parte del proceso no es una tarea fácil. En un principio, no parece posible obtener *feedback* de un software que aún no está construido. Sin embargo, sí existe una manera.

7. *Feedback* más veloz a través de prototipado

En lugar de esperar hasta que se cuente con el software para obtener el *feedback*, es recomendable hacer uso de prototipos**.**

Entonces, es tarea del *discovery team* crear prototipos del valor que se pueda entregar a los clientes. Estos prototipos pueden ser *web page wireframes*, prototipos hechos a mano, etc. El *product owner* tiene la tarea de mostrar estos prototipos a los clientes, y obtener la respuesta a la pregunta “si construyéramos esto, sería usado por el cliente?”.

En general, un diseñador se une al *discovery team* para asegurarse de que el producto sea usable. Si el *feedback* muestra que la idea es usable y valiosa, se puede adjuntar el prototipo a la historia de usuario, y agregar la historia al *product backlog*, de modo que el *delivery team* tenga una mejor guía de cómo construirla. De lo contrario, la idea es descartada.

8. Los dos ciclos

Se puede apreciar que se cuenta con dos ciclos de *feedback*:

1. El ***discovery cycle*****:** valida el valor y la factibilidad de una idea, creando prototipos en el camino.
2. El ***delivery cycle***: entrega el valor de una idea ya validada.

El *delivery team* y el *discovery team* deben trabajar juntos, formando un grupo cohesivo. El *discovery cycle* puede ser mucho más corto que el *delivery cycle*, ya que prototipar es mucho más rápido que construir software funcional. [6]

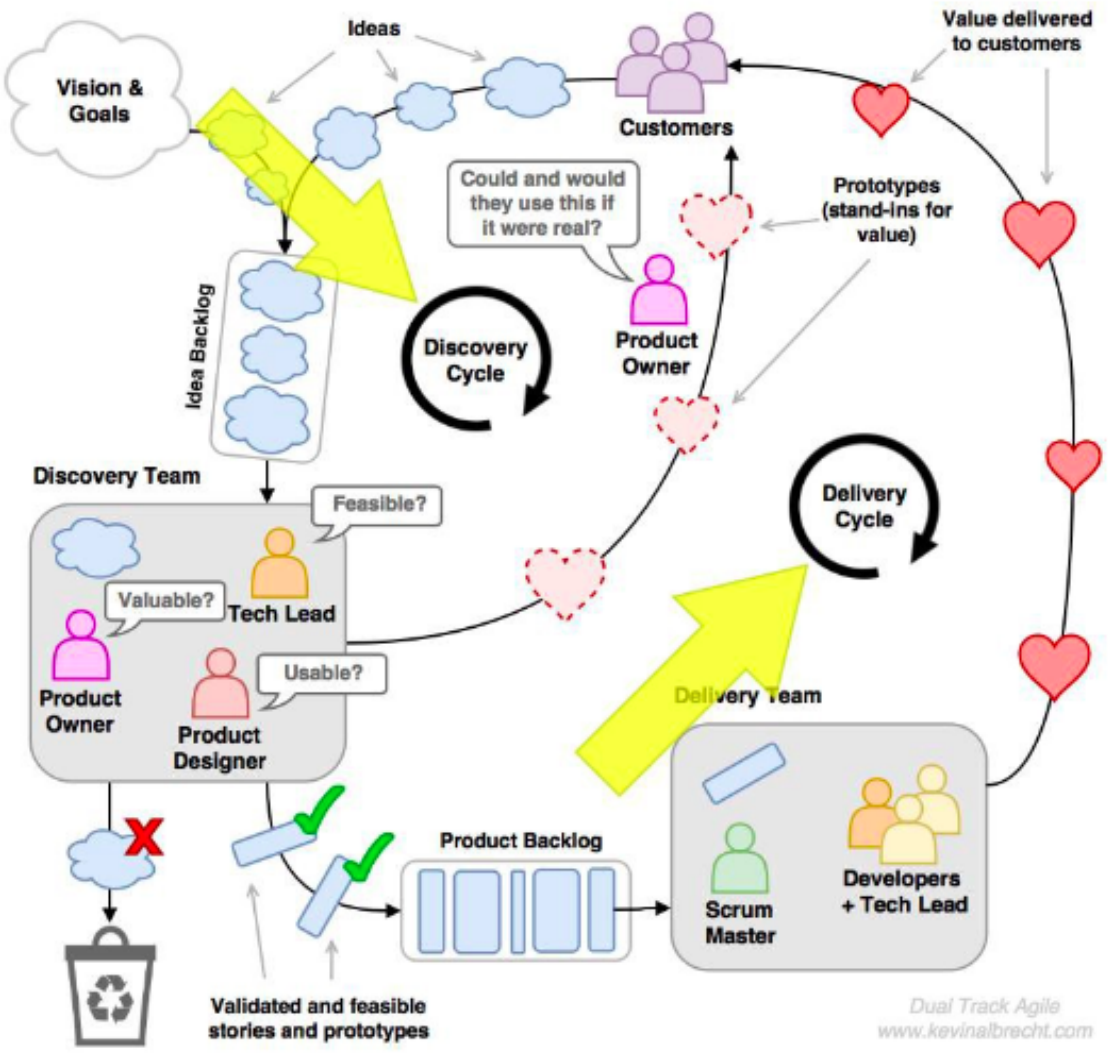


Imagen 72: ilustración de *Dual Track Agile*

12.4. Marco de trabajo Scrum

Scrum es un marco de trabajo por el cual las personas pueden abordar problemas complejos adaptativos, a la vez que entregar productos del máximo valor posible productiva y creativamente.

El marco de trabajo Scrum consiste en los equipos Scrum y sus roles, eventos, artefactos y las reglas que los relacionan y rigen sus interacciones.

El equipo Scrum

El equipo Scrum consiste en un dueño de producto (*product owner*), el equipo de desarrollo (*development team*) y un *Scrum master*. Estos son autoorganizados y multifuncionales, y entregan productos de forma iterativa e incremental, maximizando las oportunidades de obtener retroalimentación.

El dueño del producto (*product owner*)

El dueño de producto es el responsable de maximizar el valor del producto resultante del trabajo del equipo de desarrollo. Cómo se lleva a cabo esto puede variar ampliamente entre distintas organizaciones, equipos Scrum e individuos.

El dueño de producto es la única persona responsable de gestionar la lista del producto (*product backlog*). La gestión de la lista del producto incluye:

- Expresar claramente los elementos de la lista del producto.
- Ordenar los elementos en la lista del producto para alcanzar los objetivos y misiones de la mejor manera posible
- Optimizar el valor del trabajo que el equipo de desarrollo realiza
- Asegurar que la lista del producto es visible, transparente y clara para todos y que muestra aquello en lo que el equipo trabajará a continuación
- Asegurar que el equipo de desarrollo entienda los elementos de la lista del producto al nivel necesario.

El equipo de desarrollo (*development team*)

El equipo de desarrollo consiste en los profesionales que realizan el trabajo de entregar un incremento de producto terminado que potencialmente se pueda poner en producción al final de cada sprint.

Los equipos de desarrollo tienen las siguientes características:

- Son autoorganizados. Nadie (ni siquiera el Scrum Master) indica al equipo de desarrollo cómo convertir elementos de la lista del producto en incrementos de funcionalidad potencialmente desplegables.
- Los equipos de desarrollo son multifuncionales, esto es, como equipo cuentan con todas las habilidades necesarias para crear un incremento de producto.
- Scrum no reconoce títulos para los miembros de un equipo de desarrollo independientemente del trabajo que realice cada persona.
- Scrum no reconoce subequipos en los equipos de desarrollo, no importan los dominios que requieran tenerse en cuenta, como pruebas, arquitectura, operaciones o análisis de negocio.
- Los miembros individuales del equipo de desarrollo pueden tener habilidades especializadas y áreas en las que estén más enfocados, pero la responsabilidad recae en el equipo de desarrollo como un todo.

El Scrum master

El Scrum master es responsable de promover y apoyar Scrum como se define en la guía de Scrum. Los Scrum masters hacen esto ayudando a todos a entender la teoría, prácticas, reglas y valores de Scrum. El Scrum master es un líder que está al servicio del equipo Scrum..

El Scrum master da servicio al dueño de producto de varias formas, incluyendo:

- Asegurar que los objetivos, el alcance y el dominio del producto sean entendidos por todos en el equipo Scrum de la mejor manera posible
- Encontrar técnicas para gestionar la lista de producto de manera efectiva
- Ayudar al equipo Scrum a entender la necesidad de contar con elementos de lista de producto claros y concisos
- Entender la planificación del producto en un entorno empírico
- Asegurar que el dueño de producto conozca cómo ordenar la lista de producto para maximizar el valor
- Entender y practicar la agilidad
- Facilitar los eventos de Scrum según se requiera o necesite.

El Scrum Master da servicio al equipo de desarrollo de varias formas, incluyendo:

- Guiar al equipo de desarrollo en ser autoorganizado y multifuncional
- Ayudar al equipo de desarrollo a crear productos de alto valor
- Eliminar impedimentos para el progreso del equipo de desarrollo
- Facilitar los eventos de Scrum según se requiera o necesite
- Guiar al equipo de desarrollo en entornos organizacionales en los que Scrum aún no haya sido adoptado y entendido por completo.

Eventos de Scrum

En Scrum existen eventos predefinidos con el fin de crear regularidad y minimizar la necesidad de reuniones no definidas en Scrum. Todos los eventos son bloques de tiempo (*time-boxes*), de tal modo que todos tienen una duración máxima. Una vez que comienza un sprint, su duración es fija y no puede acortarse o alargarse. Los demás eventos pueden terminar siempre que se alcance el objetivo del evento, asegurando que se emplee una cantidad apropiada de tiempo sin permitir desperdicio en el proceso.

El Sprint

El corazón de Scrum es el sprint. Es un bloque de tiempo (*time-box*) de un mes o menos durante el cual se crea un incremento de producto “terminado” utilizable y potencialmente desplegable. Los sprints contienen y consisten en la planificación del sprint (*sprint planning*), los scrums diarios (*daily scrums*), el trabajo de desarrollo, la revisión del sprint (*sprint review*), y la retrospectiva del sprint (*sprint retrospective*).

Durante el sprint:

- No se realizan cambios que puedan afectar al objetivo del sprint (*sprint goal*)
- Los objetivos de calidad no disminuyen
- El alcance puede clarificarse y renegociarse entre el dueño de producto y el equipo de desarrollo a medida que se va aprendiendo más.

Planificación del sprint (*sprint planning*)

El trabajo a realizar durante el *sprint* se planifica en la planificación del *sprint*. Este plan se crea mediante el trabajo colaborativo del equipo Scrum completo. Para sprints más cortos el evento es usualmente más corto. El Scrum master se asegura de que el evento se lleve a cabo y que los asistentes entiendan su propósito.

La Planificación de Sprint responde a las siguientes preguntas:

- **¿Qué puede entregarse en el Incremento resultante del Sprint que comienza?**

  El equipo de desarrollo trabaja para proyectar la funcionalidad que se desarrollará durante el sprint. El dueño de producto discute el objetivo que el sprint debería lograr y los elementos de la lista de producto que, si se completan en el sprint, lograrían el objetivo del Sprint.

  Durante la planificación del sprint, el equipo Scrum también define un objetivo del sprint (*sprint goal*). El objetivo del sprint debería lograrse durante el sprint a través de la implementación de la lista de producto y proporciona una guía

al equipo de desarrollo de por qué se está construyendo el incremento.

- **¿Cómo se conseguirá hacer el trabajo necesario para entregar el Incremento?**
  Una vez que se ha establecido el objetivo y seleccionado los elementos de la lista de producto para el sprint, el equipo de desarrollo decide cómo construirá esta funcionalidad para formar un incremento de producto "terminado" durante el Sprint. Los elementos de la lista de producto seleccionados para este sprint, más el plan para terminarlos, recibe el nombre de lista de pendientes del sprint (*sprint backlog*).

Scrum diario (daily scrum)

El Scrum diario es una reunión con un bloque de tiempo de 15 minutos para el equipo de desarrollo. El Scrum diario se lleva a cabo cada día del sprint. En él, el equipo de desarrollo planifica el trabajo para las siguientes 24 horas. El equipo de desarrollo usa el Scrum diario para evaluar el progreso hacia el objetivo del sprint. Algunos equipos de desarrollo usarán preguntas, algunos se basarán más en discusiones. Aquí hay un ejemplo de lo que podría usarse:

- ¿Qué hice ayer que ayudó al equipo de desarrollo a lograr el objetivo del sprint?
- ¿Qué haré hoy para ayudar al equipo de desarrollo a lograr el objetivo del sprint?
- ¿Veo algún impedimento que evite que el equipo de desarrollo o yo logremos el objetivo del sprint?

Retrospectiva de sprint (*sprint retrospective*)

La retrospectiva de sprint es una oportunidad para el equipo scrum de inspeccionarse a sí mismo y de crear un plan de mejoras que sean abordadas durante el siguiente sprint. La retrospectiva de sprint tiene lugar después de la revisión de sprint y antes de la siguiente planificación de sprint. El scrum master se

asegura de que el evento se lleve a cabo y que los asistentes entiendan su propósito. El Scrum master participa en la reunión como un miembro del equipo ya que la responsabilidad del proceso Scrum recae sobre él. El propósito de la retrospectiva de sprint es:

- Inspeccionar cómo fue el último sprint en cuanto a personas, relaciones, procesos y herramientas
- Identificar y ordenar los elementos más importantes que salieron bien y las posibles mejoras
- Crear un plan para implementar las mejoras a la forma en la que el equipo Scrum desempeña su trabajo.

Artefactos de Scrum

Lista de producto (*product backlog*)

La lista de producto es una lista ordenada de todo lo que se conoce que es necesario en el producto. Es la única fuente de requisitos para cualquier cambio a realizarse en el producto. El dueño de producto es el responsable de la lista de producto, incluyendo su contenido, disponibilidad y ordenación.

La lista de producto es dinámica; cambia constantemente para identificar lo que el producto necesita para ser adecuado, competitivo y útil. La lista de producto enumera todas las características, funcionalidades, requisitos, mejoras y correcciones que constituyen cambios a realizarse sobre el producto para entregas futuras. Los elementos de la lista de producto tienen como atributos la descripción, el orden, la estimación y el valor.

Lista de pendientes del sprint (*sprint backlog*)

La lista de pendientes del sprint es el conjunto de elementos de la lista de producto seleccionados para el sprint. La lista de pendientes del sprint es una predicción hecha por el equipo de desarrollo acerca de qué funcionalidad formará parte del

próximo incremento y del trabajo necesario para entregar esa funcionalidad en un incremento “terminado”.

Definición de terminado (*definition of done*)

Los miembros del equipo deben tener un entendimiento compartido de lo que significa que el trabajo esté completado para asegurar la transparencia. Esta es la definición de terminado para el equipo Scrum y se utiliza para evaluar cuándo se ha completado el trabajo sobre el incremento de producto. Esta misma definición guía al equipo de desarrollo en saber cuántos elementos de la lista de producto puede seleccionar durante la planificación del Sprint.

12.5. Value proposition canvas

Perfil de clientes

1. Empleadores

Tarea	Importancia (1 - 5)
Tareas funcionales	
Leer CVs	5
Filtrar según calificaciones	5
Verificar calificaciones	4
Contactar consultoras	3
Verificar referencias	5
Publicar ofertas en páginas web	3
Tareas sociales	
Seleccionar candidatos competentes	5
Tareas personales/emocionales	
Sentirse seguro al seleccionar candidatos	4
Tener seguridad de las calificaciones de los candidatos a contratar	4
Tareas de apoyo	
Navegar páginas web como LinkedIn, Facebook, etc.	4
Navegar portales de universidades	5

Tabla 16: tareas de empleadores

Dolor	Severidad (1-5)
Resultados indeseables, problemas, y características	
Funcional: dificultad para contactar instituciones educativas	5

Social: hacer ver mal la empresa contratando gente no calificada.	4
Emocional: sentirse inseguro a la hora de contratar personal	3
Auxiliar: dificultad navegando redes para encontrar candidatos	3
Obstáculos	
Candidatos que demoran en enviar escolaridades	4
Candidatos que demoran en enviar sus diplomas	4
Riesgos (potenciales resultados desfavorables)	
Candidatos que mienten sobre sus calificaciones	5
Contratar candidatos no calificados	5

Tabla 17: dolores de empleadores

Ganancia	Relevancia (1 - 5)
Ganancias requeridas	
Poder verificar escolaridades de estudiantes	5
Ganancias esperadas	
Confiabilidad del sistema	4
Usabilidad del sistema	5
Seguridad del sistema	5
Ganancias deseadas	
Poder ver imágenes del diploma en la red	5
Ganancias inesperadas	
Publicar ofertas de trabajo	2

Tabla 18: ganancias de empleadores

2. Egresados

Tarea	Importancia (1 - 5)
Tareas funcionales	
Crear CV	5
Fotocopiar Diplomas	4
Fotocopiar escolaridad	4
Obtener sello de autenticidad de documento	5
Publicar CV	5
Aplicar a trabajos	3
Crear carta firmada dando permiso a la institución de confirmar sus calificaciones	5
Tareas sociales	
Presentarse como candidato competente	4
Diferenciarse del resto de los candidatos, viéndose más al tanto con nuevas tecnologías	4
Tareas Personales/emocionales	
Sentirse seguro y a gusto del CV que publica	4
Sentirse diferente del resto de los candidatos	3
Tareas de apoyo	
Ir hasta la institución educativa a obtener sello	5
Navegar páginas web como LinkedIn, Facebook, etc.	3
Ir a fotocopiadora	4
Escanear títulos académicos	4
Imprimir y firmar carta de permiso	5

Tabla 19: tareas de egresados

Dolor	Severidad (1-5)
Resultados indeseables, problemas, y características	
Funcional: fotocopiadoras que no funcionan	3
Emocional: sentir que su CV es simplemente uno más del montón	3
Emocional: frustración al tener que crear e imprimir la carta de permiso varias veces	4
Auxiliar: es molesto tener que ir hasta la institución educativa	5
Auxiliar: es molesto tener que ir hasta la fotocopiadora	5
Obstáculos	
No poder obtener sello de autenticidad debido a que la institución educativa está cerrada	5
No encontrar fotocopiadoras cerca para fotocopiar diploma o escolaridad	3
Riesgos (potenciales resultados desfavorables)	
No obtener un trabajo debido al proceso lento de verificación de calificaciones	2

Tabla 20: dolores de egresados

Ganancia	Relevancia (1 - 5)
Ganancias requeridas	
Poder ofrecer forma de verificar calificaciones sin necesidad de moverse de su casa	5
Ganancias esperadas	
Confiabilidad del sistema	4
Usabilidad del sistema	5
Seguridad del sistema	5

Poder mostrar imágenes de las calificaciones u otros archivos	5
Ganancias inesperadas	
Recibir ofertas de trabajo	2

Tabla 21: ganancias de egresados

3. Instituciones académicas

Tarea	Importancia (1 - 5)
Tareas funcionales	
Recibir llamados de empleadores	5
Verificar calificaciones en el sistema	5
Confirmar calificaciones a empleadores	5
Tareas sociales	
Tener status de institución que se preocupa por sus alumnos	5
Diferenciarse del resto de las instituciones académicas, presentándose más al tanto de nuevas tecnologías	2
Tareas Personales/emocionales	
Sentirse seguro en brindarle herramientas a los egresados para que tengan éxito en su búsqueda de trabajo	5
Tareas de Apoyo	
Ingresar en el sistema para verificar las calificaciones	3
Buscar calificaciones en el sistema	3
Contestar llamadas	4

Tabla 22: tareas de instituciones académicas

Dolor	Severidad (1-5)
Resultados indeseables, problemas, y características	

Funcional: tener que corroborar varias veces las calificaciones de un mismo egresado.	5
Emocional: funcionarios que deben desmentir las calificaciones de estudiantes.	3
Riesgos (potenciales resultados desfavorables)	
Personas que cometen fraude utilizando diplomas de la institución, dañando la reputación de la misma	5

Tabla 23: dolores de instituciones académicas

Ganancia	**Relevancia (1 - 5)**
Ganancias requeridas	
Poder verificar las calificaciones de los estudiantes	5
Disponer de toda la información asociada al estudiante	4
Poder verificar la identidad del estudiante	5
Ganancias esperadas	
Confiabilidad del sistema	4
Usabilidad del sistema	5
Seguridad del sistema	5
Poder determinar qué datos los estudiantes deben ingresar	5

Tabla 24: ganancias de instituciones académicas

Mapa de valor

Productos y servicios

- Web App:
    - Crear y editar perfil de egresado.

- Crear y editar perfil de institución educativa, ingresando títulos y requerimientos para estudiantes. Los requerimientos representan datos que los egresados deben ingresar para verificar su identidad.
- Subir calificación de egresado.
- Agregar archivos a calificación.
- Solicitar verificación de calificación por parte de institución educativa.
- Verificar calificación por parte de institución educativa.
- Compartir Perfil con quien desee.

Aliviadores de dolores

- Elimina la necesidad de los empleadores de contactar instituciones educativas, ya que las calificaciones ya están verificadas.
- Permite que los empleadores se sientan seguros al contratar personal con calificaciones verificadas
- Elimina la necesidad de esperar a que los candidatos envíen sus documentos apostillados
- Elimina la posibilidad de los candidatos de mentir acerca de sus calificaciones, por lo que elimina el riesgo de contratar candidatos que mienten sobre sus calificaciones
- Elimina el riesgo de contratar candidatos no calificados
- Permite a los egresados subir archivos visibles por posibles empleadores, eliminando la necesidad de fotocopiar archivos.
- Permite a los egresados subir una carta dándole permiso a las instituciones de verificar las calificaciones a través del sistema, debiendo hacerlo una única vez.

- Elimina la necesidad de los egresados de ir hasta la institución educativa a solicitar documentación apostillada
- Elimina la necesidad de los egresados de ir hasta una fotocopiadora
- Elimina el obstáculo de que la institución académica se encuentre cerrada
- Elimina la necesidad de las instituciones académicas de corroborar la misma calificación varias veces
- Elimina la necesidad de las instituciones académicas de corroborar calificaciones con potenciales empleadores
- Disminuye la probabilidad de que los estudiantes realicen fraude, ya que un integrante de las instituciones académicas debe confirmar las calificaciones

Creadores de ganancias

- Permite a los empleadores ver calificaciones verificadas
- Sistema confiable y seguro.
- Permite a las instituciones académicas verificar las calificaciones de sus estudiantes
- Permite a los egresados subir archivos relacionados a sus calificaciones
- Permite a los empleadores ver archivos relacionados a las calificaciones de los aplicantes
- Permite a las instituciones académicas definir un conjunto de requerimientos que los estudiantes deben ingresar para solicitar la verificación, tales como números de estudiante, o archivos, como una carta de consentimiento firmada.
- Permite a las instituciones académicas visualizar toda la información necesaria del estudiante

12.6. Preparación de entrevistas utilizando Design Thinking

Para preparar las entrevistas a los distintos usuarios potenciales, se utilizará la siguiente técnica de Design Thinking:

Preparación de una entrevista

Aunque es importante dejar espacio para conversaciones espontáneas guiadas por los usuarios, siempre se deben preparar las entrevistas. Algunas formas de preparación son:

- **Brainstorm de preguntas:** anotar tantas preguntas potenciales como sea posible. Construir sobre las ideas de los otros.
- **Identificar y ordenar temas:** identificar los temas o áreas dentro de los que caen la mayor cantidad de preguntas. Luego, determinar el orden de las preguntas que permitirá que la conversación fluya de mejor manera.
- **Refinar preguntas:** una vez agrupadas las preguntas, es posible que hayan algunas que sean redundantes o estén fuera de lugar. Estas deben ser eliminadas. Además, se debe asegurar de incluir preguntas del tipo “Por qué?”, o del tipo “Cuéntame de la última vez que …”, o preguntas centradas en cómo el usuario se siente. Es importante recordar que las preguntas más abiertas permiten que los entrevistados cuenten historias, y estas pueden llevar a una mayor *insight* para diseñar soluciones.

Entrevistar para la empatía

- **Preguntar *por qué***, aunque se piense que ya se sabe la respuesta.
- **Nunca decir “usualmente” al hacer una pregunta**, en lugar de esto, preguntar acerca de una ocurrencia específica: “Cuéntame de la última vez que ...”
- **Fomentar historias**, pues estas revelan cómo los usuarios se sienten acerca del mundo.

- **Buscar inconsistencias**. Lo que los usuarios dicen y lo que hacen puede ser diferente. Estas inconsistencias generalmente esconden *insights* interesantes
- **Prestar atención a elementos no verbales**, como lenguaje corporal y emociones.
- **No tener miedo al silencio**. Cuando se permite el silencio, se le da tiempo al usuario a reflexionar acerca de sus respuestas, lo cual puede llevar a respuestas más profundas.
- **Preguntar de forma neutral sin sugerir respuestas**.

Entrevista a empleador

Brainstorm:

1. Cuándo mira CVs con el objetivo de realizar una nueva contratación, verifica las calificaciones de los aplicantes? Cómo?
2. Alguna vez le pasó que un aplicante no tenía la calificación que decía tener?
3. Ve algún defecto en el proceso de verificación de calificaciones?
4. Cree que hay alguna posibilidad de optimizar alguna parte de este proceso?
5. En promedio, cuánto le lleva verificar la calificación de un aplicante?
6. Le ha pasado que no se ha podido contactar con la institución educativa que emitió el certificado que el aplicante dice tener? Qué ha hecho en este caso?
7. Cuál es la parte que le resulta más frustrante al intentar validar los conocimientos de un candidato?
8. En dónde suele buscar aplicantes y dónde publica ofertas de empleo?
9. Qué es lo primero que mira en el CV de un aplicante?
10. Busca corroborar de alguna forma lo que está en el CV?

Identificar y Ordenar temas

Captación de candidatos:

1. En dónde suele buscar aplicantes y dónde publica ofertas de empleo?
2. Qué es lo primero que mira en el CV de un aplicante?

Verificación de calificaciones:

3. Cuándo mira CVs con el objetivo de realizar una nueva contratación, verifica las calificaciones de los aplicantes? Cómo?
4. Alguna vez le pasó que un aplicante no tenía la calificación que decía tener?
5. En promedio, cuánto le lleva verificar la calificación de un aplicante?
6. Le ha pasado que no se ha podido contactar con la institución educativa que emitió el certificado que el aplicante dice tener? Qué ha hecho en este caso?
7. Busca corroborar de alguna forma lo que está en el CV?

