Universidad ORT Uruguay

Facultad de Ingeniería

SIGAMAS

Sistema de Gestión Agrícola y Manejo de Suelos

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

Ana Silva - 150700
Andrés Nieves - 155058
Clara Youdale - 150026
Mauro Varela - 147705

Tutor:
Álvaro Ortas

2015

Declaración de autoría

Nosotros, Ana Silva, Andrés Nieves, Clara Youdale y Mauro Varela, declaramos que el trabajo que se presenta en esta obra es de nuestra propia mano.

Podemos asegurar que:

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

Fecha: 5 de Marzo de 2015

Ana Silva

Clara Youdale

Andrés Nieves

Mauro Varela

Agradecimientos

A lo largo de este año, en el cual hemos sorteado muchos obstáculos y obtenido grandes logros, contamos con el apoyo de muchas personas, las cual sin ellas el Proyecto SIGAMAS no hubiese sido posible.

En primer lugar queremos agradecer a nuestros familiares y amigos los cuales de alguna manera u otra estuvieron en todo momento, alentando y apoyando a cada uno de los integrantes del equipo en este año.

A nuestro tutor Álvaro Ortas, quien fue el integrante número cinco del equipo, el cual con toda su experiencia y conocimiento nos fue rumbeando en el largo camino de la elaboración del proyecto. Hemos aprendido mucho de Álvaro y no solo para el desarrollo del proyecto sino como personas.

A nuestros tres revisores, Nicolás Fornaro, Marcelo Cagnani y Rafael Bentancur quienes en cada revisión nos aportaron mucho conocimiento y ayudaron a mejorar todos los aspectos de SIGAMAS.

Un reconocimiento especial a Tabaré Alcorta, Ingeniero Agrónomo, el cual nos abrió las puertas de su casa, para que desde un comienzo vayamos todas las semanas a adquirir el conocimiento que no poseíamos respecto al agro. Fue nuestro cliente el cual no solamente brindó los requerimientos del sistema sino que ayudó a validar los mismos.

A Pablo Montes, coordinador de la cartera Nacional de Proyectos Ambientales en la Dirección Nacional de Medio Ambiente, nos abrió muchas puertas en el MGAP contactándonos con autoridades, nos invitó al Congreso de Suelos realizado en Colonia y concretó reuniones con el catedrático de la Facultad de Agronomía de la Universidad de la República, Mario Pérez.

A Diego Mariani, docente de la Universidad ORT, que nos ayudó mucho con la creación del CANVAS y dio aportes muy valiosos para definir aspectos relacionados al negocio.
A Leonardo Scafarelli y Amalia Álvarez por su gran aporte en la realización del plan SQA.

A nuestros jefes en los respectivos trabajos, los cuales siempre con buena disposición nos otorgaban los días de estudio que necesitábamos para continuar avanzando con el proyecto.

A Mónica Barbazán encargada del departamento de Suelos y Aguas de la Facultad de Agronomía, quien nos brindó la posibilidad de presentar SIGAMAS frente a todos los participantes del Congreso de Suelos en el Hotel Sheraton de Colonia. Este hecho manifestó el interés de varios de los presentes lo cual simplificó la tarea de validación de SIGAMAS.

Por último queremos agradecer a la Facultad de Ingeniería de la Universidad ORT Uruguay, por la oportunidad única de realizar el proyecto de grado de nuestra carrera. Fue una experiencia inolvidable la cual nos acompañará siempre y que nos hizo crecer profesional y humanamente.

Ana, Andrés, Clara y Mauro.

Abstract

El presente documento describe el sistema SIGAMAS, realizado como proyecto final de carrera para la obtención del título de Ingeniero en Sistemas de la Facultad de Ingeniería de la Universidad ORT Uruguay.

SIGAMAS es un sistema cuyo principal objetivo es brindar un servicio de apoyo para la toma de decisiones en la agricultura, ayudando al productor a maximizar la producción y reducir los costos. Se detectó que los productores del Uruguay carecen de un sistema de características similares y en ocasiones tienen pérdidas significativas por no controlar de manera eficaz su producción.

SIGAMAS asiste a los productores a lo largo de todo el ciclo de vida del cultivo, aconsejando sobre acciones a realizar, alertando por posibles desvíos y aportando posibles mejoras para la producción. Además de mostrar datos en tiempo real, como ser, datos climáticos, cotización de monedas y precios de los principales *Commodities* agrarios.

El sistema consiste de un sitio web al que tendrán acceso los productores mediante usuario y contraseña y desde donde se encuentren tendrán la información necesaria para continuar con la producción del cultivo.