Opiniones:

8. Cree que hay alguna posibilidad de optimizar alguna parte de este proceso?
9. Ve algún defecto en el proceso de verificación de calificaciones?
10. Cuál es la parte que le resulta más frustrante al intentar validar los conocimientos de un candidato?

Refinar

Captación de candidatos:

1. En dónde suele buscar aplicantes y dónde publica ofertas de empleo?
2. Qué es lo primero que mira en el CV de un aplicante?

Verificación de calificaciones:

3. Cuándo mira CVs con el objetivo de realizar una nueva contratación, verifica las calificaciones de los aplicantes? Por qué? Cómo?
4. Alguna vez le pasó que un aplicante no tenía la calificación que decía tener?
5. En promedio, cuánto le lleva verificar la calificación de un aplicante?
6. Le ha pasado que no se ha podido contactar con la institución educativa que emitió el certificado que el aplicante dice tener? Qué ha hecho en este caso?

Opiniones:

7. Cree que hay algún defecto en el proceso de verificación de calificaciones? Siente que hay alguna posibilidad de optimización o mejora?
8. Cuál es la parte que le resulta más frustrante al intentar validar los conocimientos de un candidato?

Entrevista a coordinador de graduados de una universidad

Brainstorm:

1. Cómo es el proceso para emitir un diploma? Cuéntenos paso a paso que ocurre desde el momento en que el alumno aprueba el último examen hasta que recibe su diploma?
2. Cuánto tiempo lleva en promedio todo el proceso? Le parece razonable?
3. Cree que alguna parte podría automatizarse?
4. Haría algo diferente si de usted dependiera?
5. Cuál es su nivel de satisfacción general con el proceso?
6. Quienes son los agentes externos que intervienen?
7. Que pasa si un alumno pierde el título? Es posible solicitar una copia?

8. Guardan en algún sistema información de los graduados? En caso de ser así, qué información? Quién tiene acceso?
9. Puede una empresa contactarse para validar si alguien que dice ser graduado de su universidad efectivamente lo es? Hay alguna información que sea confidencial? De qué formas puede la empresa contactarse?
10. Los graduados reciben algún tipo de número de licencia o identificador que los identifique de manera única? En caso afirmativo, dónde puede validarse ese id?

Identificar y Ordenar temas

Emisión de diploma

1. Cómo es el proceso para emitir un diploma? Cuéntenos paso a paso que ocurre desde el momento en que el alumno aprueba el último examen hasta que recibe su diploma?
2. Cuánto tiempo lleva en promedio todo el proceso? Le parece razonable?
3. Quienes son los agentes externos que intervienen?

Diplomas ya emitidos

4. Que pasa si un alumno pierde el título? Es posible solicitar una copia?
5. Guardan en algún sistema información de los graduados? En caso de ser así, qué información? Quién tiene acceso?
6. Puede una empresa contactarse para validar si alguien que dice ser graduado de su universidad efectivamente lo es? Hay alguna información que sea confidencial? De qué formas puede la empresa contactarse?
7. Los graduados reciben algún tipo de número de licencia o identificador que los identifique de manera única? En caso afirmativo, dónde puede validarse ese id?

Opiniones

8. Cree que alguna parte podría automatizarse?
9. Cuál es su nivel de satisfacción general con el proceso?
10. Haría algo diferente si de usted dependiera?

Refinar

Emisión de diploma:

4. Cómo es el proceso para emitir un diploma? Cuéntenos paso a paso que ocurre desde el momento en que el alumno aprueba el último examen hasta que recibe su diploma?
5. Cuánto tiempo lleva en promedio todo el proceso? Le parece razonable?
6. Quienes son los agentes externos que intervienen?

Diplomas ya emitidos:

8. Que pasa si un alumno pierde el título? Es posible solicitar una copia?
9. Guardan en algún sistema información de los graduados? En caso de ser así, qué información? Quién tiene acceso?
10. Puede una empresa contactarse para validar si alguien que dice ser graduado de su universidad efectivamente lo es? Hay alguna información que sea confidencial? De qué formas puede la empresa contactarse?
11. Los graduados reciben algún tipo de número de licencia o identificador que los identifique de manera única? En caso afirmativo, dónde puede validarse ese id?
12. Cuál es el número aproximado de diplomas que se emiten anualmente?

Opiniones:

11. Cree que alguna parte podría automatizarse?
12. Cuál es su nivel de satisfacción general con el proceso?
13. Haría algo diferente si de usted dependiera?

Entrevista a graduado

Brainstorm

- Cuánto tiempo pasó aproximadamente desde que aprobó su último examen hasta que recibió su diploma?
- Qué le pareció este proceso?
- De qué forma elige presentar sus calificaciones a potenciales empleadores?
- Toma alguna medida para asegurar que su diploma es verídico?
- Le han pedido alguna forma de verificación de sus calificaciones? Cuánto ha llevado ese proceso?
- Le gustaría poder hacer uso de una forma innovadora para que sus posibles empleadores puedan verificar la validez de sus diplomas? Cree que esto puede diferenciarlo de forma positiva?

Identificar y Ordenar temas y Refinar

Obtención de diploma

- Cuánto tiempo pasó aproximadamente desde que aprobó su último examen hasta que recibió su diploma?
- Qué le pareció este proceso?

Presentación de calificaciones

- De qué forma elige presentar sus calificaciones a potenciales empleadores?

- Toma alguna medida para asegurar que su diploma es verídico?
- Le han pedido alguna forma de verificación de sus calificaciones? Cuánto ha llevado ese proceso?

Opinión acerca de aplicación

- Le gustaría poder hacer uso de una forma innovadora para que sus posibles empleadores puedan verificar la validez de sus diplomas? Cree que esto puede diferenciarlo de forma positiva?

Empathy Probe

Además de las entrevistas, se utilizará otra técnica de Design Thinking, llamada Empathy Probe. Las *empathy probes* ayudan a reducir la fricción dándole a los usuarios algo para hacer y crean espacio para que ellos digan lo que piensan. También nos da la oportunidad de observar a los usuarios antes de elegir dónde envolvernos más.

Cómo preparar una *empathy probe*

- **Crear una *probe***. Una *probe* es algo con lo que los usuarios interactúan mientras son entrevistados. Generalmente se crean *probes* en la forma de un set de cartas: una carta de pregunta y 7 cartas de respuesta que los usuarios debe ordenar. Una *probe* provoca la interacción y hace que el usuario cuente sus historias.
- **Involucrar a los usuarios.** Pedirles que compartan sus ideas en voz alta. Si la *probe* necesita que los usuarios consideren una experiencia en particular, es conveniente averiguar qué experiencia están considerando antes de que comienzan la actividad.
- **Seguir el hilo de comentarios interesantes, movimientos, o pausas**. Siempre comenzar con lo que has notado: “Mencionaste que … Podrías contarme un poco más acerca de eso?”, “Me di cuenta de que cuando leíste

esa carta, dijiste que … Qué estabas pensando?”. Prestar especial atención a material emotivo.

- **Buscar historias relevantes para las emociones que se descubren.**

***Empathy probe* a realizar**

Con el fin de comprender de mejor manera los requerimientos del sistema a desarrollar, se le pedirá a los interesados que ordenen cartas que contienen atributos de calidad del sistema en según su preferencia. Las cartas serán las siguientes:

- **Disponibilidad:** que el sistema esté funcionando un alto porcentaje del tiempo.
- **Performance**: que el sistema pueda cumplir con requerimientos de tiempo.
- **Seguridad:** que el sistema pueda proteger los datos de accesos no autorizados, mientras le provee acceso a usuarios y sistemas autorizados.
- **Usabilidad:** la capacidad de un software de ser comprendido, aprendido, usado y ser atractivo para el usuario, en condiciones específicas de uso

12.7. Transcripción de entrevistas

27/05/2019 - Paula Galloti - DVelop

Qué tipo de empresa es DVelop?

DVelop es una empresa que tiene dos unidades bien distintas. Por un lado, una unidad se encarga de hacer desarrollo de Software a medida, desarrollo de aplicaciones web o mobile para distintos tipos de industrias y clientes, en varios países. Por otro lado, tenemos una unidad que se encarga de desarrollar productos (por lo que vende licencias de software), y hace soporte y mantenimiento sobre estas licencias.

Y vos sos la encargada de buscar empleados para las dos secciones?

Sí, yo busco para las dos secciones, y después llevo a cabo las entrevistas junto con el encargado de cada sección.

En dónde suelen buscar aplicantes y dónde suelen publicar ofertas de empleo?

Eso depende un poco de rol que estemos buscando. A lo largo de estos 11 años hemos probado distintos canales. Los más comunes son *buscojob*, la página de *community* que tiene Genexus que es donde se publican ofertas laborales y también CVs de personas que buscan trabajo.

Osea que, mayoritariamente páginas web?

Sí, pero en los últimos cinco o seis llamados utilizamos una consultora. En realidad lo primero que hacemos es buscar por *linkedin* y páginas web, redes sociales y páginas web.

Cuando miran los CVs con el objetivo de realizar una nueva contratación, ustedes verifican las calificaciones de los aplicantes?

No, lo único que verificamos son las referencias. Hay algunos que nos envían los certificados que emiten las universidades, pero la verdad que formalmente no lo verificamos.

Y a ustedes les parecería útil si el día de mañana existiría una herramienta que permite verificar las calificaciones rápidamente, sin necesidad de hacer llamadas?

Sí, me parece que estaría bueno ver además con qué nota aprobó también. Creo que puede ser bueno. Lo que sí es cierto es que nosotros trabajamos con gente de gran diversidad académica en cuanto a de dónde provienen. Hay gente que viene de universidades, y otra que tiene una formación más técnica en otras instituciones. O sea, no sólo estaría bueno poder chequear que un ingeniero se haya recibido en una facultad de ingeniería, sino que también alguien que se haya hecho un curso de testing en Forge, por ejemplo. Porque muchas de las personas que entran acá no tienen título universitario.

Nosotros estamos pensando en desarrollar un sistema que ayude a verificar que los aplicantes tengan las calificaciones que dicen tener. Te voy a pedir si puedes ordenar por importancia los atributos de calidad del sistema, siendo estos: Disponibilidad, Performance, Seguridad, y Usabilidad.

1. Usabilidad
2. Seguridad
3. Disponibilidad
4. Performance

27/05/2019 - Adriana Fernández & Victoria Gonzalez - Oficina Graduados

Nosotros lo que tenemos es ingresando a gestión, una vez que son graduados, pueden obtener una constancia de título obtenido, y esas constancias se emiten con una validez de 30 días hábiles. Entonces eso tiene un código de verificación que las

empresas pueden usar para verificar la validez de las calificaciones presentadas. Una vez pasados los 30 días, las constancias se pueden volver a emitir.

Y eso es de acceso público?

Para la parte de las empresas sí, ingresan ese código y les aparece la información. Para esto, el graduado tiene que otorgar permiso porque hay una ley de protección de datos personales. Nosotros no podemos otorgar información de gente al azar. Es el mismo graduado que emite sus propias constancias, y después lo presenta para ofrecer pruebas de la autenticidad de sus diplomas.

Cómo es el proceso de solicitud de la constancia?

Eso es entrando a gestión, hay una parte que dice constancia de graduado.

Entonces es específico de esta universidad

Sí, exacto.

Y qué diferencia tiene una constancia de graduado con el título?

Tremenda diferencia! El título tiene el sello del MEC, del registro, la firma de todas las autoridades, y tiene otra validez que es diferente a la constancia.

Entonces a los efectos de presentar documentación, sirve una constancia y no es necesario llevar el título original?

No claro, después que tenes el título igual nosotros siempre recomendamos que lo fotocopien, que lo scaneen de los dos lados también como para tener un resguardo, porque los diplomas que se emiten por el MEC son registros únicos, entonces hay una prohibición de volver a expedirlos. Si después se les mancha o se les pierde no los podemos volver a dar el mismo cartón. Entonces para presentarlos a un trabajo, pueden usar estas fotocopias.

Hay alguna razón por la cual la constancia sólo dura 30 días?

La verdad que no lo sabemos.

Ustedes nos hablaron de la ley de protección de datos.

Sí, ahí tendrían que tener en cuenta si van a crear algo internacional, que cada lugar tiene sus propias leyes.

Y si una empresa llama acá y les dice que recibió un CV de un aplicante que dice ser un graduado de la ORT, no pueden darle esa información?

Por lo general, nos envían un mail solicitando algo que se llama validación de estudios, con un documento firmado por el aplicante (a veces es firma electrónica, como cuando la solicitud se hace desde otro país), y la documentación adjunta que el graduado les otorgó, como escolaridad por ejemplo. Nosotros lo único que hacemos, en caso de estar la autorización del graduado, es confirmarles los datos. Si llega a haber un dato que no está bien, les decimos que los datos son incorrectos, no los corregimos. Nosotros sólo podemos confirmar información que la empresa tiene, no darles datos adicionales.

Y ese es procedimiento específico de la ORT?

No, debería ser procedimiento de todas las universidades ya que es una ley.

Qué pasa si un egresado pierde un diploma?

Si es de grado, es decir, si fue al MEC, se le emite una constancia y se hace firmar por el MEC, pero no es el diploma mismo. Lo que hacen es certificar que tal persona tuvo registro en el MEC en tal fecha, firmado por tal escribano.

Y cuánto demora ese proceso más o menos, el de diploma y el que es en caso de extravío?

Si pierdes el diploma, se obtiene la constancia bastante rápido. Los diplomas demoran aproximadamente entre 4 y 5 meses en ser emitidos desde que son solicitados. Porque el MEC registra todos los títulos de todos los institutos privados. En nuestro caso, también los títulos intermedios. Tienen un volumen importante de diplomas.

Los graduados reciben algún tipo de identificador que los identifique de manera única?

No

30/05/2019 - Alejandra Diaz - Casa de Galicia

Contanos un poco de qué consiste tu trabajo

Nosotros seleccionamos personal de enfermería. En mi área específicamente contrato personal de enfermería y auxiliares de servicio. Lo que le pedimos al personal es el título, porque las auxiliares de servicio también tienen que hacer un curso que no es universitario. A todos les pedimos el título y se comprueba que esté autorizado por el ministerio, firmado, porque ha pasado que hay gente que los trae sin firmar, y eso no es válido para nosotros. Y dentro del currículum, por el área nuestra, les pedimos también la experiencia laboral y en qué áreas se desempeña. Eso es lo básico que pedimos nosotros para seleccionar personal en relación a la parte operativa.

Cómo verifican las calificaciones?

No tenemos un sistema de verificación. Nos traen una copia del título, o el original, y nos quedamos con una copia. Nos basamos en eso. En esta institución no verificamos llamando al ministerio o a la universidad.

Les parecería útil un sistema que permita hacerlo rápidamente?

Sí, sin duda. Aunque por suerte por ahora nunca nos pasó que alguien venga con un diploma falso. Nunca nadie me comentó que hubiera pasado algo parecido. Aparte como acá se toma gente con experiencia, en algunos casos llamamos a las instituciones para verificar su desempeño.

29/05/2019 - Josefina Vidal - CPA Ferrere

Primero que nada, puedes contar un poco de tu empresa y cuáles son las actividades que vos desempeñas?

Nosotros somos una consultora. Yo estoy en la parte de gestión de riesgos, haciendo todos los servicios relacionados a la misma, asesorando más que nada a la plaza financiera. Generalmente lo que hacemos cuando incorporamos gente nueva es buscar gente junior que está estudiando todavía para que haga un crecimiento en la empresa

Ustedes buscan aplicantes o les llegan a ustedes los aplicantes?

Sí, buscamos aplicantes por las redes sociales, o contactando directamente con las universidades. Generalmente es muy raro que alguien entre por recomendaciones.

Cuando miran los CVs con objetivos de realizar una contratación, verifican las calificaciones de los aplicantes?

No, miro la escolaridad, pero asumo como válida la información que me mandan.

Por qué no la verifican?

Porque nunca se me ocurrió que alguien pueda estar falsificando una escolaridad. Igual justo pasó un caso hace poco, en otro departamento, que una chica dijo que estaba recibida, la contrataron, y después no sé cómo saltó que no. Pero es el único caso del cual tengo conocimiento.

Les parecería útil un sistema que permita verificar las calificaciones rápidamente?

Sí, seguro, para que no vuelvan a pasar cosas como esa.

Nosotros estamos pensando en desarrollar un sistema que ayude a verificar que los aplicantes tengan las calificaciones que dicen tener. Te voy a pedir si puedes ordenar por importancia los atributos de calidad del sistema, siendo estos: Disponibilidad, Performance, Seguridad, y Usabilidad.

1. Seguridad
2. Usabilidad
3. Performance

4. Disponibilidad

03/06/2019 - Juan Palermo - Egresado de ORT

Contamos un poco en dónde estudiaste y cosa estudiaste

Estudié en la universidad ORT, en la carrera de Contador

Cuánto tiempo pasó desde que aprobaste tu último examen hasta que recibiste tu diploma?

Pasaron como 3 meses, creo.

En el tiempo entre que pasaste tu último examen y recibiste tu diploma, si querías aplicar para un trabajo, cómo hacías para demostrar la veracidad de tus calificaciones?

Con la escolaridad

De qué forma elegis presentar tus calificaciones a potenciales empleadores?

Con al escolaridad, también.

Tomás alguna medida para asegurar de que lo que presentas es verídico?

Yo no.

Y alguna vez te han pedido una forma de verificación?

Sí, me han pedido que la escolaridad esté sellada, por ejemplo.

Y cuánto tiempo te lleva ese proceso?

Y depende, si me lo piden y no lo tengo, tengo que ir hasta la facultad.

Te gustaría poder hacer uso de una forma innovadora para que sus posibles empleadores puedan verificar la validez de sus diplomas?

Obvio, estaría bueno que no tengan que pedírtelo, que ellos entren a algún portal o base de datos y ya lo puedan ver.

Crees que una herramienta así puede diferenciarte de forma positiva del resto de los aplicantes?

Sí, por supuesto. Una forma de validar las cosas que uno presenta en el currículum estaría muy bueno.

Nosotros estamos pensando en desarrollar un sistema que ayude a verificar que los aplicantes tengan las calificaciones que dicen tener. Te voy a pedir si puedes ordenar por importancia los atributos de calidad del sistema, siendo estos: Disponibilidad, Performance, Seguridad, y Usabilidad.

1. Seguridad
2. Usabilidad
3. Performance
4. Disponibilidad

31/05/2019 - Lorena Quiroga - CPA Ferrere

Primero que nada, nos podes contar un poco cuál es tu función y a qué se dedica la empresa en la que trabajas?

Trabajo en el área de consultoría en estrategia y capital humano, que es un área que brinda servicios en tema de recursos humanos a distintas empresas, tanto públicas como privadas, y yo estoy como responsable en el área de selección de personal para clientes, y otros proyectos que tengan que ver con evaluaciones, etc.

Cuando buscan personal, dónde suelen buscarlo y dónde publican las ofertas?

Por redes, por ejemplo LinkedIn, y nuestra firma publica en su facebook también y en portales de universidades. Nuestros avisos siempre salen en nuestra web y a partir de ahí generamos las distintas publicaciones para que los candidatos se postulen. En general es por redes.

Qué es lo primero que miran en el CV de los aplicantes?

Depende del cargo, en realidad nunca es lo mismo, siempre es distinto. SI tenes un requisito excluyente fuerte, en general vas a ese requisito primero, y después a la experiencia.

Esos requisitos excluyentes pueden ser calificaciones?

Pueden ser la carrera terminada, o que tengan buen nivel de inglés, no tanto por los diplomas, sino también por la antigüedad del diploma. Cuando son cargos muy junior, para nuestra firma la escolaridad es importante. En el caso de consultoría sí, la escolaridad importa.

Ustedes verifican las calificaciones de alguna forma? Por qué?

No. Si el cliente lo pide, pedimos la escolaridad. Eso sí, y en la escolaridad sale la calificación y si tiene los cursos terminados o no. Pero en general, con los cargos que nosotros manejamos, los clientes generalmente nos piden personal con experiencia de trabajo, que es lo que vale más.

Cuando verifican esas cosas, cómo lo hacen?

Pidiendo el título. Generalmente pedimos el título escaneado, o la escolaridad.

Y nunca les pasó que un aplicante diga que tenía una calificación que en realidad no tenía?

No porque pedimos la escolaridad o el título escaneado. Con llamados del sector público que tuvimos, y donde el título también es un requisito excluyente, ahí pedimos el diploma escaneado.

A vos te parecería útil o te agregaría valor una herramienta que te permita verificar rápidamente las calificaciones de los aplicantes?

Yo creo que sí, puede ser útil. Si una consultora maneja cargos junior, de primeras experiencias de trabajo en las que la escolaridad pesa, porque dice mucho del candidato, sí, puede ser muy útil. Puede estar bueno, antes de pedírsela, y de repente esperar a que te la mande. Porque me ha pasado de esperar algunos días a

que los chicos me manden la escolaridad y sería más fácil ya automáticamente sin ni siquiera pedirle el CV en linkedin, ver la escolaridad y saber si contactar o no al candidato. Porque ahora que lo pienso es muy fácil trucar una calificación con los programas de hoy en día.

Nosotros estamos pensando en desarrollar un sistema que ayude a verificar que los aplicantes tengan las calificaciones que dicen tener. Te voy a pedir si puedes ordenar por importancia los atributos de calidad del sistema, siendo estos: Disponibilidad, Performance, Seguridad, y Usabilidad.

1. Usabilidad
2. Seguridad: me parece importante porque hay un tema de confidencialidad. La escolaridad es de cada persona, y si lo quiere publicar o no es desición de cada persona. Debería decir si quiere autorizar a la universidad a confirmar su escolaridad
3. Performance
4. Disponibilidad

31/05/2019 - Jean Pierre Valencia, TangoCode

Por dónde buscas aplicantes?

Nosotros tenemos 3 formas básicamente. Una es *LinkedIn*, que es una de las más efectivas. *GlassDoor*, es otra que es buena. Y cuando necesitamos algo muy particular usamos a un *third party*, los *recruiters*. Siempre publicamos el aviso en *LinkedIn*, que es pago, en *GlassDoor* también es pago, y también usamos *Indeed*, que es gratis.

Qué es lo primero que miran en los CVs de los aplicantes?

Lo primero que miramos es la experiencia y las *skills* específicas.

Cuando miran los CVs, verifican las calificaciones de los aplicantes?

No. Nii las pedimos como requisitos. No pedimos títulos y mucho menos notas.

O sea que no las verifican porque para TangoCode son importantes los títulos?

Exactamente. Nunca pedimos certificaciones en general. Igual muchas certificaciones son fáciles de verificar porque tienen un ID asociado, que los aplicantes muchas veces ponen en *LinkedIn*. En general todas las conocidas lo tienen, Microsoft, Azure, etc. Me parece que cada vez se piden menos las certificaciones en esta industria. Nos importa que la gente sepa codificar, y muchas veces las certificaciones no dicen mucho. La industria del software es mucho más flexible que otras, como la medicina, en la que muchas veces una persona sin título no puede trabajar.

12.8. Análisis de tipo de prototipado

Prototipo desechables

Los prototipos desechables se refieren a la creación de un modelo que eventualmente será descartado, en lugar de convertirse en parte del software final entregado. Una vez que se logra la recopilación preliminar de requisitos, se construye un modelo simple del sistema para mostrar visualmente a los usuarios cómo se verán sus requisitos cuando se implementen en un sistema terminado.

El método utilizado para construirlos suele ser bastante informal, siendo el factor más importante la velocidad con la que se proporciona el modelo. El modelo se convierte en el punto de partida desde el cual los usuarios pueden volver a examinar sus expectativas y aclarar sus requisitos. Cuando se alcanza este objetivo, el modelo prototipo es desechado y el sistema se desarrolla formalmente en función de los requisitos identificados.

La ventaja principal de este tipo de prototipos es la velocidad con la cual estos pueden ser realizados. Esto permite que los usuarios pueden obtener *feedback* rápidamente sobre sus requisitos, y así refinarlos. De este modo, el equipo de desarrollo obtiene requerimientos de mejor calidad.

Prototipo evolutivo

La creación de prototipos evolutivos es bastante diferente de la creación de prototipos desechables. El objetivo principal cuando se utilizan prototipos evolutivos es construir un prototipo muy robusto de manera estructurada y refinarlo constantemente. La razón de este enfoque es que el prototipo evolutivo, cuando se construye, forma el corazón del nuevo sistema, y luego se construirán las mejoras y los requisitos adicionales.

Cuando se desarrolla un sistema utilizando prototipos evolutivos, el sistema se refina y reconstruye continuamente.

La principal ventaja de los prototipos evolutivos sobre los prototipos desechables es que son sistemas funcionales. Aunque es posible que no tengan todas las características que los usuarios han planeado, pueden usarse de manera provisional hasta que se entregue el sistema final.

Elección de prototipo

Luego de evaluar las dos opciones mencionadas, el equipo decidió utilizar prototipos desechables. Esto se debe a que el fin que se le daría a los prototipos sería realizar validaciones, y no se esperaba que estos formaran el “corazón del nuevo sistema”. Por esta razón, los prototipos desechables se adaptaron mejor a las necesidades del equipo, debido a la velocidad con la cual pueden ser construidos, permitiendo obtener *feedback* de forma más veloz.

12.9. Prototipado

Perfil de egresado (Vista de egresado)

Prototipo inicial

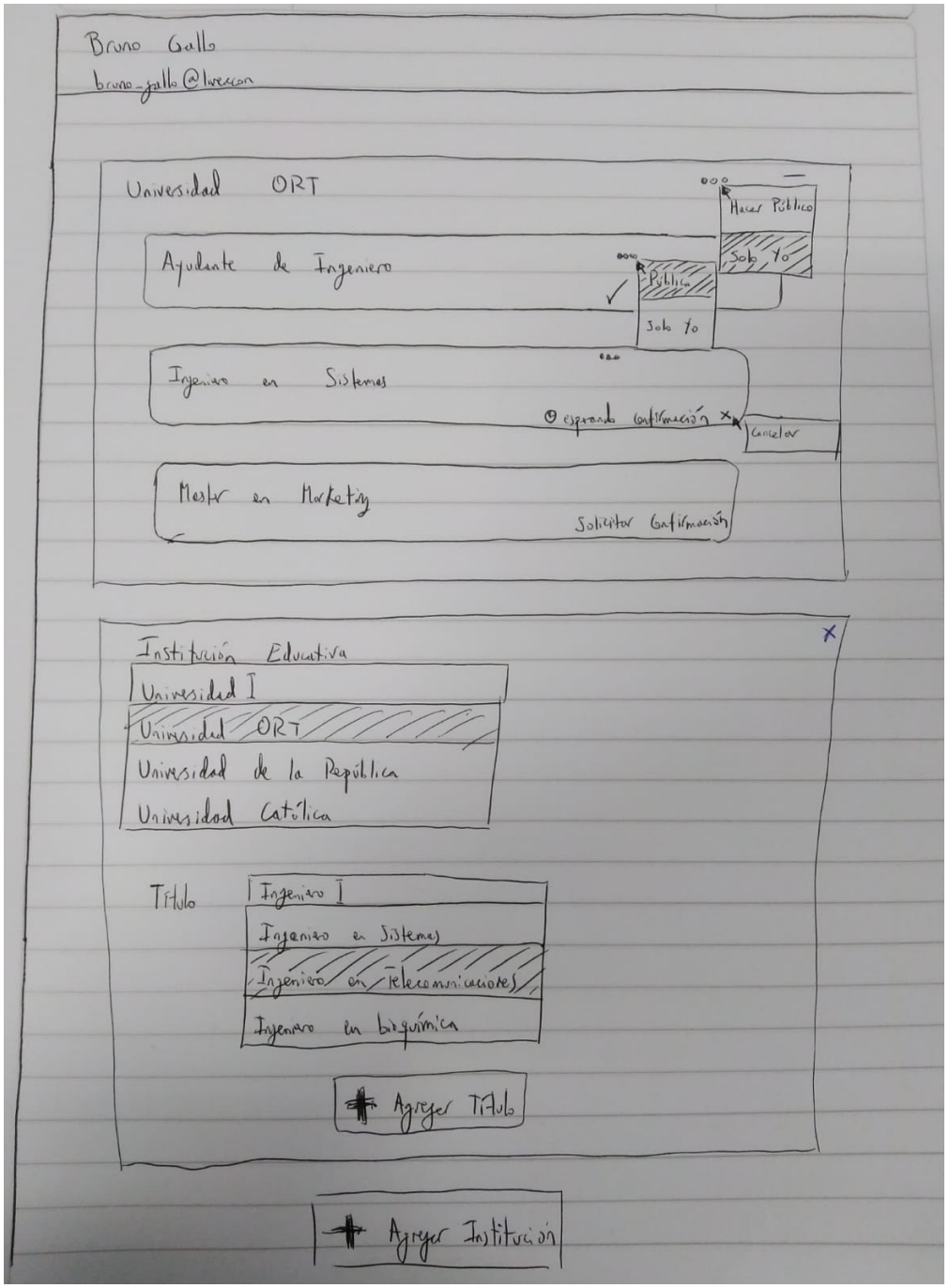


Imagen 73: prototipo inicial de perfil de egresado - vista de egresado

Durante la etapa de *discovery*, muchos/as de los entrevistados/as dieron importancia a la usabilidad del sistema. Por esta razón, se decidió validar los prototipos con Aric Drotts, una persona con amplia experiencia en interfaces de usuario, que ha desempeñado funciones de *Quality Assurance* en repetidas ocasiones a lo largo de su carrera en *TangoCode*.

Feedback

Imagen 74: validación de prototipo con Aric Drotts

- Permitir que el egresado pueda incluir más información junto con su título, tal como su promedio, su posición en la clase, o cualquier cosa que lo ayuden a destacarse del el resto.
- Agregar un tipo de visibilidad que sea “Sólo autorizado”, es decir, que ciertos títulos puedan ser vistos sólo por aquellos que el egresado autorizó.
- Agregar la posibilidad de poder visualizar mayor información acerca de la institución educativa:
    - Información de contacto

    - Imagen
    - Ubicación
    - Quizás hacer que se pueda visualizar esto de forma expandible

- Agregar la posibilidad de editar un título, para el caso en el que egresados vuelvan a tomar evaluaciones con el fin de obtener una nota más alta. En este caso, cada vez que se edite se deberá volver a pedir confirmación, y aquellos que visiten el perfil del egresado podrán visualizar la última versión del título confirmada por la institución educativa.

Segundo prototipo

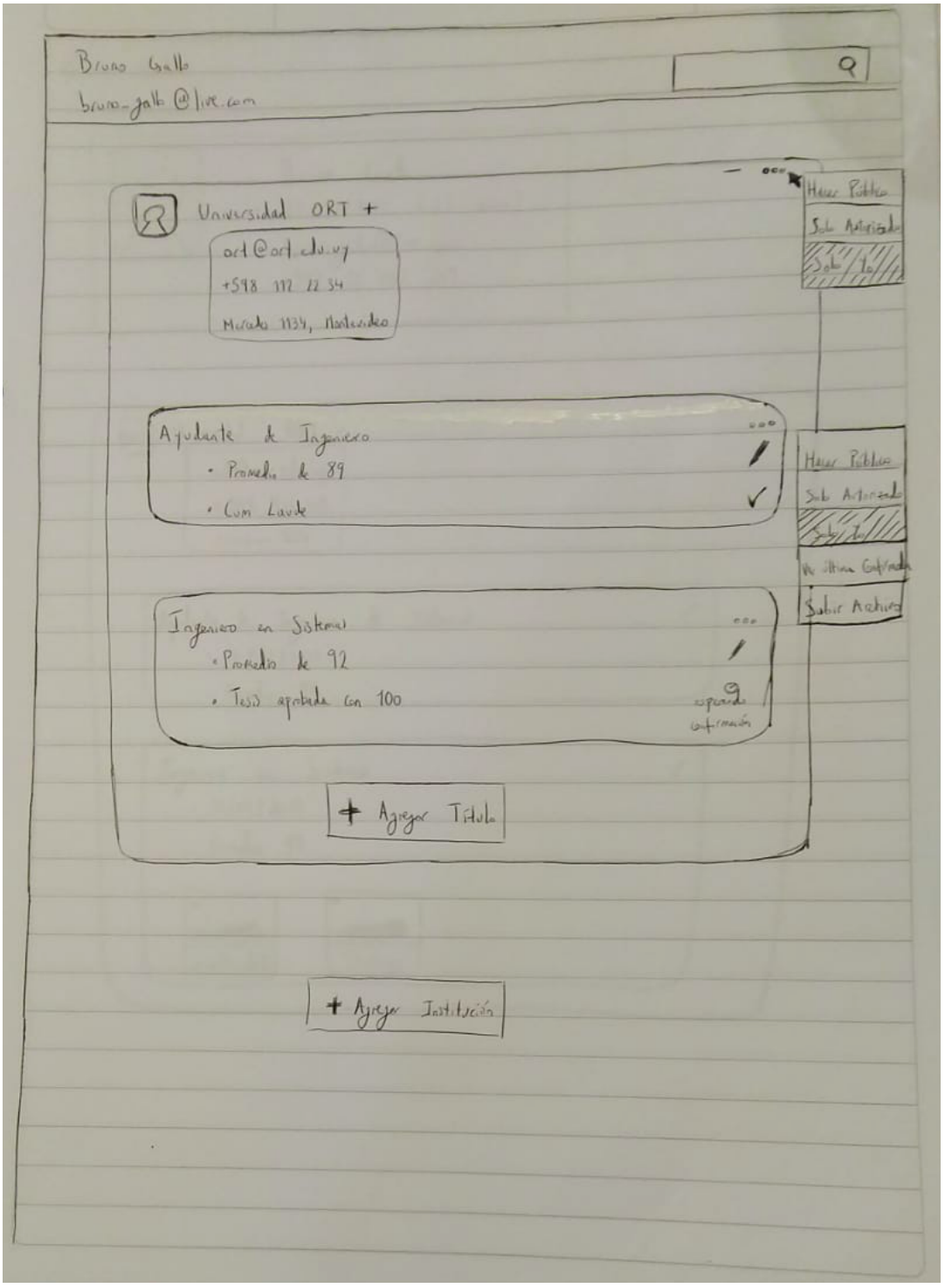


Imagen 75: prototipo final de perfil de egresado - vista de egresado

Se validó la nueva versión del prototipo con Aric Drotts una vez más, y se obtuvo la aprobación del mismo.

Imagen 76: segunda validación de prototipo con Aric Drotts

Se obtuvo además la aprobación de Juan Palermo, uno de los egresados entrevistados

Imagen 77: validación de prototipo con Juan Palermo

Perfil de egresado (Vista de empleador)

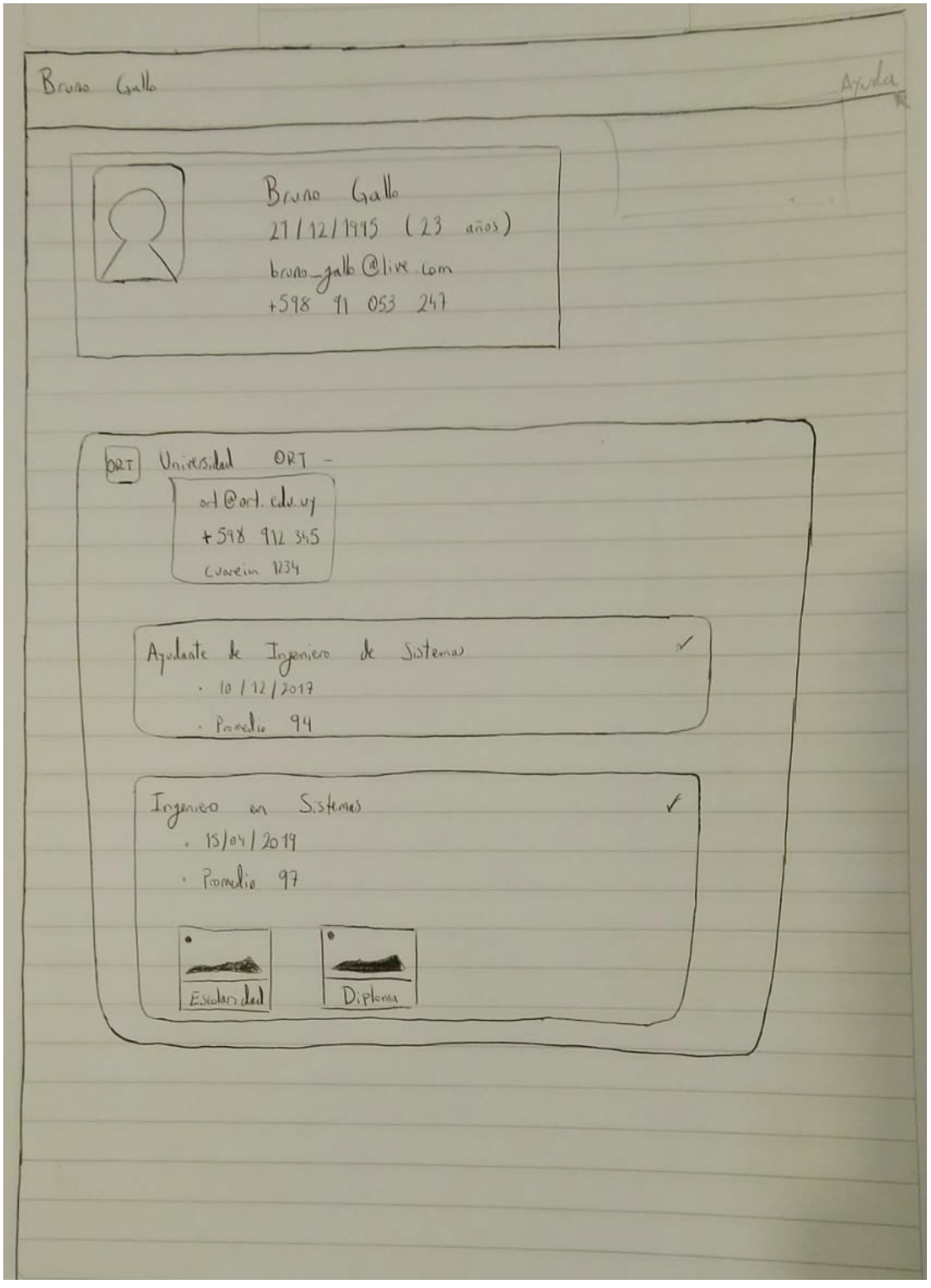


Imagen 78: prototipo de perfil de egresado - vista de empleador

Para validar el perfil del egresado como lo vería un empleador, se diseñó un prototipo y se validó con Josefina Vidal, de la consultora CPA Ferrere (una de las entrevistadas).

Imagen 79: validación de prototipo con Josefina Vidal

Feedback

En general, Josefina pareció estar conforme con el prototipo que le fue mostrado, diciendo que el mismo incluía casi toda la información necesaria para que los empleadores puedan evaluar al candidato. De todos modos, nos contó que muchas veces los candidatos que incluyen las notas con las cuales aprobaron las distintas instancias, olvidan incluir la escala que utiliza su institución académica. Es por esto que nos sería útil encontrar alguna forma de que los estudiantes siempre incluyan dicha escala. Para esto, una de las ideas fue incluir en la pantalla un texto de *tips*, y que cuando el usuario egresado pase el mouse por arriba, se despliegue un cuadro con distintos *tips* para mejorar su perfil, entre ellos el de incluir siempre la escala.

Perfil de egresado (Vista de institución educativa)

Para validar el perfil del egresado como lo vería la institución educativa, se diseñó un prototipo y se validó con María Noel Severi, de la academia ULearn.

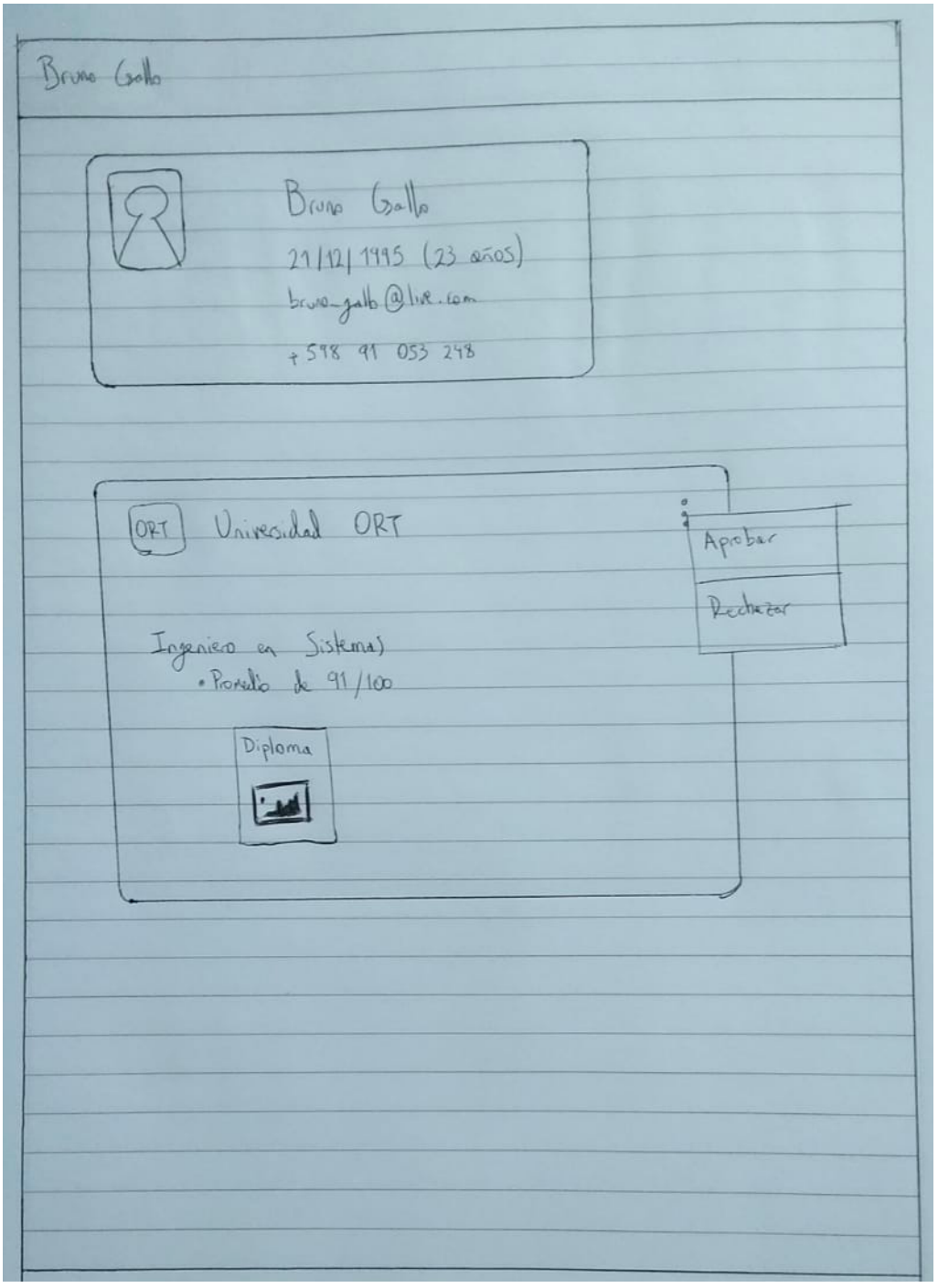


Imagen 80: prototipo de perfil de egresado - vista de institución educativa

Feedback

En general, María Noel mostró sentirse satisfecha con el prototipo que le fue mostrado. Sin embargo, ofreció algunas sugerencias:

- Que el estudiante pueda incluir información que sea solamente visible por la institución académica, pues no es relevante para los posibles empleadores, tal como número de estudiante.
- Que las instituciones académicas puedan indicarle a los estudiantes qué información deben incluir.
- Que las instituciones académicas puedan indicarle a los estudiantes qué archivos incluir, como por ejemplo, cartas firmadas que conceden autorización a las instituciones de aceptar o rechazar solicitudes de verificación.

12.10. Priorización de historias de usuario

Historia	Prioridad
Gestión de instituciones	
Procesar solicitud de institución académica	Must
Registro y edición de instituciones académicas	Must
Registro de egresado	
Registro de egresado	Must
Ingreso de egresado	Must
Ingreso con Google	Won’t
Gestión de perfil de egresado	
Imagen de perfil de egresado	Should
Datos de egresado	Should
Gestión de calificaciones de egresado	
Seleccionar institución académica por parte de egresado	Must
Ingreso de calificaciones	Must
Archivos de calificaciones	Must
Solicitar verificación a institución educativa	
Ingreso de requisitos por parte de egresado	Must
Solicitud de verificación de calificación	Must
Notificación a institución académica	Could
Recordatorio a institución académica	Won’t
Registro de institución académica	

Registro de institución académica	Must
Ingreso de institución académica	Must
Gestión de perfil de institución académica	
Datos de institución académica	Should
Imagen de perfil de institución académica	Should
Títulos de institución académica	Must
Requisitos de egresados	Must
Archivo de requerimientos	Should
Carga de títulos mediante CSV	Should
Tipo de título	Should
Restricciones de campos	Won’t
Aprobación o rechazo de solicitud de egresado	
Lista de solicitudes de verificación	Must
Visualización de perfil de egresado por parte de institución	Must
Visualización de archivo de calificación	Must
Aprobación o rechazo de solicitud de calificación	Must
Revocación de verificación	Must
Notificación a egresado	Could
Notificación a administrador	Won’t
Compartir perfil de egresado	
Generar vínculo para compartir perfil de egresado	Must

Ingreso a perfil de egresado con link generado	Must

Tabla 25: priorización de historias de usuario

12.11. User story maps

User story map inicial

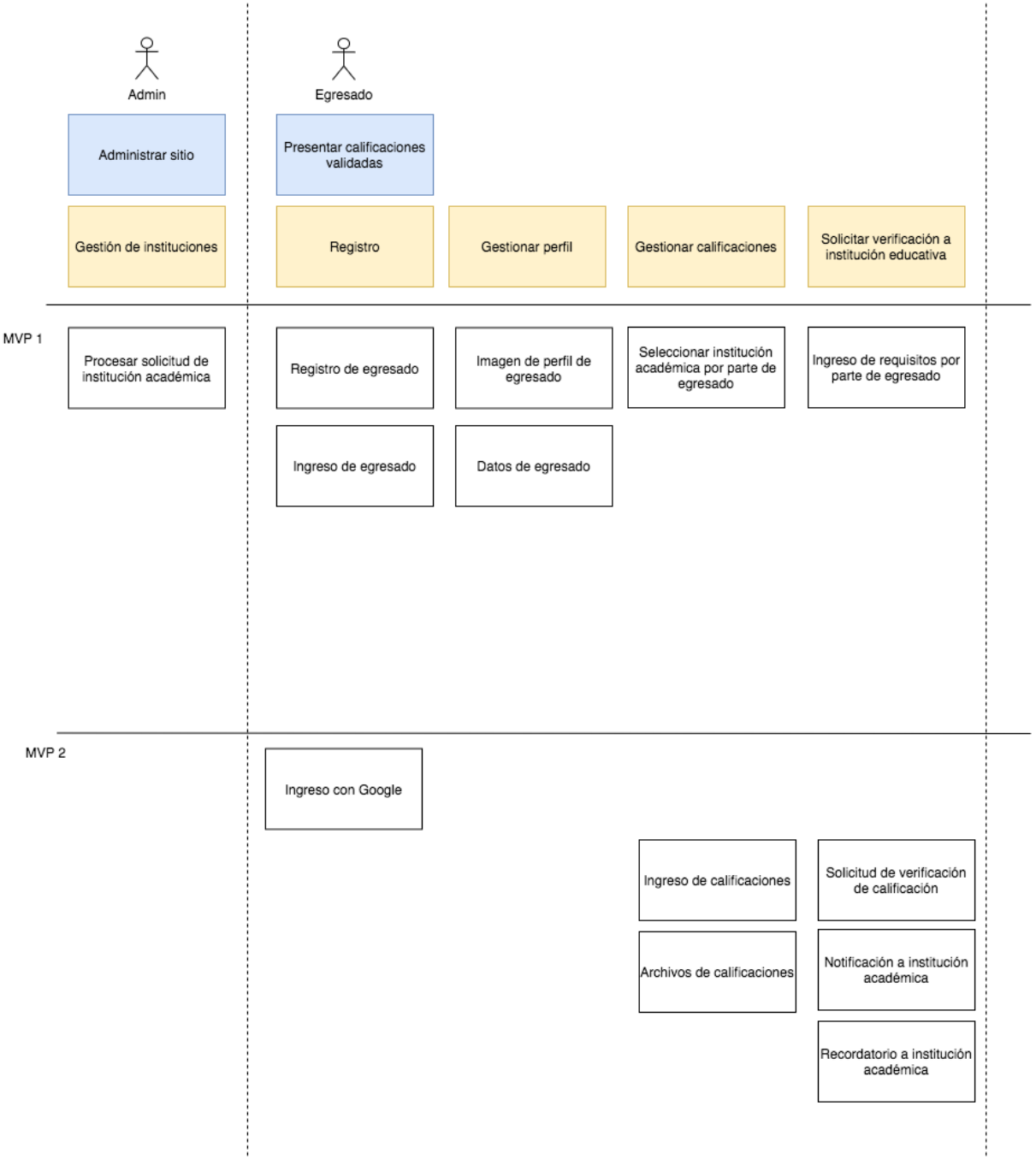


Imagen 81: *user story map* inicial - sección de administrador y egresado

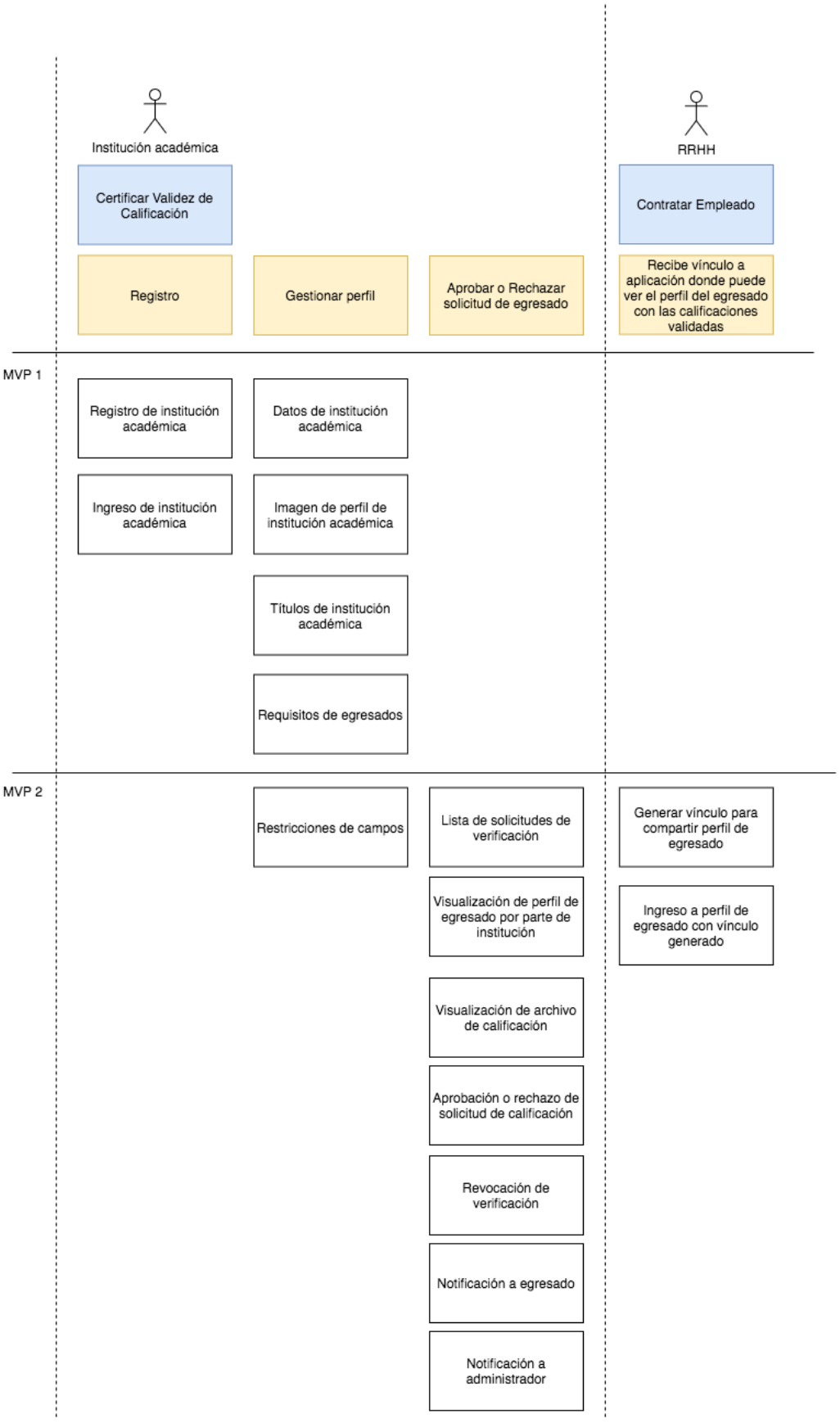


Imagen 82: *user story map* inicial - sección de institución educativa y empleador