La metodología utilizada para realizar el proyecto fue FDS, un híbrido entre las metodologías Scrum y FDD, con un ciclo de vida incremental. Desde el punto de vista tecnológico el sistema fue construido utilizando la plataforma JEE de Java.

A lo largo del proyecto se atravesaron dificultades, las cuales se sobrepasaron exitosamente, habiendo alcanzado los objetivos y construido una primera versión Beta del sistema en la cual se cumplen las principales características planteadas desde un principio. Fue una etapa muy enriquecedora y de aprendizaje, tanto profesional como en lo personal.

Se cuentan con fuertes proyecciones a futuro debido a todo el *Feedback* positivo que se recibió en las diferentes reuniones mantenidas a lo largo del proyecto.

Palabras clave

SIGAMAS; Ingeniería de Software; Scrum; FDD; *Sprint*; Manejo de suelos; Gestión agrícola;

Índice

Índice de Ilustraciones

Índice de Tablas

Glosario

ABM: Acrónimo para alta, baja y modificación.

***Back-end*:** Se le dice *Back-end* a la parte de la aplicación que controla las entradas provenientes del *Front-end*.

Backup: Respaldo. En informática se usa haciendo alude al respaldo de datos

BD: Base de Datos.

Beta Tester***s*:** Usuarios que ayudan a encontrar errores en la versión Beta de un sistema.

Branch: En los controles de revisión y la configuración de elementos de Software, un *Branch* es la duplicación de un objeto que se encuentra bajo control de revisiones.

***Browser*:** Navegador de Internet

***Bug*:** Error, defecto, falla en el Software.

***Burndown*:** Diagrama de representación gráfica del trabajo por hacer en un proyecto en el tiempo.

***Burndown Chart*:** Métrica de Scrum que permite observar el desempeño de un equipo durante una iteración.

Cambio climático: Modificación del clima con respecto al historial climático a una escala global o regional.

***Case sensitive*:** Sensible a las mayúsculas o minúsculas.

***Checkin*:** Proceso mediante el cual se registra la llegada.

***Checkout*:** Proceso mediante el cual se registra la salida.

CIE: Centro de Innovación y Emprendimiento.

Código abierto: Es la expresión con la que se conoce al Software distribuido y desarrollado libremente.

***Commit*:** Se refiere a la idea de consignar un conjunto de cambios.

***Commodities*:** es cualquier producto destinado a uso comercial. Al hablar de *commodity*, generalmente se hace énfasis en productos genéricos, básicos y sin mayor diferenciación entre sus variedades.

Cultivar: Grupo de plantas seleccionadas artificialmente por diversos métodos a partir de un cultivo más variable, con el propósito de fijar en ellas caracteres de importancia para el obtentor que se mantengan tras la reproducción.

Cultivo: Denominación genérica de cada uno de los productos de la agricultura.

Data source: Origen de datos

***Daily Meetings*:** Reunión diaria de los integrantes del equipo.

EAR (*Enterprise Archive*): Formato de archivo utilizado por Java EE para empaquetar uno o más módulos en un único archivo para el despliegue en un servidor de aplicaciones.

ECS: Elementos de configuración de Software.

EJB (*Enterprise Java Bean*): API para la construcción de aplicaciones J2EE de Oracle *Corporation*.

Elicitación: Término de computación que se refiere al traspaso de información de un punto a otro, en forma fluida.

***Epics*:** Es una *User Story* demasiado grande para completarse en un *Sprint* o no está totalmente especificada. Puede componerse de varias *User Stories*.

Erosión del suelo: Degradación del suelo que produce pérdida de la fertilización del mismo.

Esfuerzo: Es la cantidad de horas por persona que se le ha dedicado a cierta tarea.

ESRE: Especificación de requerimientos del Software

Estado fenológico: Cada una de las etapas por las que pasan las plantas a lo largo de un período vegetativo

Excel: Aplicación del paquete de Office de Microsoft que permite realizar cálculos, planillas y gráficas.

Experto del dominio: Persona con conocimientos en el área de conocimiento del proyecto.

***Extreme Programming*:** Metodología de desarrollo ágil de Software.

FDD (*Feature-Driven Develpment*): Proceso iterativo e incremental de desarrollo de Software. Metodología Ágil.

FDS (*Feature-Driven Scrum*): Fusión de FDD con Scrum

Feature: Es un conjunto de funcionalidades que el cliente desea.

***Feedback*:** Devolución, comentarios y opiniones acerca del trabajo realizado.

Fenología: Ciencia que estudia la relación entre los factores climáticos y los ciclos de los seres vivos.