User story map final

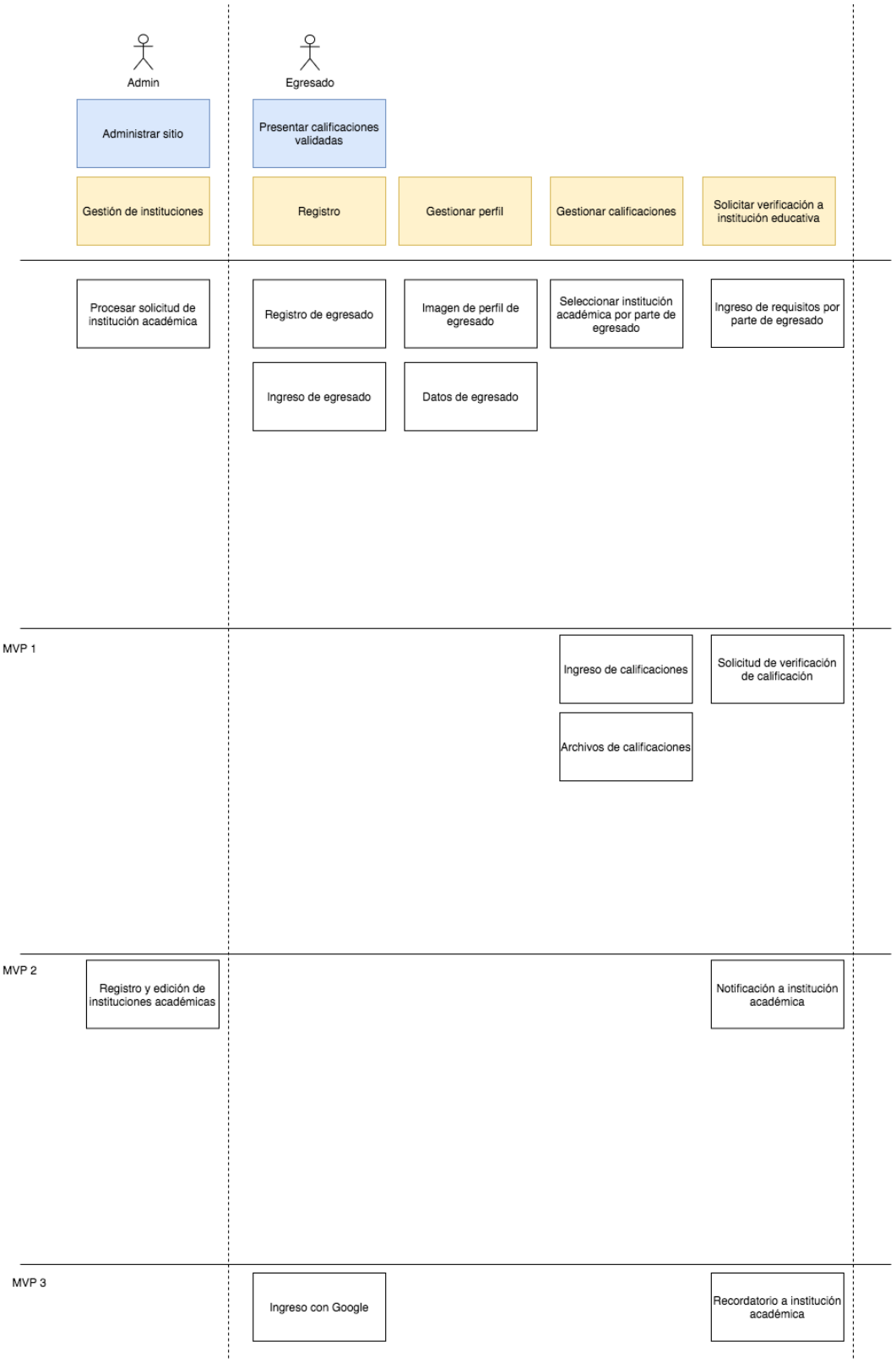


Imagen 83: *user story map* final - sección de administrador y egresado

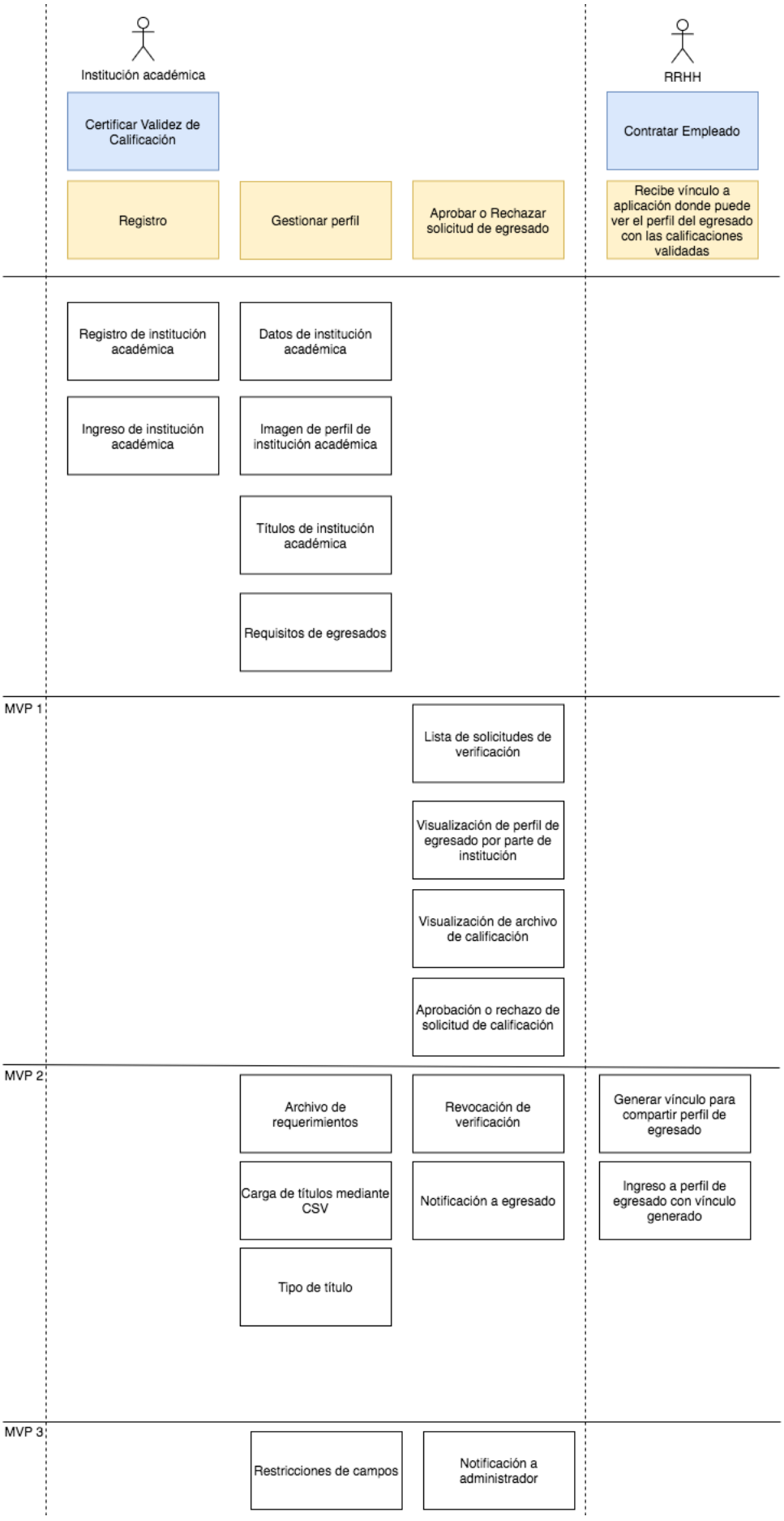

Institución académica
RRHH
Certificar Validez de Calificación
Contratar Empleado
Registro
Gestionar perfil
Aprobar o Rechazar solicitud de egresado
Recibe vínculo a aplicación donde puede ver el perfil del egresado con las calificaciones validadas
Registro de institución académica
Datos de institución académica
Ingreso de institución académica
Imagen de perfil de institución académica
Títulos de institución académica
Requisitos de egresados
MVP 1
Lista de solicitudes de verificación
Visualización de perfil de egresado por parte de institución
Visualización de archivo de calificación
Aprobación o rechazo de solicitud de calificación
MVP 2
Archivo de requerimientos
Revocación de verificación
Generar vínculo para compartir perfil de egresado
Carga de títulos mediante CSV
Notificación a egresado
Ingreso a perfil de egresado con vínculo generado
Tipo de título
MVP 3
Restricciones de campos
Notificación a administrador

Imagen 84: *user story map* final - sección de administrador y egresado

12.12. Historias de usuario

En este anexo se presenta el listado de historias de usuario agrupadas por sprint. De cada sprint se indica la fecha de inicio y fin, los puntos totales, y su objetivo.

Sprint 1 - 22/07/2019 - 05/08/2019 (19 puntos)

Objetivo: Establecer elementos básicos de configuración e implementar los registros a la aplicación.

1. Registro de egresado - 8 puntos

Descripción:

Como egresado
Quiero poder ingresar mis datos personales
Para registrarme en la aplicación

Criterios de aceptación:

1) Dado un egresado que rellena todos los campos, cuando selecciona la opción de registrarse, entonces se permite el registro y recibe un mail para confirmar su registro.

2) Dado un egresado no ingresa datos para nombre, apellido, email, contraseña, o repetición de contraseña, cuando intenta seleccionar la opción de registrarse, entonces no se permite el registro.

3) Dado un egresado ingresa email ya usado por otro usuario, cuando intenta registrarse, entonces no se permite el registro.

2. Registro de institución académica - 8 puntos

Descripción**:**

Como institución académica
Quiero poder ingresar mis datos
Para solicitar el registro en la aplicación

Criterios de aceptación:

1) Dada una institución académica que rellena todos los campos, cuando selecciona la opción de registrarse, entonces se permite el registro y recibe un mail para confirmar su registro.

2) Dada una institución académica que no ingresa datos para nombre, email, contraseña, o repetición de contraseña, cuando intenta seleccionar la opción de registrarse, entonces no se permite el registro.

3) Dada una institución académica que ingresa email ya usado por otro usuario, cuando intenta registrarse, entonces no se permite el registro.

3. Despliegue de aplicación - 2 puntos

Descripción:

Como desarrollador
Quiero poder deployar cambios en menos de 1 hora
Para agilizar el proceso de despliegue

Criterios de aceptación:

1) Desarrollador inicia proceso de deploy. Duración menor a 1 hora.

4. Repositorio de código - 1 punto

Descripción:

Como desarrollador
Quiero disponer de un repositorio donde compartir código con los demás integrantes del equipo
Para seguir el ciclo de desarrollo definido

Criterios de aceptación:

1) Dado un desarrollador, cuando desea acceder al código en el repositorio y subir cambios, entonces puede hacerlo sin problemas.

Sprint 2 - 05/08/2019 - 19/08/2019 (11 puntos)

Objetivo: Configurar los distintos ambientes de desarrollo y avanzar en los perfiles de egresados e instituciones académicas.

5. Herramienta de análisis de código - 1 punto

Descripción:

Como desarrollador
Quiero tener una herramienta de análisis de código
Para asegurar el cumplimiento de los estándares establecidos

Criterios de aceptación:

1) Dado cualquier código desarrollado, cuando este se está desarrollando, entonces se ve sometido a análisis utilizando herramientas automatizadas.

6. Ambientes de desarrollo - 2 puntos

Descripción:

Como desarrollador
Quiero disponer de diferentes ambientes de despliegue
Para asegurar el correcto funcionamiento de la aplicación antes de pasar a producción

Criterios de aceptación:

1) Dado el desarrollo del proyecto, cuando se desea realizar tareas de desarrollo, entonces existe un ambiente de desarrollo.

2) Dado el desarrollo del proyecto, cuando se desea realizar tareas de aseguramiento de calidad, entonces existe un ambiente de QA.

3) Dado el desarrollo del proyecto, cuando se desean realizar tareas de validación con usuarios, entonces existe un ambiente de Beta.

7. Ingreso de egresado - 2 puntos

Descripción:

Como egresado

Quiero poder ingresar a la aplicación con mis credenciales creadas

Para acceder a mi perfil y ingresar calificaciones

Criterios de aceptación:

1) Dado un egresado, cuando ingresa e-mail y contraseña correctos luego de confirmar su cuenta, entonces se permite el ingreso.

2) Dado un egresado, cuando ingresa e-mail y contraseña correctos sin confirmar su cuenta, entonces no se permite el ingreso.

3) Dado un egresado, cuando ingresa e-mail y contraseña incorrectos, entonces no se permite el ingreso.

4) Dado un egresado, cuando no ingresa email y/o contraseña, entonces no se permite el ingreso.

8. Datos de egresado - 2 puntos

Descripción:

Como egresado
Quiero poder modificar mis datos personales registrados
Para mantener mi información personal actualizada

Criterios de aceptación:

1) Dado un egresado que modifica datos de nombre, apellido, teléfono, y/o fecha de nacimiento, cuando selecciona la opción de guardar, entonces Se guardan los cambios

2) Dado un egresado que no ingresa datos para nombre o apellido, cuando selecciona la opción de guardar, entonces no se permite guardar los cambios

3) Dado un egresado que ingresa fecha de nacimiento inválida, cuando selecciona la opción de guardar, entonces no se permite guardar los cambios

9. Ingreso de institución - 2 puntos

Descripción:

Como institución académica
Quiero poder ingresar a la aplicación con mis credenciales creadas
Para acceder a mi perfil y verificar calificaciones

Criterios de aceptación:

1) Dada una institución académica que ingresa email y contraseña correctos luego de confirmar su cuenta y ser aceptada por administrador, cuando selecciona la opción de ingresar entonces se permite el ingreso.

2) Dada una institución académica que ingresa email y contraseña correctos sin confirmar su cuenta, cuando selecciona la opción de ingresar entonces no se permite el ingreso.

3) Dada una institución académica que ingresa email y contraseña correctos sin ser aceptada por administrador, cuando selecciona la opción de ingresar entonces no se permite el ingreso.

4) Dada una institución académica que ingresa email y contraseña incorrectos, cuando selecciona la opción de ingresar entonces no se permite el ingreso.

5) Dada una institución académica que no ingresa email y/o contraseña, cuando selecciona la opción de ingresar entonces no se permite el ingreso.

10. Datos de institución académica - 2 puntos

Descripción:

Como institución académica
Quiero poder modificar la información registrada
Para mantener mi información actualizada

Criterios de aceptación:

1) Dada una institución académica que modifica datos de nombre, teléfono, y/o dirección, cuando selecciona la opción de guardar entonces se guardan los cambios

2) Dada una institución académica que no ingresa datos para nombre, cuando selecciona la opción de guardar entonces no se permite guardar los cambios

Sprint 3 - 19/08/2019 - 02/09/2019 (7 puntos)

Objetivo: Facilitar la instalación del ambientes de desarrollo y avanzar en los perfiles de egresados e instituciones académicas.

11. Imagen de perfil de institución - 2 puntos

Descripción:

Como institución académica
Quiero poder subir una imagen a mi perfil
Para identificarme dentro de la plataforma

Criterios de aceptación:

1) Dada una institución académica que selecciona la opción de cambiar imagen de perfil, cuando aparece el selector de archivos, entonces sólo puede seleccionar archivos de tipo imagen.

2) Dada una institución académica que selecciona la opción de cambiar imagen de perfil, cuando selecciona un archivo de tipo imagen, entonces este se guarda como su nueva imagen de perfil

3) Dada una institución académica que selecciona un archivo, cuando este es de tamaño mayor a 1 MB, entonces se muestra mensaje de error.

12. Imagen de perfil de egresado - 2 puntos

Descripción:

Como institución académica
Quiero poder subir una imagen a mi perfil
Para identificarme dentro de la plataforma

Criterios de aceptación:

1) Dado un egresado que selecciona la opción de cambiar imagen de perfil, cuando aparece el selector de archivos, entonces sólo puede seleccionar archivos de tipo imagen.

2) Dado un egresado que selecciona la opción de cambiar imagen de perfil, cuando selecciona un archivo de tipo imagen, entonces este se guarda como su nueva imagen de perfil

3) Dado un egresado que selecciona un archivo, cuando este es de tamaño mayor a 1 MB, entonces se muestra mensaje de error.

13. Instalación de ambiente de desarrollo - 3 puntos

Descripción:

Como desarrollador
Quiero preparar el ambiente
Para poder comenzar a contribuir al proyecto en menos de 4 horas

Criterios de aceptación:

1) Dado un desarrollador que desea contribuir al proyecto, cuando prepara su ambiente de instalación, entonces puede instalar ambiente completo en tiempo menor a 4 horas, y una vez hecho, puede acceder a todo el código, ejecutar todas las pruebas unitarias, y también ejecutar el front end en su computadora de forma local.

Sprint 4 - 02/09/2019 - 16/09/2019 (10 puntos)

Objetivo: Finalizar proceso de registro de instituciones académicas y avanzar en el perfil de instituciones académicas.

14. Procesar solicitud de institución académica - 5 puntos

Descripción:

Como administrador
Quiero poder aceptar o rechazar solicitudes de registro de instituciones académicas
Para permitir o no el ingreso de instituciones al sistema

Criterios de aceptación:

1) Dado un administrador, cuando accede al sistema, entonces debe poder ver todas las solicitudes pendientes de registro de instituciones académicas

2) Dado un administrador que acepta una solicitud, cuando la institución académica luego intente ingresar en el sistema con su usuario, entonces puede hacerlo.

3) Dado un administrador que acepta una solicitud, cuando el administrador navega hacia la pestaña de solicitudes aceptadas, entonces puede ver la solicitud recién aceptada en la parte superior de la lista.

4) Dado un administrador que rechaza una solicitud, cuando la institución académica luego intente ingresar en el sistema con su usuario, entonces no puede hacerlo.

5) Dado un administrador que rechaza una solicitud, cuando el administrador navega hacia la pestaña de solicitudes rechazadas, entonces puede ver la solicitud recién rechazada en la parte superior de la lista.

15. Títulos de institución académica - 5 puntos

Descripción:

Como institución académica
Quiero poder definir la lista de títulos
Para que los egresados puedan solicitar su verificación

Criterios de aceptación:

1) Dada una institución académica que ingresa nombre de título, cuando selecciona la opción de “agregar”, entonces el mismo se agrega a la lista de títulos existentes en orden alfabético.

2) Dada una institución académica no ingresa nombre de título, cuando selecciona la opción de “agregar”, entonces no se permite agregar un título sin nombre

3) Dada una institución académica que edita el nombre de un título existente, cuando selecciona la opción de “guardar”, entonces se guarda el nuevo nombre del título

4) Dada una institución académica, cuando elimina un título existente, entonces se muestra un mensaje de confirmación, y al seleccionar la opción de “aceptar” el título es eliminado.

Sprint 5 - 16/09/2019 - 30/09/2019 (16 puntos)

Objetivo: Finalizar funcionalidades de perfil de instituciones académicas e iniciar el desarrollo del ingreso de calificaciones por parte de egresados.

16. Requisitos de egresados - 8 puntos

Descripción:

Como institución académica
Quiero poder crear un conjunto de requisitos
Para poder verificar la identidad de los egresados

Criterios de aceptación:

1) Dada una institución académica que ingresa nombre, tipo, y descripción de requisito, cuando selecciona la opción “Agregar”, entonces el requisito es agregado a la lista de requisitos.

2) Dada una institución académica que no ingresa nombre y/o tipo, cuando intenta seleccionar la opción “Agregar”, entonces no se permite seleccionar la opción “Agregar”.

3) Dada una institución académica que modifica el nombre y/o descripción de un requisito, Cuando selecciona la opción “guardar”, entonces se guardan los cambios.

4) Dada una institución académica, cuando intenta cambiar el tipo de un requisito ya guardado, entonces el sistema no permite hacerlo.

5) Dada una institución académica, cuando elimina un requisito existente, entonces se muestra un mensaje de confirmación y al seleccionar la opción de “aceptar”, el requisito es eliminado.

17. Selección de institución académica por parte de egresados - 3 puntos

Descripción:

Como egresado
Quiero poder seleccionar una institución académica
Para poder ingresar las calificaciones que obtuve en la misma

Criterios de aceptación:

1) Dado un egresado, cuando intenta ingresar calificaciones, entonces debe poder visualizar una lista con todas las instituciones académicas, pudiendo filtrar por nombre.

2) Dado un egresado, cuando selecciona una institución académica, entonces aparece una sección debajo con el nombre de la misma como título.

3) Dado un egresado, cuando selecciona una institución académica, entonces esta no debe aparecer más entre las opciones seleccionables.

18. Ingreso de requisitos por parte de egresados - 5 puntos

Descripción:

Como egresado
Quiero poder completar los datos requeridos por la institución académica seleccionada
Para verificar mi identidad

Criterios de aceptación:

1) Dado un egresado que ha seleccionado una institución, cuando se dirige a la sección de la misma, entonces se deben visualizar todos los requisitos ingresados por la misma.

    a) Si el requisito es de tipo “campo”, se debe visualizar un campo en el cual se deben poder ingresar datos.

    b) Si el requisito es de tipo “PDF”, se debe poder subir únicamente archivos de tipo PDF.

c) Si el requisito es de tipo “Imagen”, se debe poder subir únicamente imágenes.

2) Dado un egresado que ha seleccionado una institución, cuando elimina los archivos subidos a requerimientos de tipo “PDF” e “Imagen”, entonces los mismos dejan de estar asociadas a los requerimientos.

3) Dado un egresado que ha seleccionado una institución, cuando edita los datos ingresados a requerimientos de tipo “Campo”, entonces los nuevos datos son guardados.

Sprint 6 - 30/09/2019 - 14/10/2019 (12 puntos)

Objetivo: Finalizar el desarrollo del ingreso de calificaciones por parte de egresados y la solicitud de verificación de dichas calificaciones.

19. Ingreso de calificaciones - 5 puntos

Descripción:

Como egresado
Quiero poder ingresar calificaciones asociadas a títulos
Para pedir su verificación por parte de las instituciones

Criterios de aceptación:

1) Dado un egresado que ha seleccionado una institución, cuando se dirige a la sección de la misma, entonces se deben visualizar todos los títulos que ofrece la institución.

2) Dado un egresado que ha seleccionado una institución, cuando selecciona un título, entonces se debe visualizar una sección para el título seleccionado.

3) Dado un egresado que ha seleccionado un título, cuando se dirige a la sección dedicada al título seleccionado, entonces se debe poder ingresar información asociada al título en forma de texto.

20. Archivos de calificaciones - 2 puntos

Descripción:

Como egresado
Quiero poder adjuntar archivos a mis calificaciones
Para agregar información a mis calificaciones

Criterios de aceptación:

1) Dado un egresado que ha seleccionado un título, cuando se dirige a la sección dedicada al título seleccionado, entonces debe poder adjuntar la cantidad de archivos deseada, de tipo “PDF” o imagen.

2) Dado un egresado que ha adjuntado un archivo a una calificación, cuando elimina el archivo, entonces el mismo deja de estar asociado a la calificación.

3) Dado un egresado que ha adjuntado un archivo a una calificación, cuando edita los nombres de los archivos subidos, entonces los nuevos nombres son guardados.

21. Solicitación de verificación de calificación - 5 puntos

Descripción:

Como egresado
Quiero poder solicitar a la institución académica de la cual egresé que certifique la validez de mi diploma
Para poder tener la calificación verificada en mi perfil

Criterios de aceptación:

1) Dado un egresado que ha seleccionado una institución, cuando ingresa datos para todos los requerimientos solicitados por la institución académica, entonces debe poder solicitar la verificación de la calificación.

2) Dado un egresado que ha seleccionado una institución, cuando no ingresa datos para todos los requerimientos solicitados por la institución académica, entonces no debe poder solicitar la verificación de la calificación.

3) Dado un egresado, que ha solicitada la la verificación de la calificación, cuando deshace la acción, entonces la solicitud es eliminada

4) Una vez solicitada la la verificación de la calificación, el egresado no puede volver a solicitarla.

5) Previo a solicitar una verificación, el egresado deberá seleccionar una casilla asegurando que comprende la acción que está realizando.

Sprint 7 - 14/10/2019 - 28/10/2019 (14 puntos)

Objetivo: Finalizar el desarrollo del proceso de verificación de calificaciones.

22. Lista de solicitudes de verificación - 3 puntos

Descripción:

Como institución académica
Quiero poder visualizar la lista de solicitudes de certificación aprobadas, rechazadas y pendientes
Para saber cuáles debo verificar, y cuáles ya he verificado

Criterios de aceptación:

1) Dada una institución académica, cuando un egresado solicita la verificación de una calificación, entonces esta deberá aparecer en la pestaña de solicitudes pendientes en la pantalla de solicitudes de la institución académica.

2) Dada una institución académica, cuando se dirige a la pantalla de solicitudes, entonces debe poder ver un resumen de la información del egresado desde la misma pantalla, sin necesidad de ingresar al perfil del egresado.

23. Visualización de perfil de egresado por parte de institución - 5 puntos

Descripción:

Como institución académica
Quiero poder visualizar el perfil de un egresado que ha solicitado mi verificación
Para comprobar que la información ingresada sea correcta y poder verificar su identidad

Criterios de aceptación:

1) Dada una institución académica, cuando selecciona una solicitud, entonces el sistema debe redirigir al usuario al perfil del egresado, en el cual sólo se verá la información ingresada por el egresado para la propia institución académica. Además, la institución académica sólo deberá ver información perteneciente a títulos para los cuales el egresado ha solicitado verificación previamente.