***Glassfish*:** Servidor de aplicaciones Java.

***Google Drive*:** Servicio de alojamiento de archivos en la nube.

GP: Gerente de proyecto.

***Hosting*:** Servicio que se provee a usuarios de internet en un sistema para poder almacenar información a través de la web.

***Hostname*:** Nombre de dominio.

HTTP (*Hypertext Transfer Protocol*): Protocolo de transferencia de hipertexto.

Indentación: Anglicismo que define el espacio que se genera en el código al realizar una tabulación.

Ingeniería de Software: Es la aplicación sistemática de métodos, herramientas y técnicas para transformar requerimientos establecidos en un eficaz y eficiente sistema de software.

Ingeniero Agrónomo: Profesional que maneja los recursos naturales renovables en forma racional

***Internet Explorer*:** Navegador web de Microsoft.

JAR (*Java archive*): Archivo Java el cual permite ejecutar aplicaciones Java.

Java: Java es un lenguaje de programación orientado a objetos creado en 1995 diseñado por Sun Microsystems.

JEE: Java *Platform*, *Enterprise Edition*, es una plataforma de programación que permite desarrollar aplicaciones empresariales.

JPA (*Java Persistence API*): Tecnología para el desarrollo de aplicaciones web basadas en Java.

Manejo de Suelos: Uso considerado del suelo para evitar la erosión del mismo.

***Merge*:** Es una operación fundamental que reconcilia múltiples cambios realizados sobre archivos controlados mediante revisiones.

MGAP: Ministerio de Ganadería, Agricultura y Pesca.

Modelo fenológico: Es la estimación del cronograma de los distintos estados fenológicos.

MSF *(Microsoft Solutions Framework)*: Es un conjunto de principios, modelos, disciplinas, conceptos y guías desarrollado por Microsoft para entregar soluciones de tecnología de la información.

MVOTMA: Ministerio de Vivienda, Ordenamiento Territorial y Medio Ambiente.

MySQL: Es un sistema de gestión de base de datos relacional.

Netbeans: Es un entorno de desarrollo libre hecho principalmente para el lenguaje de programación Java.

Open Source: Software libre.

ORM *(Object-Relational Mapping)*: Es una técnica de programación que permite convertir datos entre el sistema de tipos utilizado en un lenguaje de programación orientado a objetos y la utilización de una base de datos relacional como motor de persistencia.

ORTsf: Software *Factory* de la Universidad ORT Uruguay.

Parcela: Pequeña porción de terreno.

***Path*:** Ruta de acceso.

***Planning Poker*:** Método de estimación consensuado entre integrantes de un equipo.

Plugin: Es una aplicación que completa a otra

POJO (*Plain Old Java Object)*: Es una sigla utilizada por programadores Java para enfatizar el uso de clases simples y que no dependen de un *Framework* especial.

PrimeFaces: Es una librería de componentes de código abierto que permite facilitar la creación de aplicaciones web.

***Product Backlog*:** Conjunto de requisitos priorizados.

Productor Agrícola: Es una persona que cultiva la tierra con la intención de sembrar un producto agrícola para comercializar o para fines de alimentación humana o animal.

Prototipo: El prototipo de un sistema es una versión inicial de este sistema que está disponible en etapas tempranas del proceso de desarrollo.

Proyecto Agrícola: Es un conjunto de actividades que se realizan para sembrar un producto agrícola.

QA: Aseguramiento de la calidad.

Ranking: Lista ordenada por un atributo en particular

Resiliencia: Es el término empleado en ecología de comunidades y ecosistemas para indicar la capacidad de estos de absorber perturbaciones, sin alterar significativamente sus características de estructura y funcionalidad; pudiendo regresar a su estado original una vez que la perturbación ha terminado

Retrabajo: Repetición de un trabajo realizado

Rotaciones de cultivos: Conjuntos de cultivos realizados cronológicamente en un periodo de tiempo

RUP: Proceso Racional Unificado, proceso de desarrollo de Software.

SaaS (Software *as a Service)*: Software como un servicio.

SCM: Gestión de la configuración del Software.

Scrum: Marco de trabajo para el desarrollo ágil del Software.

Scrum *master*: Persona que mantiene los procesos y formas de trabajo en Scrum.

Selenium: Entorno de pruebas de Software para aplicaciones basadas en la web.

Skype: Software que permite comunicaciones de texto, voz y video sobre Internet.