2) Dada una institución académica, cuando se encuentra en el perfil de un egresado, entonces debe poder ver los requisitos ingresados por el egresado

3) Dada una institución académica, cuando se encuentra en el perfil de un egresado, entonces debe poder descargar los archivos asociados a requisitos.

4) Dada una institución académica que se encuentra en el perfil de un egresado, cuando descarga un archivo que ha sido modificado sin autorización, entonces se debe mostrar un mensaje de error indicando lo sucedido, y el archivo no estará más disponible.

24. Visualización de archivo de calificación - 3 puntos

Descripción:

Como institución académica
Quiero poder ver la imagen o pdf subida por un egresado como prueba de su graduación
Para corroborar que sean válidos

Criterios de aceptación:

1) Dada una institución académica que se encuentra en el perfil de un egresado, cuando la institución académica descarga aquellos archivos vinculados a calificaciones, entonces los mismos deben poder ser descargados.

2) Dada una institución académica que se encuentra en el perfil de un egresado, cuando descarga un archivo asociado a una calificación que ha sido

modificado sin autorización, entonces se debe mostrar un mensaje de error indicando lo sucedido, y el archivo no estará más disponible.

25. Aprobación o rechazo de solicitud de verificación - 3 puntos

Descripción**:**

Como institución académica
Quiero poder optar por aceptar o rechazar una solicitud de certificación de título según la veracidad del mismo
Para que el egresado pueda contar con la calificación verificada en su perfil, o para evitar fraudes

Criterios de aceptación:

1) Dada una institución académica, cuando verifica una calificación, entonces la solicitud deberá luego poder verse en la pestaña de solicitudes aceptadas en la pantalla de lista de solicitudes.

2) Dada una institución académica, cuando verifica una calificación, entonces el egresado debe poder ver la calificación verificada en su perfil.

3) Dada una institución académica, cuando rechaza una calificación, entonces la solicitud deberá luego poder verse en la pestaña de solicitudes rechazadas en la pantalla de lista de solicitudes.

4) Dada una institución académica, cuando rechaza una calificación, entonces se debe bloquear el usuario del egresado.

5) Dada una institución académica, cuando se encuentra procesando la solicitud de un egresado, entonces debe poder solicitar un cambio en la calificación, incluyendo un mensaje de qué debe cambiar el egresado.

6) Dada una institución académica, cuando solicita un cambio a una calificación, entonces se la solicitud deberá luego poder verse en la pestaña de cambios solicitados en la pantalla de lista de solicitudes.

Sprint 8 - 28/10/2019 - 11/11/2019 (16 puntos)

Objetivo: Una vez finalizado el *release* anterior, comenzar con la implementación de las historias de mayor prioridad tomando en cuenta aquellas que surgieron luego de realizar la primer validación con Pablo San Nicolás.

26. Restablecimiento de contraseña - 5 puntos

Descripción:

Como usuario
Quiero poder restablecer mi contraseña en caso de olvido
Para poder ingresar al sistema

Criterios de aceptación:

1) Dado un usuario que ha olvidado su contraseña, cuando el usuario selecciona la opción “olvidé mi contraseña” en la pantalla de ingreso, entonces es redirigido a una pantalla en la cual puede ingresar su email.

2) Dado un usuario que se encuentra en dicha pantalla, cuando el usuario ingresa su email, entonces debe recibir un vínculo por email para restablecer su contraseña.

3) Dado un usuario que se selecciona dicho vínculo, cuando es redirigido al sistema, entonces debe poder seleccionar una nueva contraseña. La contraseña debe ser de 8 caracteres o más.

4) Dado un usuario que ha restablecido su contraseña, cuando intenta ingresar con su nueva contraseña, entonces el sistema debe permitir su ingreso.

5) Dado un usuario que ha restablecido su contraseña, cuando intenta ingresar con su antigua contraseña, entonces el sistema no debe permitir su ingreso.

27. Archivos de requerimientos - 3 puntos

Descripción:

Como institución académica
Quiero poder proporcionar un archivo de ejemplo para los requerimientos

Para que los egresados puedan utilizarlo

Criterios de aceptación:

1) Dada una institución académica, cuando ingresa un requerimiento de tipo “PDF” o “Imagen”, entonces debe poder adjuntar un archivo dicho requerimiento.

2) Dada una institución académica que adjunta un archivo, cuando intenta descargar el archivo adjunto a los requerimientos, entonces debe poder hacerlo.

3) Dada una institución académica, cuando elimina el archivo adjunto a un requerimiento, el mismo deja de estar adjunto a dicho requerimiento.

4) Dado un egresado que ha seleccionado una institución, cuando un requerimiento de tipo “PDF” o “Imagen” tiene un archivo asociado, entonces el egresado debe poder descargarlo.

28. Registro y edición de instituciones académicas - 8 puntos

Descripción:

Como administrador
Quiero poder crear y editar perfiles de instituciones académicas
Para que estas puedan utilizar el sistema

Criterios de aceptación:

1) Dado un administrador, cuando ingresa en el sistema, entonces debe poder visualizar una lista con todas las instituciones académicas registradas.

2) Dado un administrador, cuando ingresa en el sistema, entonces debe poder crear una nueva institución académica.

3) Dado un administrador autenticado, cuando ingresa nombre, email de ingreso, y contraseña, entonces se permite el registro.

4) Dado un administrador autenticado, cuando no ingresa nombre, email de ingreso, y/o contraseña, entonces no se permite el registro.

5) Dado un administrador autenticado, cuando crea una institución académica, entonces se redirige al administrador al perfil de la misma.

6) Dado un administrador autenticado, cuando selecciona una institución académica de la lista, entonces se redirige al perfil de la institución académica.

7) Dado un administrador, cuando se encuentra en el perfil de una institución, entonces puede editar datos de la institución académica, tales como imagen de perfil, nombre, dirección, y teléfono.

8) Dado un administrador, cuando se encuentra en el perfil de una institución, entonces puede editar los títulos de la institución académica.

9) Dado un administrador, cuando se encuentra en el perfil de una institución, entonces puede editar los requerimientos de la institución académica.

10) Dado un administrador, cuando se encuentra en el perfil de una institución, entonces puede generar un token para editar el email de ingreso para el usuario de la institución académica, el cual es enviado a su email.

11) Dado un administrador, cuando se encuentra en el perfil de una institución, entonces no debe poder utilizar un token generado por él mismo para editar el email de ingreso para el usuario de la institución académica.

12) Dado un administrador, cuando se encuentra en el perfil de una institución, entonces debe poder utilizar un token generado por otro administrador para editar el email de ingreso para el usuario de la institución académica.

13) Dado un administrador, cuando se encuentra en el perfil de una institución, entonces no debe poder aprobar calificaciones por parte de la institución académica

Sprint 9 - 11/11/2019 - 25/11/2019 (15 puntos)

Objetivo: Finalizar la implementación de las historias que surgieron luego de realizar la primer validación con San Nicolás y comenzar con el desarrollo de la funcionalidad de compartir perfil.

29. Carga de títulos mediante CSV - 5 puntos

Descripción:

Como institución académica
Quiero disponer de la posibilidad de cargar los datos referidos a las distintas carreras mediante la importación de un archivo CSV
Para facilitar este proceso

Criterios de aceptación:

1) Dado un administrador o el usuario de la institución académica, cuando se encuentran en el perfil de la institución académica, entonces deben poder subir títulos cargados de un archivo CSV.

2) Dado un administrador o el usuario de la institución académica, cuando seleccionan el archivo, entonces se visualizarán aquellos títulos nuevos presentes en el archivo CSV.

3) Dado un archivo CSV seleccionado, cuanto contiene errores, entonces se deberán poder visualizar.

4) Dada la selección de un archivo CSV sin errores, cuando se selecciona “Aceptar”, entonces los nuevos títulos serán mostrados en el perfil de la institución.

30. Revocación de verificación - 5 puntos

Descripción:

Como institución académica
Quiero poder revocar una validación en caso de haber sido dada por error
Para corregir mis errores

Criterios de aceptación:

1) Dada una institución académica, cuando revoca la validación de una calificación, entonces la misma no se debe ver como validada en el perfil del egresado, sino que se debe ver como pendiente.

2) Dada una institución académica, cuando revoca el rechazo de una calificación, entonces, en caso de ser el único rechazo recibido por el egresado, su usuario se desbloquea.

3) Dada una institución académica, cuando revoca la solicitud de cambio de una calificación, entonces la misma no se debe ver con un cambio solicitado en el perfil del egresado, sino que se debe ver como pendiente.

31. Generar vínculo para compartir perfil de egresado - 5 puntos

Descripción:

Como egresado
Quiero poder generar un vínculo para compartir mi perfil
Para que otros puedan ver mis calificaciones

Criterios de aceptación:

1) Dado un egresado que se encuentra en su perfil, cuando cuenta con calificaciones verificadas, entonces debe poder compartir su perfil, generando un vínculo de una duración de 1 día, 1 semana, 1 mes, o 3 meses.

2) Dado un egresado que se encuentra en su perfil, cuando no cuenta con calificaciones verificadas, entonces no debe poder generar el vínculo compartir su perfil.

Sprint 10 - 25/11/2019 - 09/12/2019 (14 puntos)

Objetivo: Finalizar el desarrollo de la funcionalidad de compartir perfil.

32. Ingreso a perfil de egresado con vínculo generado - 8 puntos

Descripción:

Como empleador
Quiero poder ingresar al perfil de un egresado haciendo uso del vínculo compartido por él
Para poder ver sus calificaciones verificadas

Criterios de aceptación:

1) Dado un usuario no autenticado, cuando hace uso de un vínculo válido y no expirado, entonces debe poder visualizar el perfil del egresado

2) Dado un usuario no autenticado que hace uso de un vínculo válido, cuando ingresa al perfil del egresado, entonces debe poder ver sólo calificaciones verificadas, con visibilidad seleccionada como “sólo autorizado” por el egresado.

3) Dado un usuario no autenticado que hace uso de un vínculo válido, cuando ingresa al perfil del egresado, entonces debe poder descargar los archivos asociados a calificaciones.

4) Dado un usuario no autenticado, cuando hace uso de un vínculo no válido o expirado, entonces no debe poder visualizar el perfil del egresado.

33. Tipo de título - 2 puntos

Descripción:

Como institución académica
Quiero poder establecer por cada carrera si se trata de un título técnico, terciario, master, o postgrado según estándar ISCED 2011
Para agregar información al título

Criterios de aceptación:

1) Dado un administrador o una institución académica, cuando ingresa un título, entonces se debe seleccionar el tipo de título según el estándar ISCED 2011. Si no se ingresa un tipo de título, no se debe permitir guardar el mismo.

2) Dado un administrador o una institución académica, cuando edita un título, entonces se debe poder editar el tipo de título.

3) Dado un archivo CSV de títulos, cuando se va a realizar el importe, entonces se debe especificar el tipo de título en el archivo.

4) Dado un archivo CSV de títulos dentro del cual existen títulos sin tipo, cuando se va a realizar el importe, entonces se debe mostrar un error al intentar importarlo.

34. Notificación a institución académica - 2 puntos

Descripción:

Como institución académica
Quiero recibir una notificación cada vez que haya una nueva solicitud de verificación
Para poder procesar la solicitud

Criterios de aceptación:

1) Dado un egresado, cuando solicita la verificación de una calificación, entonces se debe enviar un mail a la institución académica. El mismo debe contener un vínculo al perfil del egresado.

35. Notificación a egresado - 2 puntos

Descripción:

Como egresado
Quiero recibir una notificación cada vez que haya aprobado o rechazado mi solicitud de verificación
Para estar al tanto de la situación

Criterios de aceptación:

1) Dada una institución académica, cuando acepta, rechaza, o solicita cambios en una calificación, entonces se debe enviar un email al egresado. El mismo debe contener un vínculo a su perfil.

Trabajo pendiente

36. Notificación a administrador - 2 puntos

Descripción:

Como administrador
Quiero recibir una notificación cada vez que una validación es rechazada por algún motivo
Para estar al tanto de la situación

Criterios de aceptación:

1) Dada una institución académica, cuando rechaza una calificación, entonces se debe enviar un email al administrador notificándole del hecho. El mismo debe contener un vínculo al perfil del egresado.

37. Recordatorio a institución - 2 puntos

Descripción:

Como egresado
Quiero tener la opción de enviar un recordatorio a la institución académica de la cual egresé
Para que certifique la validez de mi diploma

Criterios de aceptación:

1) Dada una solicitud de verificación, cuando haya transcurrido más de una semana de realizada y esta no haya sido aceptada o rechazada por la institución, entonces el egresado debe poder enviar un recordatorio a la institución académica. Se debe enviar un email a la institución conteniendo un vínculo al perfil del egresado.

2) Dada una solicitud de verificación, cuando no haya transcurrido más de una semana de realizada, entonces el egresado no debe poder enviar un recordatorio.

38. Ingreso con Google - 5 puntos

Descripción:

Como egresado
Quiero poder acceder a la aplicación usando mi usuario de google
Para agilizar el proceso de ingreso

Criterios de aceptación:

1) Dado un usuario, cuando se encuentra en la pantalla de ingreso, entonces se debe poder seleccionar la opción “Ingresar con Google”.

2) Dado un usuario, cuando selecciona dicha opción, entonces debe poder registrarse utilizando credenciales válidas de su cuenta de Google.

3) Dado un usuario que selecciona dicha opción, cuando un usuario ya está utilizando el email ingresado, entonces no se le permitirá registrarse utilizando esas credenciales.

4) Dado un usuario que se ha registrado con sus credenciales de Google, cuando intenta ingresar al sistema usando dichas credenciales, entonces el sistema debe permitir su ingreso.

39. Restricciones de campos - 8 puntos

Descripción:

Como institución académica
Quiero poder incluir restricciones en los requisitos de tipo “Campos” creados
Para poder prevenir errores de egresados

Criterios de aceptación:

1) Dada una institución académica, cuando cree requisitos de tipo campo, entonces debe poder seleccionar el tipo de campo del que se trata: numérico, de largo fijo, o fecha.

2) Dado el egresado, cuando ingrese datos para el campo, entonces la interfaz debe hacer cumplir las restricciones ingresadas por la institución académica.

12.13. Escenarios de requerimientos de atributos de calidad

Seguridad

Escenario Refinado 1: Interacción con *back-end* API - usuario autenticado y autorizado		
Escenario		Un usuario autenticado y autorizado realiza una petición a un endpoint de la API del *back-end*.
Objetivos de negocio		Seguridad: Disponibilidad
Atributos de Calidad Relevantes		Seguridad
Escenario	**Estímulo**	Petición a endpoint de la API
	Fuente de estímulo	Usuario autenticado y autorizado
	Ambiente	Tiempo de ejecución - operando normalmente
	Artefacto	Back End
	Respuesta	Se procesa la petición de forma adecuada
	Medida de Respuesta	Todo usuario autenticado y autorizado puede interactuar con el sistema
Preguntas		Existen distintos roles de usuarios?
Inconvenientes		

Tabla 26: escenario de requerimiento de seguridad 1

Escenario Refinado 2: Interacción con *back-end API* - usuario autenticado no autorizado		
Escenario		Un usuario autenticado y autorizado realiza una petición a un endpoint de la API del *back-end*.
Objetivos de negocio		Seguridad: Confidencialidad, Integridad
Atributos de Calidad Relevantes		Seguridad
Esce	**Estímulo**	Petición a endpoint de la API
	Fuente de estímulo	Usuario autenticado no autorizado
	Ambiente	Tiempo de ejecución - operando normalmente

nario	**Artefacto**	Back End
	Respuesta	No se procesa la petición, se retorna una respuesta de tipo *Unauthorized*.
	Medida de Respuesta	Todo usuario autenticado no autorizado no puede interactuar con el sistema
Preguntas		
Inconvenientes		

Tabla 27: escenario de requerimiento de seguridad 2

Escenario Refinado 3: Interacción con *back-end API* - usuario no autenticado		
Escenario		Un usuario no autenticado realiza una petición a un endpoint de la API del *back-end*.
Objetivos de negocio		Seguridad: Confidencialidad, Integridad
Atributos de Calidad Relevantes		Seguridad
Escenario	**Estímulo**	Petición a endpoint de la API
	Fuente de estímulo	Usuario no autenticado
	Ambiente	Tiempo de ejecución - operando normalmente
	Artefacto	Back End
	Respuesta	No se procesa la petición, se retorna una respuesta de tipo *Unauthorized*.
	Medida de Respuesta	Todo usuario no autenticado no puede interactuar con el sistema
Preguntas		
Inconvenientes		

Tabla 28: escenario de requerimiento de seguridad 3

Escenario Refinado 4: Modificación no autorizada de datos	
Escenario	Un usuario modifica los datos de una calificación de forma ilícita en la base de datos
Objetivos de negocio	Seguridad: Integridad
Atributos de Calidad	Seguridad

Relevantes		
Escenario	**Estímulo**	Modificación de datos no autorizada
	Fuente de estímulo	Usuario
	Ambiente	Tiempo de ejecución - operando normalmente
	Artefacto	Back End, Base de datos
	Respuesta	El sistema debe detectar una modificación no autorizada en la base de datos
	Medida de Respuesta	Toda inconsistencia en la base de datos debe ser detectada para asegurar la integridad y validez
Preguntas		Cómo implementar un mecanismo que detecte estas inconsistencias?
Inconvenientes		

Tabla 29: escenario de requerimiento de seguridad 4

Escenario Refinado 5: Comunicación entre front-end y back-end		
Escenario		Interacción entre *front-end* y *back-end* a través de la API del *back-end*
Objetivos de negocio		Seguridad: Confidencialidad, Integridad
Atributos de Calidad Relevantes		Seguridad
Escenario	**Estímulo**	*Front-end* le envía una petición al *back-end*
	Fuente de estímulo	*Front-end*
	Ambiente	Tiempo de ejecución - operando normalmente
	Artefacto	*Front-end, back-end*
	Respuesta	El *back-end* debe poder asegurar que el *front-end* fue el emisor de la petición, y que la misma no ha sido modificada por terceros, y vice-versa
	Medida de Respuesta	Debe existir un mecanismo que permita asegurar toda comunicación entre el *front-end* y el *back-end*.
Preguntas		

Inconvenientes	

Tabla 30: escenario de requerimiento de seguridad 5

Usabilidad

Escenario Refinado 1: Navegación		
Escenario		Un usuario navega por la plataforma web
Objetivos de negocio		Navegabilidad
Atributos de Calidad Relevantes		Usabilidad
Escenario	**Estímulo**	Usuario navega la interfaz
	Fuente de estímulo	Usuario
	Ambiente	Tiempo de ejecución - operando normalmente
	Artefacto	Interfaz de usuario
	Respuesta	El sistema debe permitir al usuario navegar con facilidad
	Medida de Respuesta	El usuario debe poder navegar hacia cualquier pantalla con una cantidad menor o igual a 3 clicks
Preguntas		
Inconvenientes		

Tabla 31: escenario de requerimiento de usabilidad 1

Escenario Refinado 2: Errores de usuario		
Escenario		Un usuario comete un error al utilizar la interfaz
Objetivos de negocio		Recuperación de errores
Atributos de Calidad Relevantes		Usabilidad
Escena	**Estímulo**	Operación en la interfaz de usuario que provoca un efecto no deseado
	Fuente de estímulo	Usuario
	Ambiente	Tiempo de ejecución - operando normalmente

r i o	**Artefacto**	Interfaz de usuario
	Respuesta	El sistema debe permitir al usuario deshacer la acción, recuperándose exitosamente de su error
	Medida de Respuesta	Toda acción realizada en la interfaz de usuario debe poder ser deshecha, sin tener consecuencias permanentes. Para cada resultado indeseado, se debe desplegar un mensaje de error descriptivo del problema.
Preguntas		
Inconvenientes		

Tabla 32: escenario de requerimiento de usabilidad 2

Escenario Refinado 3: Visualización de pantalla		
Escenario		Un usuario se encuentra en cualquier pantalla de la interfaz
Objetivos de negocio		Aumentar eficacia y eficiencia de la hora de utilizar el sistema.
Atributos de Calidad Relevantes		Usabilidad
E s c e n a r i o	**Estímulo**	Navegación hasta determinada pantalla
	Fuente de estímulo	Usuario
	Ambiente	Tiempo de ejecución - operando normalmente
	Artefacto	Interfaz de usuario
	Respuesta	Visibilidad de estatus, consistencia.
	Medida de Respuesta	En cualquier momento de la experiencia del usuario, se debe poder visualizar información acerca de la situación del usuario, manteniendo consistencia a través de la interfaz.
Preguntas		
Inconvenientes		

Tabla 33: escenario de requerimiento de usabilidad 3

Escenario Refinado 4: Estado de actividades

Escenario		Un usuario se encuentra realizando cualquier actividad en la UI, y no puede continuar, pero tampoco desea perder el progreso realizado hasta el momento.
Objetivos de negocio		Permitir al usuario pausar cualquier actividad en curso, almacenando el estado actual, para que pueda continuar en otro momento sin perder el progreso.
Atributos de Calidad Relevantes		Usabilidad
Escenario	**Estímulo**	Usuario no desea continuar con una actividad
	Fuente de estímulo	Usuario
	Ambiente	Tiempo de ejecución - operando normalmente
	Artefacto	Interfaz de usuario
	Respuesta	Posibilidad de almacenar estado actual para continuar con las actividades a posteriori
	Medida de Respuesta	En cualquier momento de la experiencia del usuario, se debe poder pausar la actividad de momento, almacenando el estado actual, para poder continuarla en otro momento.
Preguntas		
Inconvenientes		

Tabla 34: escenario de requerimiento de usabilidad 4

Modificabilidad

Escenario Refinado 1: Implementación de modificación de código		
Escenario		Se requiere la realización de una modificación en alguna porción del sistema
Objetivos de negocio		Implementación de cambios de forma ágil
Atributos de Calidad Relevantes		Modificabilidad
Esc	**Estímulo**	Solicitud de cambio en el código
	Fuente de estímulo	Desarrollador

e n a r i o	**Ambiente**	Tiempo de desarrollo
	Artefacto	Código
	Respuesta	La modificación se implementa sin efectos secundarios
	Medida de Respuesta	El impacto de cambio no se propaga al resto del sistema, quedando recluido en la porción del sistema en la cual se implementó
Preguntas		Qué patrón de arquitectura permite encapsular las distintas secciones del sistema disminuyendo el acoplamiento y el impacto de cambio?
Inconvenientes		

Tabla 35: escenario de requerimiento de modificabilidad 1

Escenario Refinado 2: Despliegue de modificación de código		
Escenario		Se realiza una modificación en alguna porción del sistema que luego debe ser desplegada
Objetivos de negocio		Despliegue de cambios de forma ágil
Atributos de Calidad Relevantes		Modificabilidad
E s c e n a r i o	**Estímulo**	Implementación de cambio en el código
	Fuente de estímulo	Desarrollador
	Ambiente	Tiempo de desarrollo
	Artefacto	Código
	Respuesta	La modificación se despliega sin efectos secundarios
	Medida de Respuesta	Todo cambio debe poder ser desplegado de forma independiente, sin necesidad de re-desplegar todo el sistema
Preguntas		Qué patrón de arquitectura permite modificar y desplegar distintas partes de forma independiente?
Inconvenientes		

Tabla 36: escenario de requerimiento de modificabilidad 2

12.14. Análisis de proveedor de servicios web

A pedido del cliente, se utilizará Amazon Web Services como proveedor de servicios web. Sin embargo, se realizó un análisis del mismo con el objetivo de determinar si es adecuado para el sistema:

Necesidad del Sistema	**Servicios de Amazon Web Services**
Ejecución de código del back-end	Amazon EC2: servicio web que provee capacidad de cómputo segura y redimensionable. Está diseñado para facilitar la escalabilidad, permitiendo configurar la capacidad de forma sencilla. Amazon Lambda: permite la ejecución de código sin la necesidad de gestionar servidores. Sólo se paga por el tiempo de cómputo consumido. Con Lambda, es posible ejecutar código de virtualmente cualquier aplicación, sin necesidad de administración. Sólo hace falta subir el código, y Lambda se encarga de todo lo necesario para ejecutar y escalar el código con alta disponibilidad. Incluso es posible configurar el código para ser ejecutado por otro servicio de AWS.
Almacenamiento de archivos	Amazon S3: provee un interfaz durable, altamente disponible, y segura, para almacenar y obtener data. Amazon S3

	también mantiene varias copias redundantes de la data.
Host para página web	Amazon S3: es posible hostear una página web en Amazon S3. Es tan simple como configurar un bucket para hosting de páginas web, y luego subir el código al mismo.
Envío de emails	Amazon SES: es un servicio de envío de emails basado en la nube, diseñado para facilitar a las aplicaciones el envío de mails de marketing, notificaciones, y transacciones, de forma confiable y costo-efectiva.
Almacenamiento de datos	Amazon RDS: facilita la configuración, el funcionamiento y el escalado de una base de datos relacional en la nube. Proporciona una capacidad costo-efectiva y redimensionable, mientras que automatiza tareas de administración que requieren mucho tiempo, como el aprovisionamiento de hardware, la configuración de la base de datos, la aplicación de parches y las copias de seguridad. Amazon DynamoDb: es una base de datos de documentos y clave-valor que ofrece un rendimiento de milisegundos a cualquier escala. Es una base de

	datos totalmente administrada y duradera con seguridad integrada, copia de seguridad y restauración, y almacenamiento en caché en memoria para aplicaciones a escala de Internet.

Tabla 37: análisis de servicios ofrecidos por *AWS*

Es posible concluir que Amazon Web Services satisface las necesidades de la aplicación, a veces ofreciendo varias alternativas para la misma necesidad.

12.15. Análisis de hosting para back-end: Amazon Lambda vs Amazon EC2

Por un lado, AWS Lambda es un recurso de cómputo de tipo *on-demand*, ofrecido como FaaS (*function-as-a-service*). Antes de la emergencia de soluciones ágiles como AWS Lambda, los equipos debían asignar recursos basado en pronósticos, asegurando que las demandas de memoria y cómputo no sobrepasaran los límites que sus sistemas soportaban.

Con servicios como AWS Lambda, los recursos de cómputo pueden escalar y disminuir automáticamente basándose en la demanda en tiempo real. AWS Lambda soporta varios lenguajes, tales como Node JS y Ruby.