***Smartphones*:** Teléfono inteligente.

***Sprint*:** Iteración en la metodología Scrum.

***Sprint Retrospective*:** Retrospectiva del *Sprint*.

***Sprint Review*:** Reunión para la revisión del *Sprint*.

SQA: Aseguramiento de la calidad del Software.

Stack: Lista ordenada a la cual se accede a sus elementos de forma de LIFO *(Last In First Out)*

Stakeholders: Todas las partes interesadas en la realización de un proyecto de Software, que se ven afectados tanto negativa como positivamente.

***Story Points*:** Medida arbitraria utilizada en la metodología Scrum.

Tab: Término utilizado para denominar a una pestaña HTML.

***Tablets*:** Computadora portátil de mayor tamaño que un teléfono inteligente, integrada en una pantalla táctil.

Template: Plantilla.

Testing: Prueba de Software.

User friendly: Interfaz amigable del usuario.

***User Story*:** Historia de usuario, forma para describir un requisito en Scrum.

Versión Beta: Representa generalmente la primera versión completa del programa informático o de otro producto, que es posible que sea inestable pero útil para que las de inspección previa

WAR (*Web Application Archive*): Archivo JAR utilizado para distribuir una colección de clases Java, JavaServer *Pages*, archivos XML, que juntos constituyen una aplicación web.

Whatsapp: Aplicación de mensajería multiplataforma.

1. INTRODUCCIÓN

El presente documento tiene el objetivo de describir el desarrollo del proyecto SIGAMAS, Sistema de Gestión Agrícola y Manejo de Suelos, que se realizó en el período de Marzo del 2014 a Marzo del 2015. El mismo surge como proyecto final de la carrera de Ingeniería en Sistemas de la Universidad ORT de Uruguay en el marco del laboratorio de Ingeniería de Software (Software *factory* - ORTsf).

1.1. Entorno conceptual de Software *Factory*

El Laboratorio denominado ORT Software *Factory*, ORTsf, es una organización académica dedicada a la enseñanza de prácticas de Ingeniería de Software, a la mejora de procesos de Software, a la transferencia de tecnología a la industria y a la producción de Software. [1]

1.1.1. Visión

Ser líderes en la generación de conocimiento sobre la producción de Software de calidad. Ser un referente dentro de la Universidad, en el medio y la región, en la creación y aplicación de prácticas de Ingeniería de Software para la producción de productos de calidad. [2]

1.1.2. Misión

El laboratorio de Ingeniería de Software es una organización abocada a la formación de profesionales que desarrollen productos que satisfagan a sus clientes, focalizando la atención en la producción de Software de forma industrial y en proveer tecnología probada al mercado. [2]

1.1.3. Políticas de calidad

ORT Software *Factory* es una organización abocada a la producción de Software de forma industrial, focalizando la atención en la formación de profesionales que realicen productos que satisfagan a sus clientes.

La meta perseguida es consolidar una posición de liderazgo en el mercado en la generación de conocimiento sobre la producción de Software de calidad.

La mejora continua de la calidad de los productos se quiere lograr a través de la mejora del proceso de desarrollo de Software, capacitando y brindando la posibilidad de que cada integrante pueda desarrollar sus habilidades, y jerarquizando al cliente como el objetivo principal.

Dar tecnología probada al mercado productor de Software de la región, y divulgar resultados como forma de mejorar el entorno son algunos de los objetivos de la organización. [2]

1.2. El equipo

Ana Silva

Objetivos Personales

- Crecer tanto profesional como personalmente
- Conocer mejor mis habilidades en el trabajo en equipo para explotarlas y mis debilidades para superarlas.
- Aplicar todos los conocimientos aprendidos a lo largo de la carrera y ganar experiencia al respecto.
- Recibirme de Ingeniero en Sistemas.

Más información: LinkedIn Ana Silva Tealde

Andrés Nieves

Objetivos Personales

- Profundizar mis conocimientos en desarrollo de aplicaciones JEE
- Aplicar técnicas de Ingeniería de Software para hacer más eficiente el desarrollo
- Salvar el proyecto de fin de carrera
- Profundizar conocimientos en diseño de procesos

Más información: LinkedIn Andrés Nieves

Clara Youdale

Objetivos Personales

- Realizar un proyecto emprendedor que sea útil para el nicho seleccionado
- Aprender del mundo agrícola para expandir mis conocimientos en nuevas áreas
- Aplicar los conocimientos adquiridos a lo largo de la carrera para crear un producto de calidad
- Obtener el título de Ingeniero en Sistemas

Más información: LinkedIn Clara Youdale

Mauro Varela

Objetivos Personales

- Instruirme y formar mi primera experiencia en un emprendimiento.
- Aprender a trabajar en equipo con personas con las cuales no había trabajado previamente.
- Poner en práctica todos los conocimientos que previamente fueron adquiridos en la carrera.
- Incorporar nuevos conocimientos respecto a tecnologías de desarrollo.