Las arquitecturas construidas utilizando funciones como AWS Lambda son comúnmente llamadas arquitecturas *serverless*.

Por otro lado, Amazon EC2 (*Elastic Compute Cloud*) es un servicio de infraestructura virtual. Provee recursos de cómputo *on-demand* a través de los cuales se pueden crear poderosos servidores en la nube. Todo el hardware de EC2 es fragmentado en varios recursos que son ofrecidos en la forma de instancias, y que son escalables en términos de memoria de cómputo y poder de procesamiento.

También provee opciones flexibles para hostear la aplicación en varias plataformas con seguridad estrecha para arquitecturas *multi-tenant*. Estas instancias pueden ser accedidas a través de HTTP o HTTPS, lo cual permite a los desarrolladores crear aplicaciones similares a aquellas con infraestructura *on-premise*.

EC2 requiere el manejo y aprovisionamiento del ambiente. Cada instancia de EC2 ejecuta no sólo una copia de un sistema operativo, sino también una copia de todo el hardware que el sistema necesita para ejecutar. En contraste, AWS Lambda sólo requiere suficientes recursos de sistema y dependencias como para ejecutar un programa específico. Además, AWS Lambda permite crear bloques de código portables, facilitado el desarrollo, testeo, y despliegue.

	AWS Lambda	AWS EC2
Setup y gestión de ambiente	No se requiere mucho trabajo, y escala automáticamente.	Se requiere instalar y configurar el software requerido.
Seguridad	Se cuenta con un alto número de funciones, lo cual aumenta la superficie de ataque. Sin embargo, es menos probable que se violen las vulnerabilidades debido a que las funciones son *stateless*. Los agentes maliciosos tienden a crecer con el tiempo, lo cual no es posible en funciones sin estado.	Se debe atender a la capa de seguridad a nivel de instancia. Cada instancia puede tener varias capas de seguridad que controla el tráfico a través de distintos protocolos como TCP, UDP, etc. Crear políticas es un trabajo cansador que requiere ensayo y error.
Timeout	Presenta un timeout de 15 minutos, lo cual limita el tipo de tareas que puede realizar. No es adecuado para largas y complejas.	Presenta opciones más flexibles. Se puede trabajar con tareas largas.
Dependencias	Pueden ser un problema debido al tamaño límite de paquete de 50 MB.	No hay restricciones en cuanto al tamaño de almacenamiento.
Escalabilidad	Es uno de los mayores beneficios de AWS Lambda, ya que es automática y se hace	Debe ser configurada manualmente, pero es totalmente regulable.

	en respuesta al tráfico entrante. Maneja hasta 3000 ejecuciones concurrentes dependiendo de la región. Sin embargo, no es configurable y depende de la escalabilidad de EC2.	
Disponibilidad	On-Demand. Están disponibles todo el tiempo. Son levantados y bajados dependiendo de la demanda. Como no se debe de pagar el tiempo mientras, se ahorra en costos.	Siempre disponibles. Aunque se puede escalar la cantidad de instancias, siempre debe haber por lo menos una levantada en todo momento.
Latencia	Presentan *cold starts*, los cuales ocurren cuando se ejecuta una función que lleva un tiempo considerable estando inactiva. Es el tiempo que se lleva en construir el contenedor dentro del cual se ejecutará la función.	Generalmente, no ocurren *cold starts*, a menos que se esté creando una nueva instancia.
Costo	Se paga únicamente por el tiempo de ejecución y memoria consumida.	El precio es mayor debido a que siempre hay al menos una instancia ejecutándose. En casos de arquitecturas escalables, los costos son aún mayores.

Tabla 38: análisis comparativo entre *AWS Lambda* y EC2 [7]

Teniendo en cuenta la comparación previa, se decidió optar por AWS Lambda. Esto se debe al ahorro en no sólo en costos, sino también en tiempo que se tendría que invertir en la configuración de un EC2. AWS Lambda parecen ser la elección apropiada para nuestro caso de uso, ya que además de las ventajas de escalabilidad y disponibilidad que presenta, muchas de las desventajas de AWS Lambda no aplican al sistema. En primer lugar, no se requieren funciones de larga ejecución, por lo que el timeout no es un problema. Además, tampoco se incluirán dependencias pesadas. Finalmente, la performance no es un aspecto crítico del sistema, por lo que los *cold starts* no presentan un problema significativo.

12.16. Análisis de tecnologías para back-end

Como posibilidades para el desarrollo del back-end se analizaron las alternativas que se detallan a continuación:

Node JS

Node JS es un entorno de ejecución javascript que permite la ejecución de código javascript del lado del servidor.

Entre las ventajas de Node JS se encuentran las siguientes:

- **Escalabilidad**: Node JS es considerado como la mejor opción para las arquitecturas de microservicios, debido a que es una tecnología *ligthweight*.
- **Facilidad de Aprendizaje:** JavaScript es uno de los lenguajes de programación más usados entre desarrolladores, por lo que facilita su aprendizaje.
- **Mayor performance:** Node JS ejecuta el código JavaScript utilizando el motor *Google V8* que compila el código JavaScript directamente a código de máquina.
- **Comunidad grande y activa:** Cuenta con *NPM*, un manejador de paquetes por defecto, que también sirve como mercado para cientos de miles de librerías de código abierto gratuitas.
- **Manejo de concurrencia:** Node JS permite el procesado de varias peticiones de forma simultánea. Puede manejar peticiones concurrentes de mejor forma que otros lenguajes como Ruby on Rails y Python.
- **Soporte Extensivo:** Los desarrolladores de Node JS obtienen soporte de varias herramientas. Por ejemplo, es posible probar el código fuente de Node JS utilizando herramientas tales como Jasmin.

Desventajas de Node JS:

- **Menos eficiente con operaciones intensivas para el CPU:** Los modelos asincrónicos de Node JS tienen ciertas limitaciones. Mientras Node JS es apropiado para manejar múltiples eventos y peticiones concurrentes, no es tan eficiente cuando de operaciones intensivas para el CPU se trata, como por ejemplo, generar gráficas o redimensionar imágenes.
- **Inmadurez de herramientas:** a pesar de que los módulos centrales de Node JS son estables y pueden ser considerado maduros, existen varias herramientas en el registro NPM que son de baja calidad o no están documentadas o testeadas adecuadamente. [8]

Ruby on Rails

Ruby on Rails (RoR) es un *framework* web popular utilizado para el desarrollo de aplicaciones web construido sobre el lenguaje de programación Ruby.

Entre las ventajas de RoR se encuentran las siguientes:

- **Mejores estándares de la industria:** RoR es un *opinionated framework*, lo cual quiere decir que guía a los usuarios a hacer las cosas de cierta manera. De esta forma, promueve los mejores estándares de desarrollo web.
- **Velocidad de desarrollo:** RoR fue creado con la idea de prototipado y desarrollo veloz. Es un sistema bien desarrollado de módulos, scripts generadores, y un manejador de paquetes eficiente, permitiendo crear aplicaciones complejas con unos pocos comandos.
- **Comunidad:** Rails es un *framework* de código abierto, soportado por una gran comunidad. El ecosistema de RoR contiene una gran variedad de gemas (equivalentes a los paquetes NPM).

Entre las desventajas de RoR se encuentran las siguientes:

- **Velocidad de ejecución y performance:** Uno de los argumentos más frecuentes en contra de RoR es su baja velocidad de ejecución, lo cual dificulta escalar las aplicaciones RoR.

- **Baja flexibilidad:** RoR es un *opinionated framework* con muchas dependencias incluidas. Al iniciar un proyecto, se debe configurar el ruteo, migraciones de base de datos, y otros módulos que vienen incluidos con el *framework*. Estos módulos por defecto son útiles si se quiere crear una aplicación con funcionalidades estándar, pero pueden ser un obstáculo si se quiere desarrollar algo más personalizado.
- **Alto costo de malas decisiones durante el desarrollo:** Las malas decisiones arquitectónicas durante etapas iniciales del proyecto pueden costar más en Rails que en cualquier otro *framework* [9].

Análisis comparativo

	Node JS		Ruby on Rails	
Adaptabilidad a la Solución a Implementar.	Alto	3	Medio	2
Comunidad	Alto	3	Alto	3
Experiencia del Equipo	Alto	3	Bajo	1
Librerías de Utilidad	Alto	3	Alto	3
Curva de Aprendizaje	Medio	2	Bajo	1
Total	14		10	

Tabla 39: análisis comparativo entre *Node JS* y *Ruby on Rails*

12.17. Análisis de base de datos: SQL vs NoSQL

A continuación, se realiza un análisis comparativo de las dos tecnologías.

SQL	NoSQL
Relacional.	No relacional.

Almacena data en tablas.	Almacena data en documentos JSON, pares clave-valor, o grafos.
Útil para soluciones en las cuales toda la data tiene la misma estructura.	Ofrece flexibilidad ya que no todos los registros tiene que tener la misma estructura.
Las relaciones usualmente se capturan usando un modelo normalizado, y realizando *joins* para resolver referencias a través de distintas tablas.	Las relaciones usualmente son representadas almacenando toda la data relativa a un objeto dentro del mismo registro.

Tabla 40: análisis comparativo entre *SQL* y *NoSQL* [10]

Teniendo en cuenta las diferencias entre SQL y NoSQL, se debe además considerar algunas características del proyecto. En primer lugar, inicialmente no se contó con un modelo de datos definido, como para determinar el esquema de una base de datos relacional. Además, se deben tener en mente los requerimientos de modificabilidad del sistema, ya que al tratarse de un MVP, es altamente probable que se deban realizar cambios más adelante. NoSQL ofrece cierta versatilidad para introducir cambios, mientras que la dificultad es mayor en SQL. Teniendo en cuenta estos factores, se decidió por una base de datos NoSQL.

12.18. Análisis de tecnologías para front-end

Como posibilidades para el desarrollo del front-end se analizaron las alternativas que se detallan a continuación:

React JS

React JS es una librería JavaScript que combina la velocidad de JavaScript y utiliza una nueva forma de renderizar páginas web, haciéndolas altamente dinámicas y responsivas a los input de los usuarios. Fue introducida por Facebook en el 2011.

Entre las ventajas de React JS se encuentran las siguientes:

- **Utilización de DOM Virtual mejora la experiencia de usuario y facilita el trabajo del desarrollador:** El DOM (Document Object Model) es una estructura lógica de documentos en HTML, XHTML, o formatos XML, con estructura de árbol. Los navegadores web utilizan motores para transformar el HTML a DOM. La principal preocupación acerca del DOM tradicional es la forma en que procesa los cambios, como entradas de usuarios, consultas, etc. Para responder a cada cambio, se requiere actualizar los árboles DOM, los cuales son bastante grandes hoy en día, conteniendo hasta miles de elementos. React logra incrementar la velocidad de las actualizaciones utilizando un DOM virtual. Esto hace que los actualizaciones se hagan más velozmente, permitiendo la construcción de una interfaz altamente dinámica.
- **Permite reutilización de componentes:** Esto ahorra tiempo de forma significativa. Todos los componentes en React JS están aislados, lo cual permite su reuso.
- **Flujo de data en una dirección provee código más estable:** React JS permite trabajar directamente con componentes y usa *data-binding* en dirección descendente, para asegurar que cambios en estructuras hijas no afecten a sus padres. Esto hace que el código sea más estable.

- **Librería de código abierto de Facebook, en desarrollo constante y de comunidad abierta**

- **Redux: un contenedor de estado conveniente.** Aunque también esté presente en Angular, es más usado en React, y las probabilidades de encontrarlo en un proyecto de React son mucho más altas que en los de Angular. También hay más material disponible para enfrentar la curva de aprendizaje de React Redux. Redux simplifica el manejo y almacenado del estado de componentes en grandes aplicaciones con muchos elementos dinámicos. Redux almacena el estado de la aplicación de forma tal que todos los componentes dentro de la aplicación pueden accederlo.

- **Amplio espectro de herramientas para React y Redux:** Tanto React como Redux vienen con un set de herramientas decentes que simplifican el trabajo de los desarrolladores.

Entre las desventajas de React JS se encuentran las siguientes:

- **Alto ritmo de desarrollo:** el ambiente está constantemente cambiando, y los desarrolladores deben re-aprender las nuevas formas de hacer las cosas.

- **Documentación pobre:** algunos miembros de la comunidad de React piensan que las tecnologías de React se actualizan a un ritmo tan acelerado que no hay tiempo para escribir instrucciones adecuadas.

- **Barrera de JSX:** React utiliza JSX, una extensión de sintaxis que permite combinar HTML con JavaScript. Esto tiene sus propios beneficios, pero puede presentar una curva de aprendizaje bastante empinada [11].

Angular JS

Angular es una plataforma de software de código abierto utilizada para construir interfaces. Forma parte del ecosistema JavaScript, y fue introducida por Google en el 2009.

Entre las ventajas de Angular JS se encuentran las siguientes:

- **Enlace de datos bidireccional:** Angular JS fue construido con la arquitectura *Model-View-Controller*. El *framework* sincroniza al Modelo con la Vista, y cada vez que el modelo cambia, también lo hace la vista. Esto ahorra tiempo a los desarrolladores que no deben invertir tiempo en sincronizar la vista con el modelo.

- **Directivas:** Esta funcionalidad permite a los desarrolladores asignar comportamiento especial al DOM, creando contenido dinámico con HTML.

- **Inyección de dependencias:** Las dependencias definen cómo las distintas piezas de código interactúan, y cómo los cambios en un componente impactan en otros. Con Angular JS, se pueden utilizar inyectores que definen las dependencias como elementos externos, permitiendo a los componentes desacoplarse de sus dependencias.

- **Comunidad:** Angular JS cuenta con una fuerte comunidad que provee material de entrenamiento, discusiones, y herramientas.

Entre las desventajas de Angular JS se encuentran las siguientes:

- **Performance**: las aplicaciones dinámicas no siempre funcionan tan bien. Las SPAs (*Single Page Applications*) complejas pueden resultar inconvenientes de usar debido a su tamaño.

- **Curva de Aprendizaje empinada:** a pesar de ser un instrumento versátil, Angular JS presenta siempre más de una forma de completar cualquier tarea. Esto genera cierta confusión [12]

Análisis comparativo

	React JS		Angular JS	
Adaptabilidad a la Solución a Implementar.	Alto	3	Medio	2
Comunidad y Soporte	Alto	3	Alto	3
Experiencia del Equipo	Alto	3	Medio	2
Librerías de Utilidad	Alto	3	Alto	3
Curva de Aprendizaje	Medio	2	Medio	2
Total	14		12	

Tabla 41: análisis comparativo entre *React JS* y *Angular JS*

12.19. Configuración de nodos de una blockchain privada

Configuración con Fat Jar en computadora local

En primer lugar, se debe verificar que el equipo cumple con los siguientes requerimientos:

- 2 cores
- 8 GB RAM
- 2 GB almacenamiento
- OS x64

También es necesario instalar Java 8 JDK. Este se puede descargar desde la siguiente URL: https://www.java.com/download/

Luego, es necesario descargar el Fat JAR o Uber JAR. Es posible hacerlo desde la siguiente URL: https://github.com/rsksmart/rskj/releases. Para ello se debe descargar el archivo de extensión **.jar**

Para configurar una red privada seguir los siguientes pasos:

1. Extraer el contenido del jar con un programa extractor de archivos
2. Dentro de la carpeta extraída dirigirse a la carpeta *config*
3. Crear un nuevo archivo <NOMBRE_DE_RED>.conf (Ej: *private.conf*). Completar con el siguiente contenido

```
blockchain.config {
    name = private
    hardforkActivationHeights = {
        bahamas = 0,
```

```
        afterBridgeSync = -1,
        orchid = 0,
        orchid060 = 0,
        wasabi100 = 0,
        twoToThree = 0
    },
    consensusRules = {
        rskip97 = -1 # disable orchid difficulty drop
    }
}

peer {

        discovery = {
                # if peer discovery is off
                # the peer window will show
                # only what retrieved by active
                # peer [true/false]
                enabled = false

                # List of the peers to start
                # the search of the online peers
                # values: [ip:port]
                ip.list = [
                     <LISTA DE DIRECCIONES IP PÚBLICAS DE LOS
NODOS DE LA RED PRIVADA>
                ]
        }
```

```
    # Port for server to listen for incoming connections
    port = 5050
    # Network id
    networkId = <IDENTIFICADOR DE LA RED>
}
miner {
    client {
        delayBetweenBlocks = 20 seconds
    }
}
genesis = devnet-genesis.json
# hello phrase will be included in the hello
hello.phrase = <NOMBRE DE LA RED A INCLUIR EN MENSAJES DE GREET>
database {
    # place to save physical storage files
    dir = ${user.home}/.rsk/privatenet/database
    reset = true
}
wallet {
    enabled = true
    accounts = []
}
```

Tabla 42: contenido de archivo de configuración para red privada

El identificador de la red es un número único que identifica a la red, determinado por los usuarios de la red. Es importante que la configuración sea la misma en todos los nodos de la red.

4. Abrir el archivo *reference.conf* ubicado en la raíz del directorio extraído

5. Setear los valores de *miner.server.enabled* y *miner.client.enabled* en *true*

```
miner {
    # The default gas price
    minGasPrice = 0

    server {
        enabled = true
        isFixedClock = false
    }

    client {
        enabled = true

        # If autoMine is true, it will set delayBetweenBlocks in 0
        # otherwise it will use what is define in delayBetweenBlocks setting.
        autoMine = false

        # The time the miner client will wait after mining a block and before mining the next one.
        # This allows mining a sane number of blocks per minute when the difficulty is low (e.g. for regtest).
        delayBetweenBlocks = 0 seconds

        # The time the miner client will wait to refresh the current work from the miner server
        delayBetweenRefreshes = 1 second
    }
}
```

Imagen 85: configuración de archivo *reference.conf*

6. Reconstruir el archivo jar. Para hacerlo, se puede utilizar el programa extractor de archivos, y luego cambiarle la extensión al archivo resultante a *.jar*

Para ejecutar el nodo, se debe correr el siguiente comando en la consola o terminal:

```
$ java -cp <DIRECCION-A-RSKJ-JAR> co.rsk.Start
--<NOMBRE-DE-RED>
```

Reemplazar <DIRECCION-A-RSKJ-JAR> con la dirección del archivo JAR. A modo de ejemplo: C:/RskjCode/rskj-core-1.1.0-WASABI-all.jar.

Si no se ve salida, significa que el nodo se está ejecutando. Para verificarlo, se puede abrir una nueva pestaña de consola o terminal (es importante que no cierre

esta o interrumpa el proceso) y enviar una solicitud al servidor RPC HTTP del nodo. Este es un ejemplo usando cURL:

```
$ curl localhost:4444/1.1.0/ -X POST -H "Content-Type:
application/json" --data
'{"jsonrpc":"2.0","method":"eth_blockNumber","params":[],"id":
1}'
```

La respuesta debería ser similar a lo siguiente:

```
{"jsonrpc":"2.0","id":1,"result":"0xfc0"},
```

En donde la propiedad `result` es el número del último bloque que se ha sincronizado (en hexadecimal).

12.20. Sprint retrospectives con Retrium

La herramienta Retrium para retrospectivas en línea es una herramienta que hace que las retrospectivas para equipos distribuidos sean fáciles y efectivas. El equipo puede usar Retrium para reflexionar, aprender y mejorar continuamente. La herramienta Retrium proporciona diferentes ejercicios que los equipos ágiles pueden usar para hacer retrospectivas, como *4Ls*, *Mad Sad Glad* y *Start Stop Continue*.

A continuación, se incluyen algunas capturas a modo de ejemplo para visualizar las distintas formas en que se llevaron adelante estas reuniones.

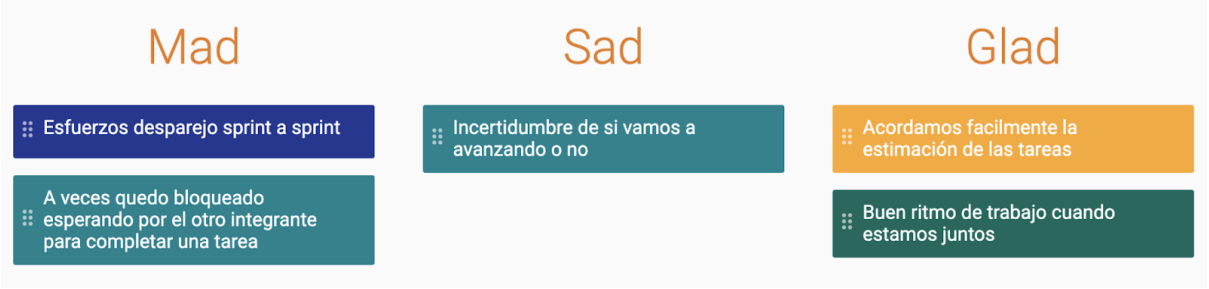


Imagen 86: retrospectiva con Retrium

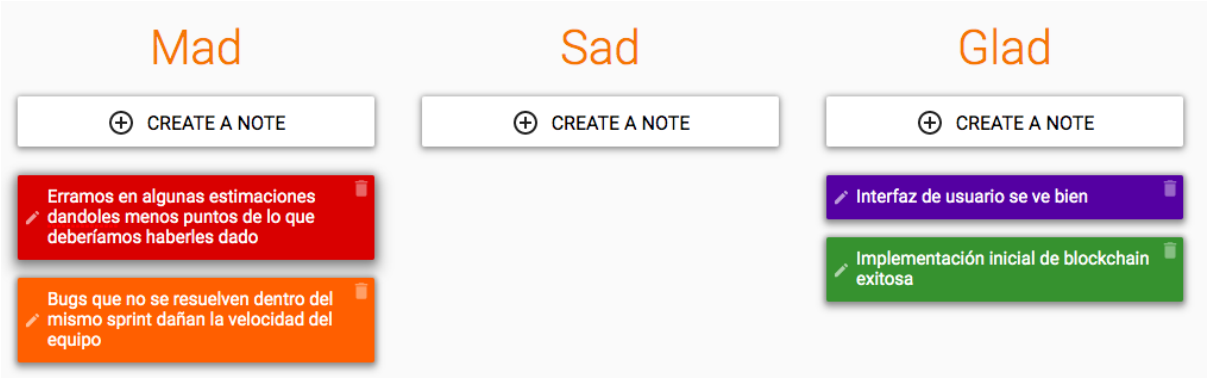


Imagen 87: retrospectiva con Retrium

Imagen 88: retrospectiva con Retrium

Imagen 89: retrospectiva con Retrium

Una vez elegida la forma en que va a llevarse adelante la reunión, Retrium permite asignar un período de tiempo para que cada uno de los integrantes escriba sus tópicos. Mientras esto ocurre, no es posible ver lo que están escribiendo los demás.

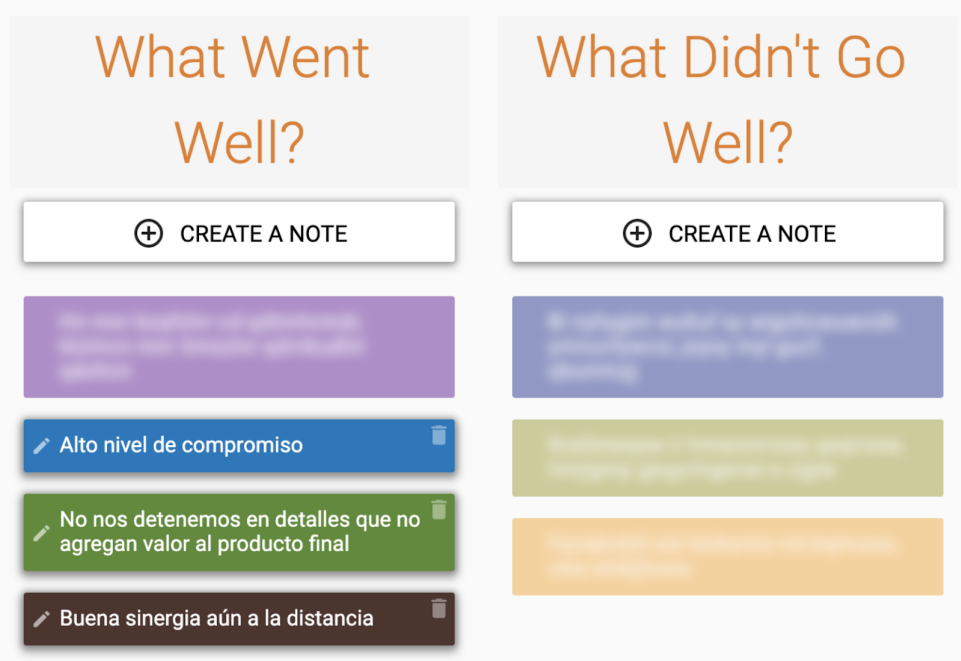


Imagen 90: retrospectiva con Retrium

Una vez terminado el tiempo, se muestran todos los tópicos escritos y se le asignan 3 votos a cada uno de los integrantes para que voten sobre cuales les gustaría charlar.

Imagen 91: retrospectiva con Retrium

A continuación, se asignan 5 minutos para charlar sobre cada uno de los 3 temas más votados.

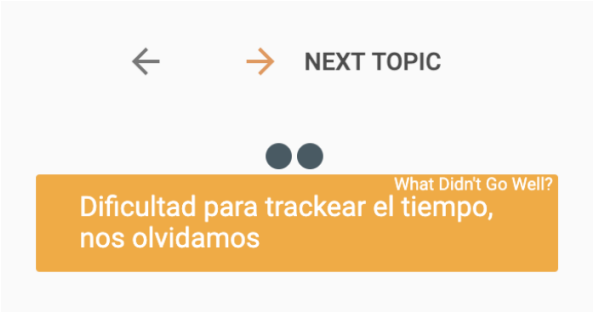


Imagen 92: retrospectiva con Retrium

Finalmente se escriben las acciones a llevar a cabo para corregir lo que no se ha hecho bien o podría mejorar, que serán revisadas al comienzo de la próxima reunión retrospectiva.

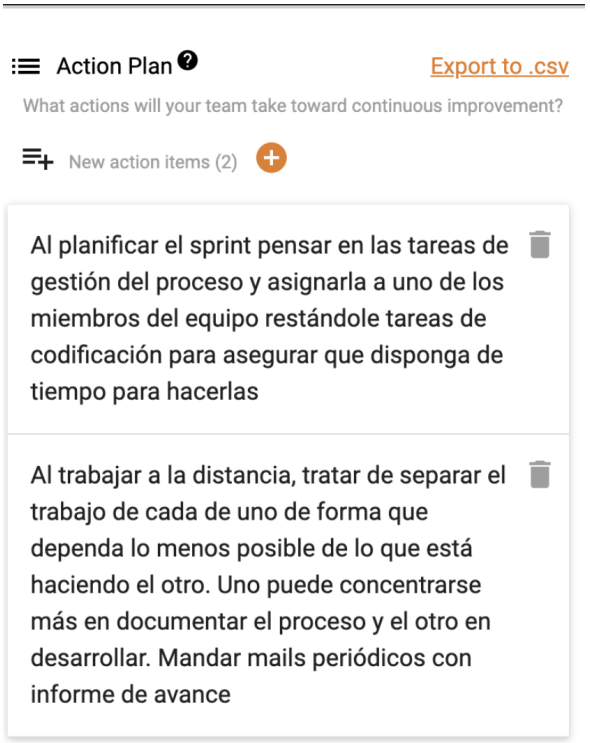


Imagen 93: retrospectiva con Retrium

12.21. Pruebas con usuarios

A continuación, se detalla la planificación del proceso de pruebas con usuarios. El objetivo de este plan fue contar con un proceso estandarizado para ejecutar pruebas con usuarios, y de este modo obtener resultados comparables de los cuales fue posible extraer estadísticas.

Cómo y cuándo

De modo de facilitar las pruebas con usuarios, el lugar de la prueba varió con el usuario. En algunas ocasiones fueron los integrantes del equipo quienes se trasladaron hasta las casas u oficinas de los usuarios, y en otras fue al revés. Siempre se eligieron horarios que se adapten a las conveniencias de los usuarios.

Alcance

El alcance de las pruebas varió con el tipo de usuario y la etapa en la cual se realizaron. Como fue detallado en el capítulo de Ingeniería de requerimientos, existieron tres *releases*, y se realizaron pruebas luego de cada uno de ellos.

En el caso de los estudiantes/egresados, se hicieron pruebas luego del segundo y el tercer *release*. Para las pruebas realizadas luego del segundo *release*, el alcance incluyó las siguientes tareas:

1. Registro en página web.
2. Inicio de sesión.
3. Subir imagen de perfil.
4. Editar datos personales.
5. Seleccionar institución educativa.
6. Ingresar datos para requisitos de tipo campo para institución educativa.

7. Ingresar datos para requisitos de tipo archivo para institución educativa.
8. Seleccionar título.
9. Ingresar información para título.
10. Subir archivo asociado a título.
11. Solicitar verificación por parte de institución académica.

Para las pruebas realizadas luego del tercer *release*, al alcance previamente definido se le sumaron las siguientes tareas:

12. Ver vista previa de perfil.
13. Compartir perfil.

En el caso de las instituciones académicas, se hicieron pruebas luego del primer y el tercer *release*. Para las pruebas realizadas luego del primer *release*, el alcance incluyó las siguientes tareas:

1. Registro en página web.
2. Inicio de sesión.
3. Subir imagen de perfil.
4. Editar datos de institución.
5. Ingresar títulos.
6. Ingresar requisitos de tipo campo para institución educativa.
7. Ingresar requisitos de tipo archivo para institución educativa.

Para las pruebas realizadas luego del tercer *release*, al alcance previamente definido se le sumaron las siguientes tareas:

8. Cargar títulos con archivo CSV.

9. Cargar archivo asociado a requerimiento definido.
10. Ingresar títulos indicando tipo de título.
11. Ver lista de solicitudes de verificación de calificaciones.
12. Seleccionar solicitud pendiente.
13. Ver perfil de egresado.
14. Descargar archivos de requisitos.
15. Descargar archivos de calificaciones.
16. Aceptar, rechazar, o solicitar cambio en solicitud de verificación.

En el caso de los empleadores, se hicieron pruebas únicamente luego del tercer *release*, debido a que las funcionalidades que les correspondían fueron de las últimas en desarrollarse. El alcance de las pruebas incluyó las siguientes tareas:

1. Ingresar al sistema con vínculo generado por egresado.
2. Visualizar perfil y calificaciones.
3. Descargar archivos asociados a calificaciones.

En todos los casos se creó un guión de forma tal de que el participante de la prueba realice las tareas en el orden establecido previamente.

Tiempo de cada prueba

Se estableció un tiempo máximo de 30 minutos por prueba. Esto fue decidido para evitar que los participantes se cansen, lo que generalmente conduce a una disminución en la calidad del feedback.

Ejecución de las pruebas

En lugar de ir directo a los escenarios de prueba, previo a comenzar con las pruebas se definió realizar un breve preámbulo, explicando exactamente lo que se

estaba tratando de lograr a través de las pruebas. De esta forma, se garantizó darle al participante una idea clara de lo que se esperaba de él, aliviando la presión y facilitando la prueba. Se puso especial énfasis en que se estaba probando al software, y no al participante, por lo que no había respuestas o acciones incorrectas.

Durante las pruebas, no se interrumpió el flujo de los participantes de la prueba ofreciéndoles orientación y asesoramiento. Sin embargo cada vez que finalizaban una tarea, se le pidió a los participante que realicen comentarios acerca de las acciones que estaban realizando. También se tomaron apuntes de la finalización o no de las tareas por parte de los participantes, tomando nota además de los comentarios realizados [13]. En el caso de las pruebas con alumnos y egresados, al tener acceso a una mayor muestra, se utilizó un template para registrar el resultado de cada prueba. De esta forma, se pudo comparar y extraer datos de los resultados de las pruebas con los egresados de forma más sencilla y subjetiva, se utilizó el siguiente template para registrar dichos resultados. El mismo puede ser visto a continuación

	Se completó	
Tarea	Sí	No
Registro en página web		
Comentarios:		
Inicio de sesión		
Comentarios:		
Subir imagen de perfil		
Comentarios:		
Editar datos personales		
Comentarios:		
Seleccionar institución educativa		

Comentarios:		
Ingresar datos para requisitos de tipo campo para institución educativa		
Comentarios:		
Ingresar datos para requisitos de tipo archivo para institución educativa		
Comentarios:		
Seleccionar título		
Comentarios:		
Ingresar información para título		
Comentarios:		
Subir archivo asociado a título		
Comentarios:		
Solicitar verificación por parte de institución académica		
Comentarios:		
Ver vista previa de perfil		
Comentarios:		
Compartir perfil		
Comentarios:		

Tabla 43: template usado para registrar el resultado de cada prueba.

12.22. Encuestas de satisfacción

A continuación, se puede observar la encuesta de satisfacción enviada a los usuarios que participaron en las pruebas realizadas luego del tercer release. La

misma tiene como fin además medir la seguridad y usabilidad percibida por ellos. Se presentaron un conjunto de afirmaciones y se le pidió a los usuarios que indiquen qué tan de acuerdo estaban con ellas.

Finalmente, se les pidió dar un puntaje a la aplicación, siendo 1 el más bajo y 5 el más alto.

Imagen 94: encuesta de satisfacción enviada a usuarios

12.23. Análisis según heurísticas de Nielsen

Con el objetivo de evaluar la usabilidad de la aplicación web desarrollada, se realizó un análisis utilizando las heurísticas de Nielsen [14]. El análisis fue de utilidad tanto

para identificar aquello que iba de acuerdo con las heurísticas de Nielsen, como también para corregir aquellas discrepancias.

Visibilidad del estado del sistema

“El sistema siempre debe mantener a los usuarios informados sobre lo que está sucediendo, a través de feedback apropiado dentro de un tiempo razonable.”

Se intentó siempre proveer mensajes de error o de confirmación en todas las situaciones. Algunos ejemplos son:

- Al confirmar una cuenta (cuenta ya confirmada).
- Al registrar una institución (registro exitoso).
- Al registrarse un egresado (utilizando un email ya utilizado por otro usuario).

Imagen 95: registro de egresado usando *email* ya usado por otra cuenta

Imagen 96: registro de institución académica exitoso

Imagen 97: intento de confirmación de cuenta ya confirmada

Coincidencia entre el sistema y el mundo real.

"El sistema debe hablar el idioma de los usuarios, con palabras, frases y conceptos familiares para el usuario, en lugar de términos orientados al sistema."

Al momento de verificar esta heurística, se encontraron varios textos en inglés, los cuales debieron ser traducidos para ser compatibles con el idioma de los usuarios. A modo de ejemplo, se reemplazó el texto "Log Out" con "Cerrar Sesión"

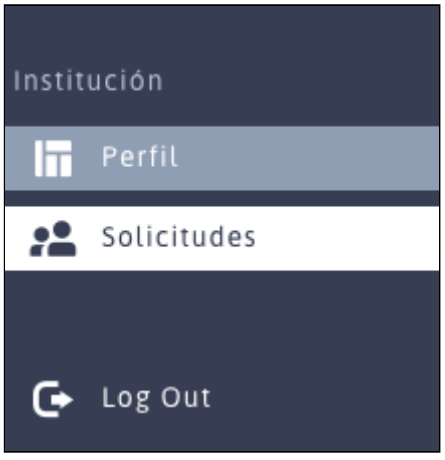


Imagen 98: texto de cierre de sesión en inglés

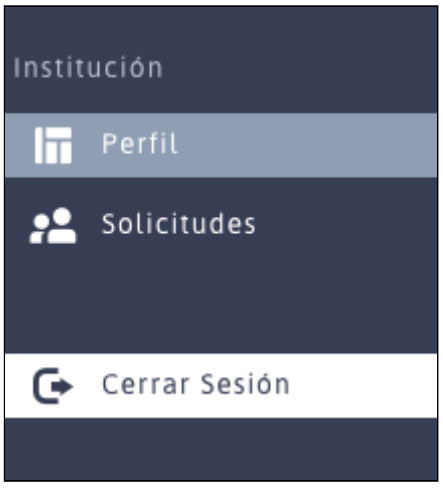


Imagen 99: texto de cierre de sesión en español

También, al momento de registrarse, se intentaron expresar los términos y condiciones en un lenguaje entendible por los usuarios, utilizando la menor cantidad de términos técnicos posible.

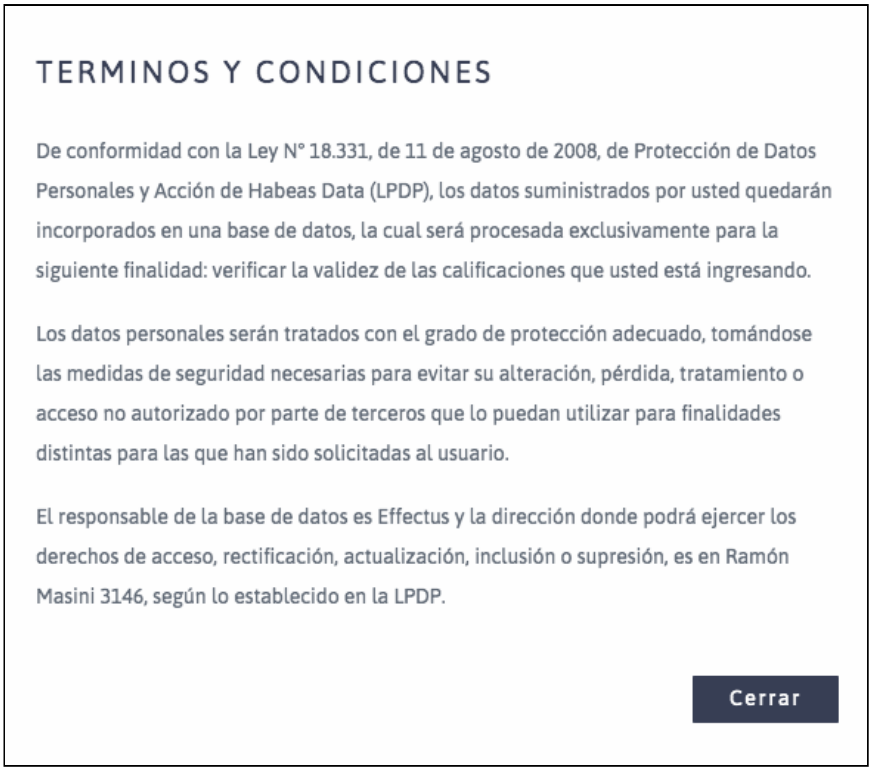


Imagen 100: términos y condiciones

Control del usuario y libertad

“Los usuarios a menudo eligen las funciones del sistema por error y necesitarán una ‘salida de emergencia’ claramente marcada para salir del estado no deseado sin tener que pasar por un diálogo extendido. Soporte para deshacer y rehacer.”

Se intentó siempre ofrecer opciones para que el usuario deshaga acciones o cancele cambios hechos sin intención. Algunos ejemplos son:

- Perfil de institución académica o egresado: se ofrece la opción de cancelar, lo cual revierte los cambios a la última vez que se ha guardado.
- Se ofrece al egresado deshacer la acción al solicitar la verificación por parte de la institución académica.
- Se ofrece a la institución académica deshacer la acción una vez que se verifica o rechaza una solicitud de verificación de un egresado.

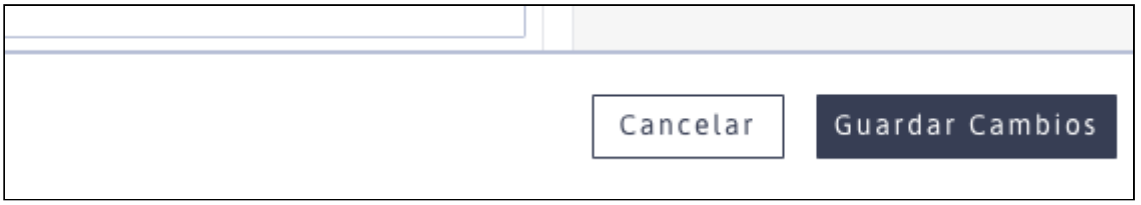


Imagen 101: opción de cancelar en perfil de egresado

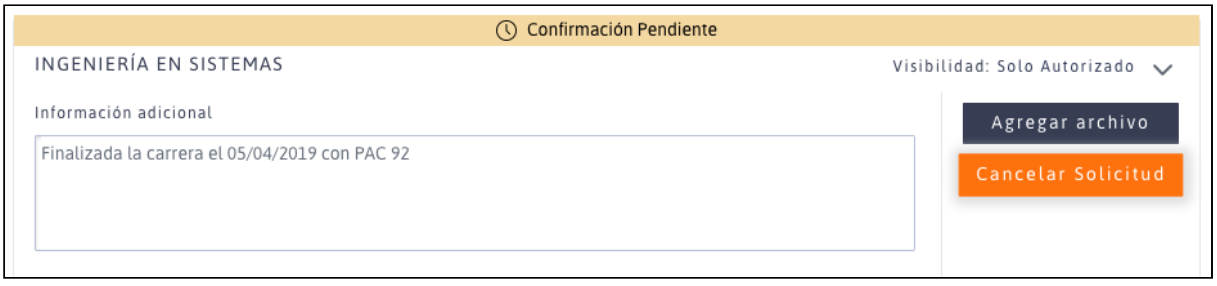


Imagen 102: opción de cancelar solicitud de verificación en perfil de egresado

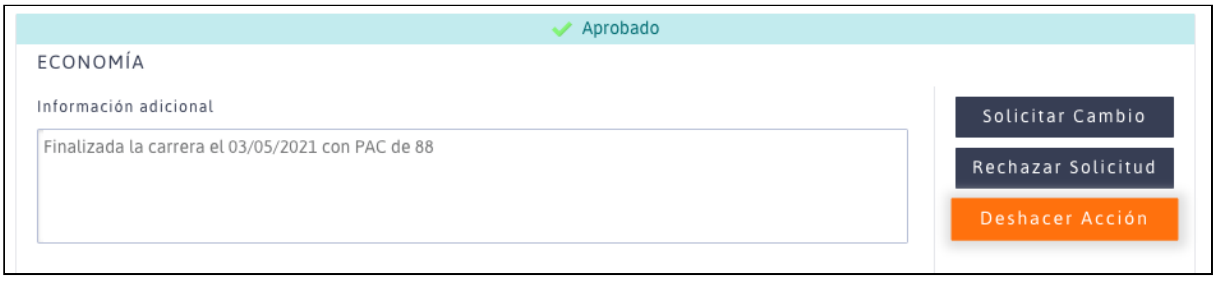


Imagen 103: opción de cancelar aprobación de calificación

Consistencia y estándares

“Los usuarios no deberían tener que preguntarse si diferentes palabras, situaciones o acciones significan lo mismo.”

Se intentó ser consistentes en la mayor medida posible. A modo de ejemplo, se intentó que todos los botones tengan la misma apariencia, y que todos los botones de “Cancelar” sean iguales.

Imagen 104: perfil de institución académica - vista de administrador

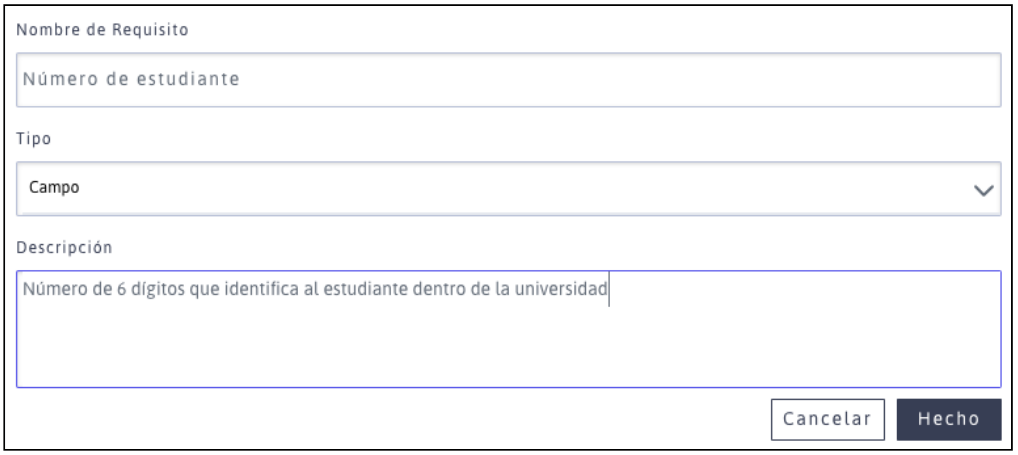


Imagen 105: ingreso de requisito

En el caso de ingreso de títulos por parte de institución académica, se debió modificar la apariencia de los botones “Cancelar” y “Hecho” para que sea consistente con el resto de los botones en la aplicación.

Antes:

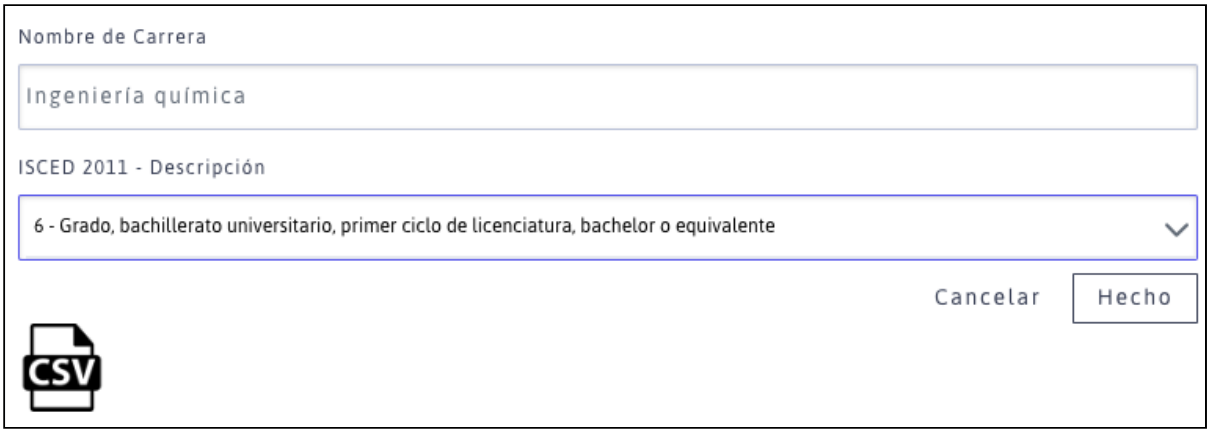


Imagen 106: ingreso de título previo a modificar apariencia de botones

Después:

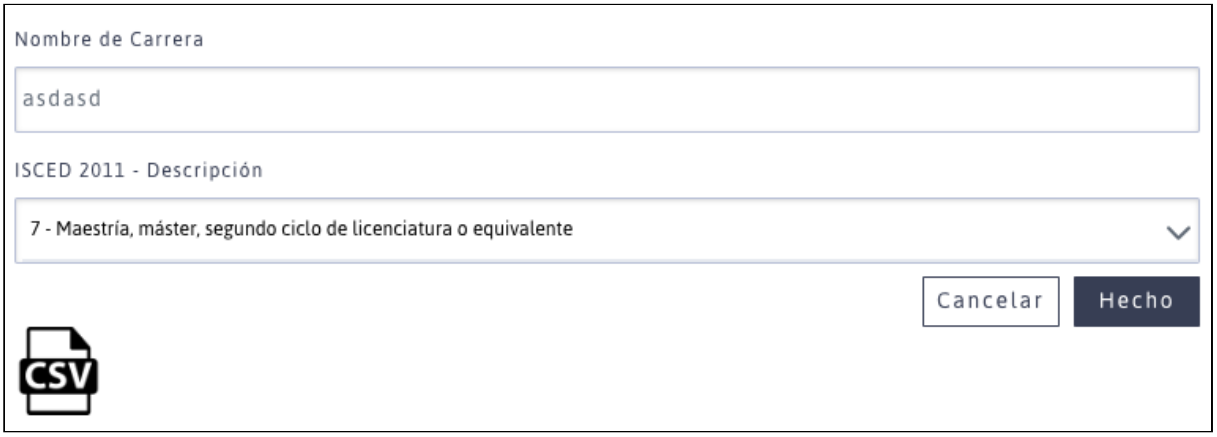


Imagen 107: ingreso de título luego de modificar apariencia de botones

Finalmente, se modificó el siguiente texto de inicio de sesión, para que trate al usuario de “usted”, y de esta forma ser consistente con el resto de la aplicación.

Antes:

Imagen 108: inicio de sesión previo a modificar texto

Después:

Imagen 109: inicio de sesión luego de modificar texto

Prevención de errores

“Incluso mejor que buenos mensajes de error es un diseño cuidadoso que evita que ocurran problemas en primer lugar. Se deben eliminar las condiciones propensas a errores o verificar y presentar a los usuarios una opción de confirmación antes de comprometerse con la acción.”

Para ir de acuerdo con esta heurística, se incluyeron mensajes de confirmación en acciones relevantes. Algunos ejemplos son:

- Al cancelar los cambios no guardados en el perfil de egresado o institución académica.
- Al eliminar un requisito por parte de una institución académica.
- Al solicitar el egresado la verificación de una calificación por parte de una institución académica.

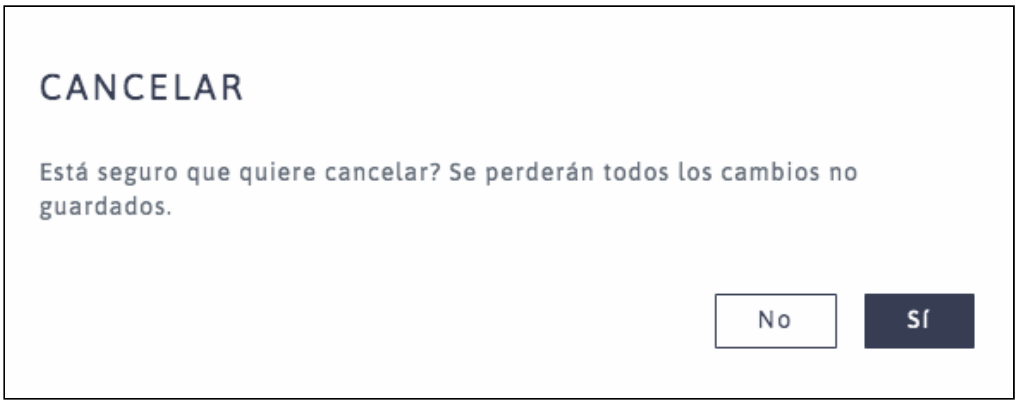


Imagen 110: confirmación al cancelar cambios en perfil de egresado

Imagen 111: confirmación al eliminar requisito en perfil de institución académica

Imagen 112: confirmación al solicitar verificación de calificación

Reconocimiento en lugar de recordar

“Minimizar la carga de memoria del usuario haciendo visibles los objetos, las acciones y las opciones. El usuario no debería tener que recordar información de

una parte del diálogo a otra. Las instrucciones de uso del sistema deben ser visibles o fácilmente recuperables siempre que sea apropiado".

Se intentó que toda la información necesaria siempre esté disponible en el sistema. A modo de ejemplo, al momento de verificar una calificación, las instituciones académicas disponen de toda la información necesaria. En la misma pantalla pueden visualizar todo lo necesario para verificar la identidad del egresado y además toda la información de la calificación ingresada por el egresado.

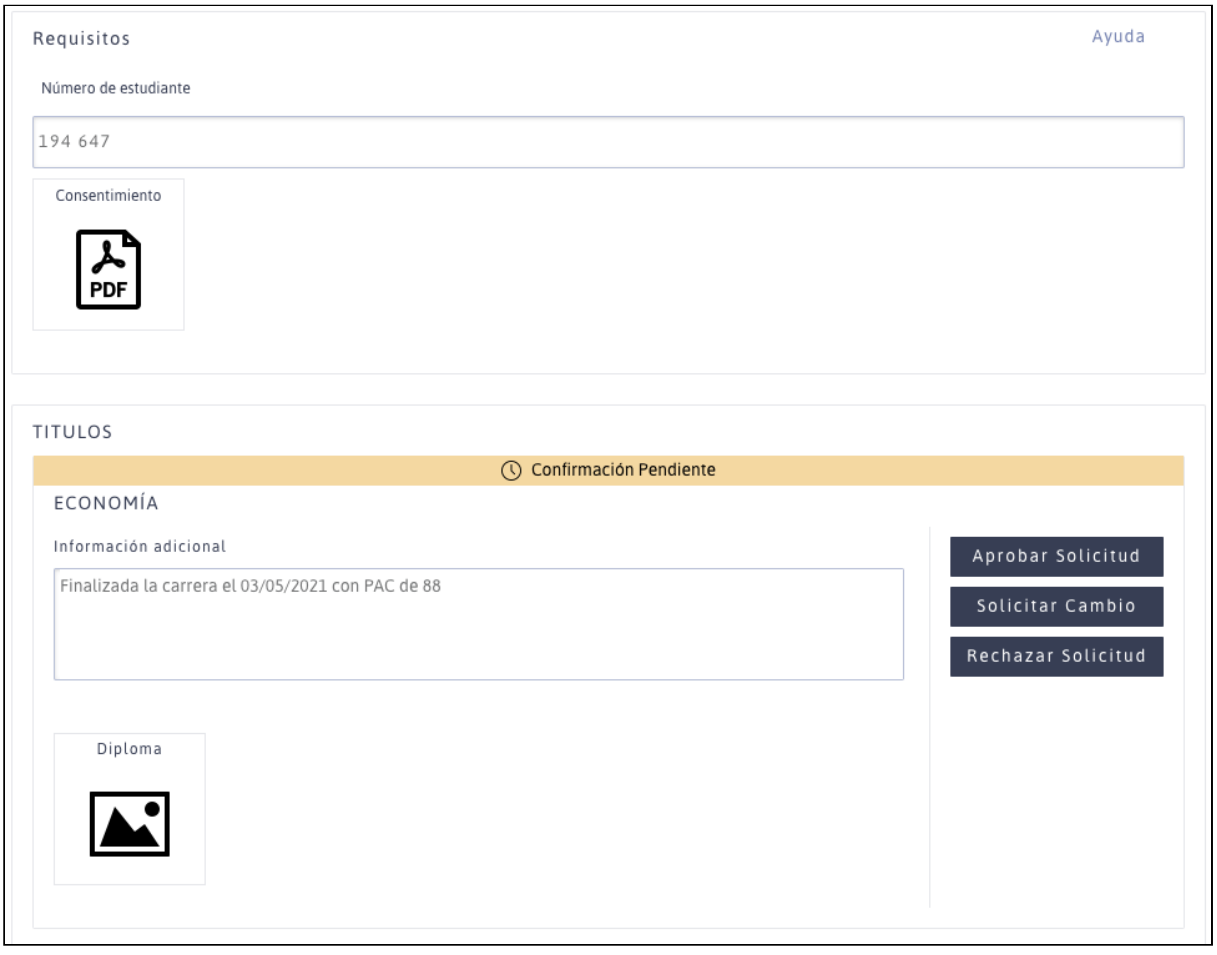


Imagen 113: visualización de perfil de egresado por parte de institución académica

También se intentó respetar esta heurística en el perfil del egresado haciendo que las opciones estén siempre visibles. Previamente, se contaba con el siguiente menú flotante:

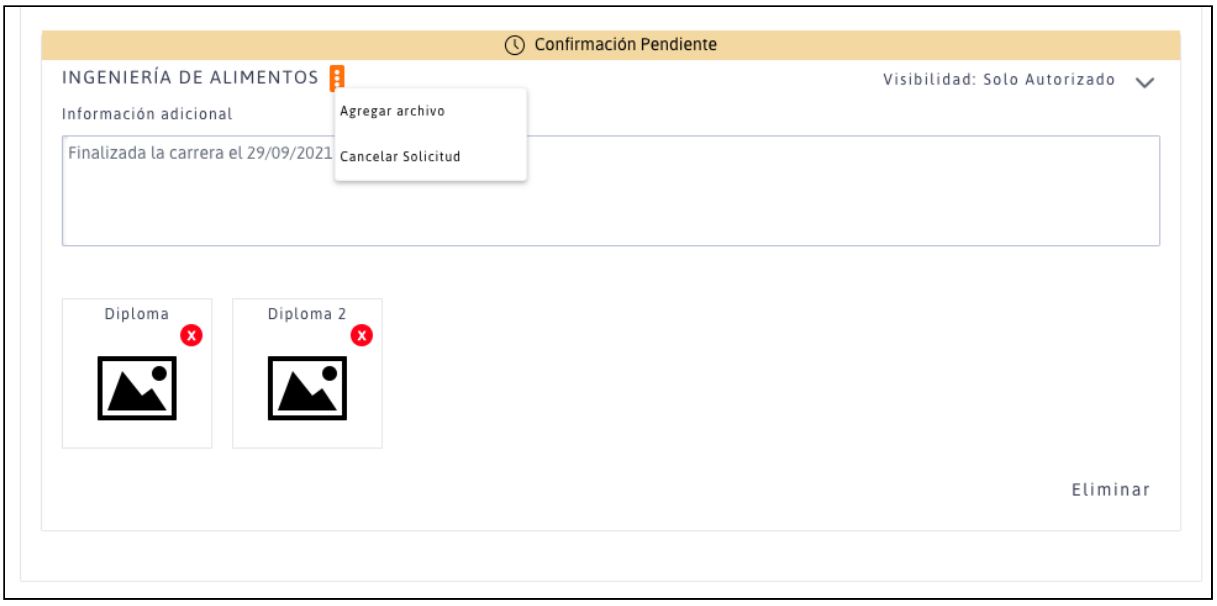


Imagen 114: visualización de calificación en perfil de egresado antes del cambio

Con el objetivo de hacer que todas las acciones sean visibles, se quitó el menú flotante y se agregaron botones siempre visibles:

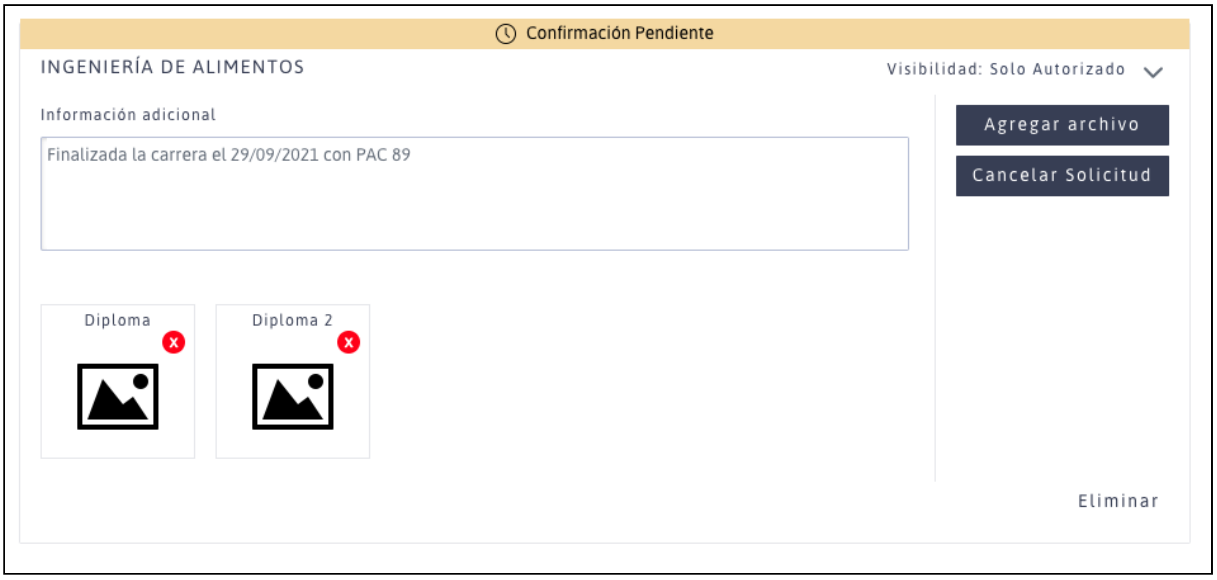


Imagen 115: visualización de calificación en perfil de egresado antes del cambio

Flexibilidad y eficiencia de uso

“Los aceleradores, no vistos por el usuario principiantes, a menudo pueden acelerar la interacción para el usuario experto, de modo que el sistema puede atender tanto a usuarios inexpertos como experimentados. Permitir a los usuarios personalizar las acciones frecuentes.”

Un ejemplo de los aceleradores que se incluyeron es en la sección del perfil de la institución académica, en la cual el usuario puede crear títulos rápidamente utilizando el *tab* para pasar de un campo a otro, la barra espaciadora para seleccionar el tipo de título, y luego el enter para crearlo.

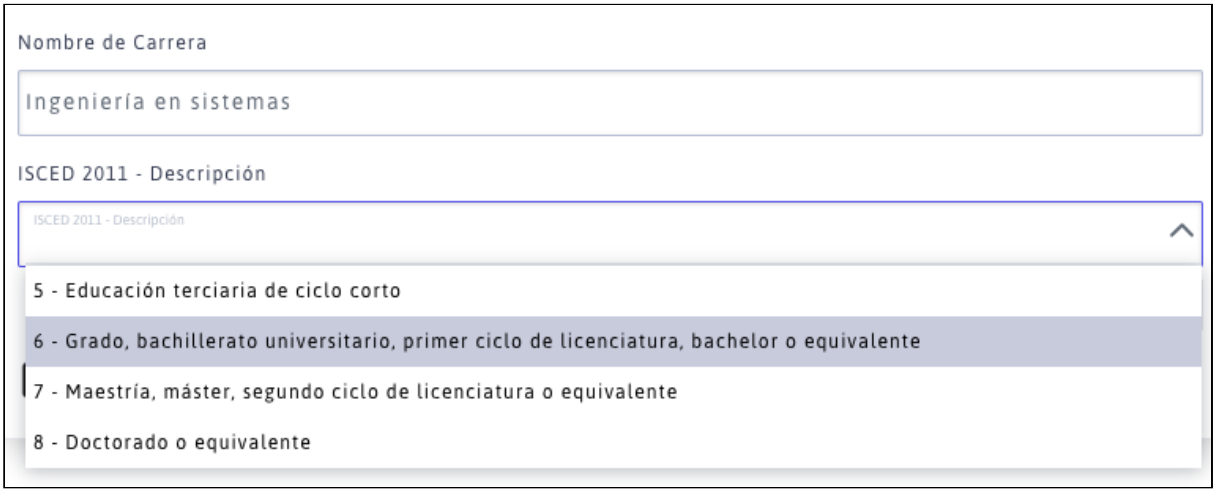


Imagen 116: creación de título

Ayudar a los usuarios a reconocer, diagnosticar y recuperarse de errores
“Los mensajes de error deben expresarse en lenguaje sencillo (sin códigos), indicar con precisión el problema y sugerir constructivamente una solución.”

Se intentó incluir mensajes de error descriptivos en lenguaje entendible por el usuario. Algunos ejemplos de estos son:

- Cuando una institución académica intenta ingresar al perfil de un egresado que no existe o no ha ingresado calificaciones para su institución.
- Cuando un usuario intenta ingresar con una cuenta que aún no ha sido verificada.

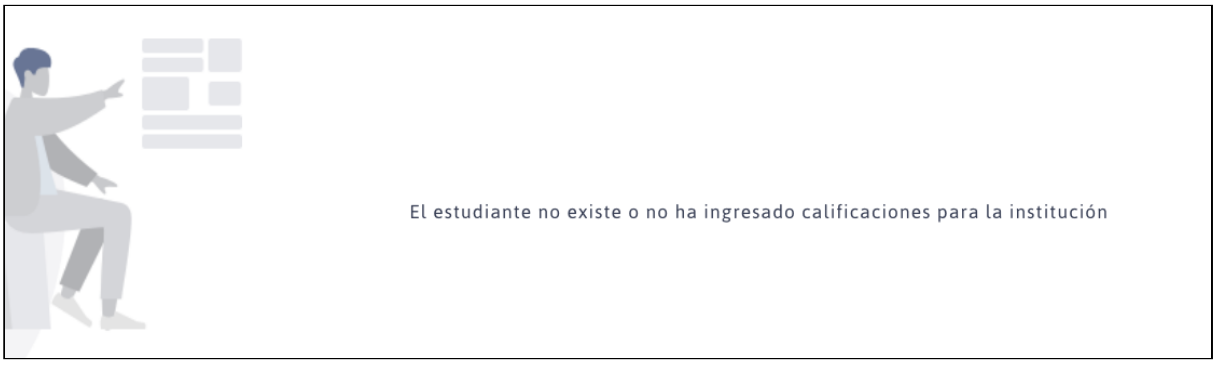


Imagen 117: mensaje de error al acceder a perfil de egresado

Imagen 118: mensaje de error al iniciar sesión con cuenta no confirmada

Ayuda y documentación

“Aunque es mejor si el sistema se puede usar sin documentación, puede ser necesario proporcionar ayuda y documentación. Cualquier información de este tipo debe ser fácil de buscar, centrarse en la tarea del usuario, enumerar los pasos concretos que se deben realizar y no ser demasiado grande.”

Se incluyeron varias ayudas a lo largo de la aplicación. Las ayudas pueden ser brindadas por el sistema. Algunos ejemplos son:

- Ingreso de títulos en perfil de institución académica (ofrecida por el sistema).
- Ingreso de requisitos en perfil de institución académica (ofrecida por el sistema).

Imagen 119: ayuda en creación de títulos

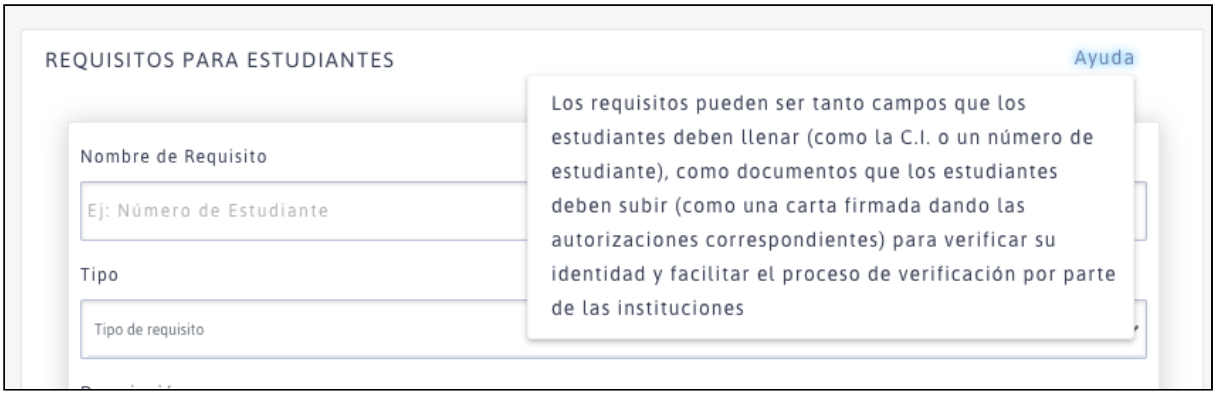


Imagen 120: ayuda en creación de requisitos

Las ayudas también pueden ser ingresadas por las instituciones académicas para que los egresados que ingresen calificaciones para su institución puedan verlas. Esto sucede con los requisitos. En el perfil de institución académica, si un requisito se ve de la siguiente manera:

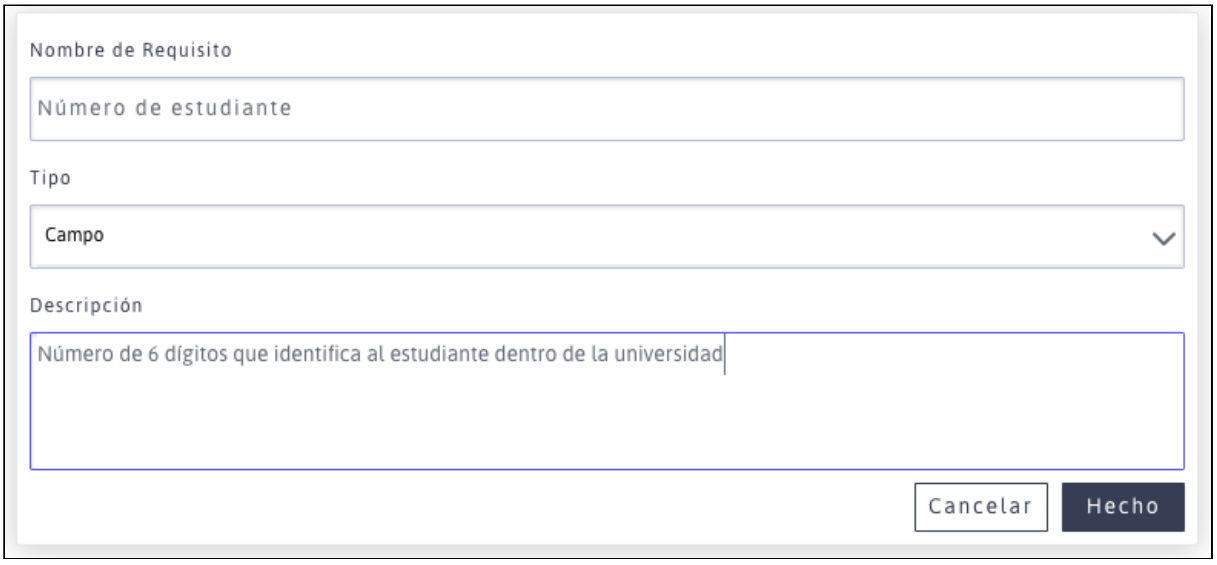


Imagen 121: ingreso de descripción de requisito

Luego el egresado vería lo siguiente a la hora de ingresar datos para esa institución académica:

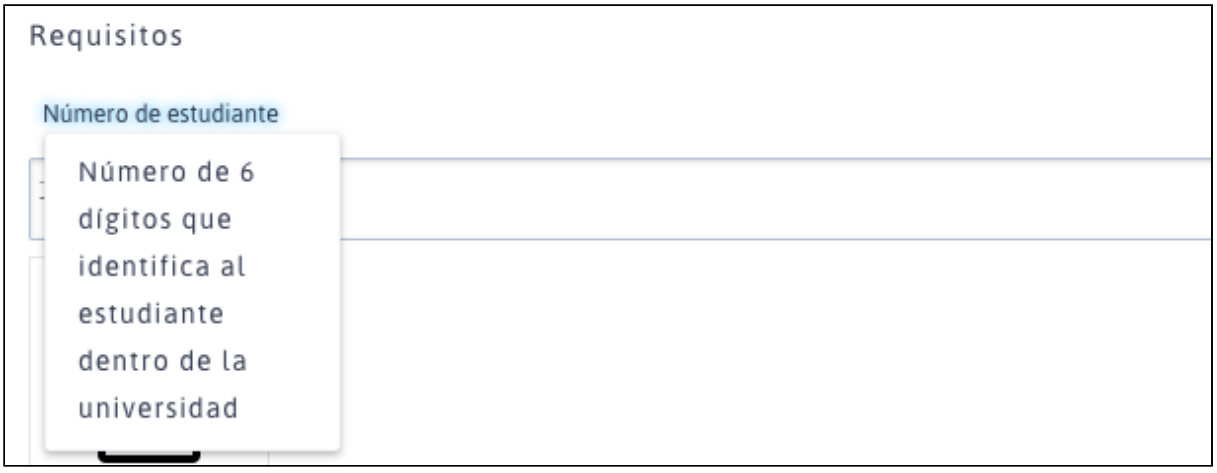


Imagen 122: ayuda al ingresar información para requisito

12.24. Encuesta de clima de trabajo en el equipo

Con el fin de medir el grado de satisfacción de cada integrante del equipo con el ambiente de trabajo, se realizó una encuesta al finalizar el proyecto. Los resultados de la misma fueron posteriormente utilizados para medir el cumplimiento de objetivos de equipo. Para realizarla, se utilizaron preguntas presentes en el documento “Clima de trabajo en el equipo”, de ORTsf.

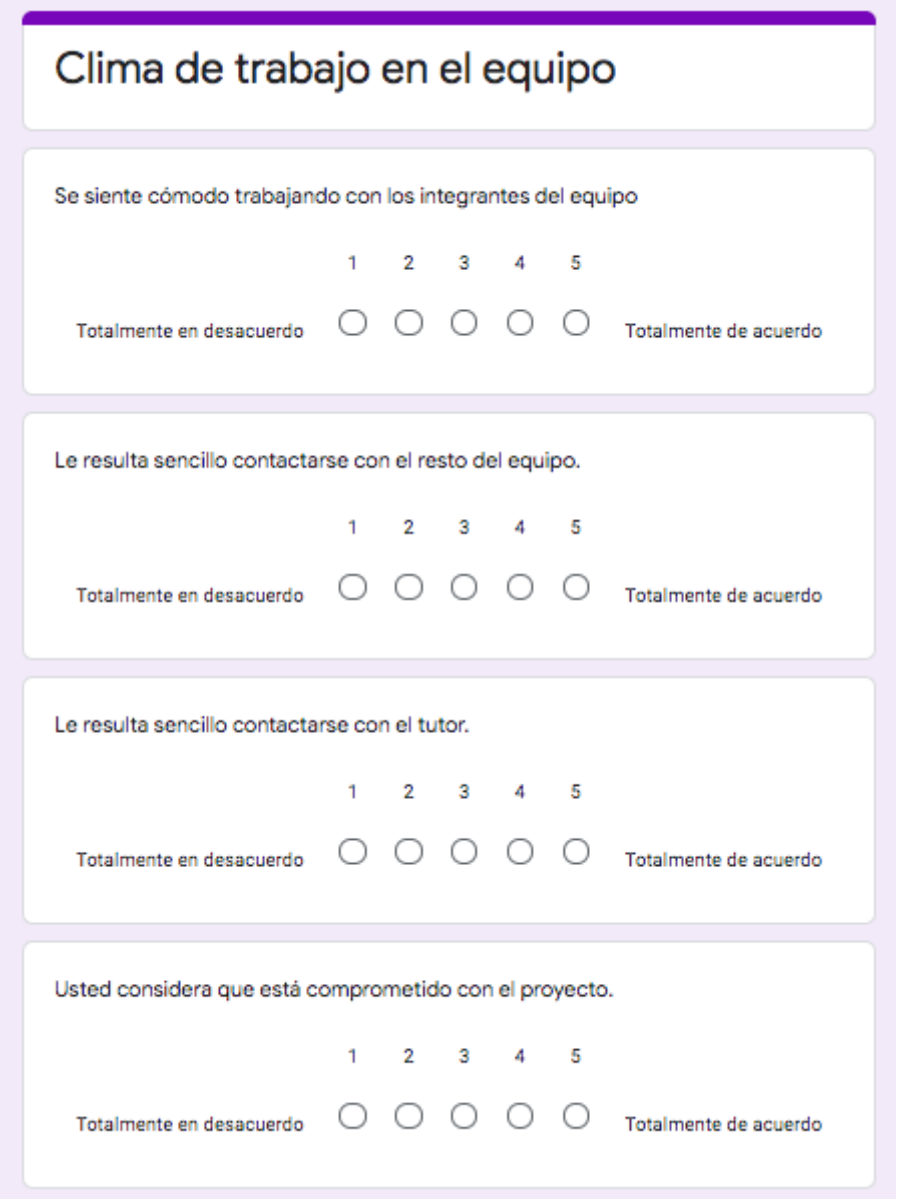


Imagen 123: encuesta de clima de trabajo en el equipo

12.25. Referencias bibliográficas del anexo

[1] Google Docs, "RSK Educate - Executive Course", 2020. [Online]. Available: https://drive.google.com/file/d/1nATjtMEEINSbYEaAceNEXno-2nLr2MG0/view. [Accessed: 04- Mar- 2020].

[2] BitInfoCharts, "Ethereum Avg. Transaction Fee chart",2020. [Online]. Available: https://bitinfocharts.com/comparison/ethereum-transactionfees.html. [Accessed: 04- Mar- 2020].

[3] BitInfoCharts, "Bitcoin Avg. Transaction Fee chart", 2020. [Online]. Available: https://bitinfocharts.com/comparison/bitcoin-transactionfees.html. [Accessed: 04- Mar- 2020].

[4] Blockgeeks, "Smart Contract Platforms Comparison: RSK vs Ethereum vs EOS vs Cardano", 2020. [Online]. Available: https://blockgeeks.com/guides/smart-contract-platforms-comparison-rsk-vs-ethereum-vs-eos-vs-cardano/. [Accessed: 04- Mar- 2020].

[5] Stateofthedapps.com, "State of the DApps — DApp Statistics", 2020. [Online]. Available: https://www.stateofthedapps.com/stats. [Accessed: 04- Mar- 2020].

[6] Medium, "Dual Track Agile: Focusing on Customer Value", 2020. [Online]. Available: https://medium.com/kevin-on-code/dual-track-agile-focusing-on-customer-value-a2e39312585b. [Accessed: 04- Mar- 2020].

[7] Insights on Latest Software Technologies - Simform Blog, "AWS Lambda vs EC2: Comparison of AWS Compute Resources", 2020. [Online]. Available: https://www.simform.com/aws-lambda-vs-ec2/. [Accessed: 04- Mar- 2020].

[8] Netguru.com, "Node.js vs Ruby on Rails Comparison: Which Environment Should I Choose for My Next Project?", 2020. [Online]. Available: https://www.netguru.com/blog/node-js-ruby-on-rails-which-environment-to-choose-next-project. [Accessed: 04- Mar- 2020].

[9] Netguru.com, "Ruby on Rails: Pros & Cons. What You Should Know Before Choosing The Technology", 2020. [Online]. Available: https://www.netguru.com/blog/pros-cons-ruby-on-rails. [Accessed: 04- Mar- 2020].

[10] Community.cloudera.com, "What is the difference between SQL and NO sql? Which one is faster??", 2020. [Online]. Available: https://community.cloudera.com/t5/Support-Questions/What-is-the-difference-between-SQL-and-NO-sql-Which-one-is/td-p/152336. [Accessed: 04- Mar- 2020].

[11] AltexSoft, "The Good and the Bad of ReactJS and React Native", 2020. [Online]. Available: https://www.altexsoft.com/blog/engineering/the-good-and-the-bad-of-reactjs-and-react-native/. [Accessed: 04- Mar- 2020].

[12] AltexSoft, "The Good and the Bad of Angular Development", 2020. [Online]. Available: https://www.altexsoft.com/blog/engineering/the-good-and-the-bad-of-angular-development/. [Accessed: 04- Mar- 2020].

[13] Smashing Magazine, "A Comprehensive Guide To User Testing — Smashing Magazine", 2020. [Online]. Available: https://www.smashingmagazine.com/2018/03/guide-user-testing/. [Accessed: 04- Mar- 2020].

[14] Nielsen Norman Group, "10 Heuristics for User Interface Design: Article by Jakob Nielsen", 2020. [Online]. Available: https://www.nngroup.com/articles/ten-usability-heuristics/. [Accessed: 04- Mar- 2020].