Más información: LinkedIn Mauro Varela

1.3. Objetivos

Al comenzar el proyecto en abril 2014, el equipo se planteó una lista de objetivos a cumplir para marzo 2015. Estos objetivos del proyecto fueron divididos en dos categorías: aquellos propiamente del proyecto y objetivos de equipo, los cuales fueron gestionados explícitamente por ser novedosos al nunca haberlos enfrentado.

Estos objetivos ayudaron a mantener un foco activo durante la elaboración del proyecto y a alertar sobre posibles desvíos no deseados. El cumplimiento de los objetivos fue importante para el equipo ya que permitió observar el avance del proyecto y ajustarlo si alguno se veía incumplido.

1.3.1. Objetivos de equipo

- **Proyecto en equipo**

Aprender a realizar un proyecto de cero con 4 integrantes y lograr terminarlo con éxito ya que al principio del proyecto, ninguno de los integrantes había realizado un proyecto académico u obligatorio conjuntamente con otras 3 personas.

El objetivo estará cumplido si todos los integrantes finalizan con una buena relación luego de aprobado el proyecto.

- **Aceptación del tema elegido**

Lograr un consenso y aceptación en el tema elegido, todos los integrantes deben estar motivados con la idea general del proyecto y que el proyecto satisfaga todas las expectativas personales y académicas. Se realizó una reunión entre los integrantes del equipo donde se evaluaron cada una de las propuestas.

Este objetivo se considera cumplido si el tema es aceptado por todos los integrantes.

1.3.2. Objetivos del proyecto

- **Versión Beta**

El principal objetivo del proyecto es el de crear una versión Beta del sistema que sea útil para los usuarios finales y que aporte valor agregado a quienes la utilicen. Los usuarios de la primera versión Beta del sistema serán importantes para optimizar el sistema e ir adaptándolo a sus necesidades. Estos usuarios serán *Beta* Testers. Los usuarios finales deberán evidenciar mejoras en los procesos y resultados de las tareas que realicen al utilizar el sistema SIGAMAS. Mediante el *Feedback* obtenido se logrará en un futuro ofrecer una versión definitiva con la cual el equipo se incursionará en el mercado.

La razón por la cual se optó por una versión Beta y no un producto funcionando correctamente y con el 100% de las funcionalidades, lo cual hubiese sido lo óptimo, es el limitante del tiempo que establece el laboratorio ORTsf.

Esta primera versión Beta no contará con pruebas exhaustivas pero se asegurará que no contenga fallas de categoría grave.

Para medir el éxito de este objetivo se utilizará la métrica “Cantidad de fallas”, las cuales deberían tener cantidad cero de fallas graves como resultados, es decir se aceptan solamente fallas leves o medias.

Este objetivo claramente condiciona y rige la forma de trabajo del proyecto, los procesos de control y aseguramiento de la calidad se realizarán tomando en cuenta el objetivo final de versión Beta.

1.3.3. Objetivos del proceso

Aplicar conocimientos adquiridos

Aplicar los conocimientos adquiridos a lo largo de toda la carrera, es decir aplicar Ingeniería de Software. Todas las materias que se han visto por separado y que enriquecen en determinados aspectos de la Ingeniería de Software, debieron ser realizadas para cada aspecto del proyecto.

Este objetivo se considerará cumplido si el proceso creado se ejecuta con éxito (que las métricas del mismo tengan resultados satisfactorios).

Creación de un proceso

No solo se pretende lograr una primera versión del producto sino también un proceso el cual se utilizará en futuros emprendimientos. El proceso es muy importante para lograr un producto de calidad y si se puede contar con un proceso validado será muy enriquecedor para el futuro del equipo como emprendedores. Para medir el éxito del proceso, se utilizarán las métricas que se muestran en el Capítulo “Descripción del proceso”.

Se considerará cumplido este objetivo si dichas métricas presentan resultados satisfactorios.

1.3.4. Conclusiones

A continuación se detalla para cada objetivo, si el mismo fue cumplido o la etapa en la que este se encuentra:

Objetivos de equipo

Proyecto en equipo: el objetivo no solo fue cumplido sino que todos los integrantes mantienen entre sí una relación de amistad la cual subsiste fuera del ámbito de proyecto.

Aceptación del tema elegido: todos los integrantes del equipo eligieron a SIGAMAS como el proyecto que más les interesaba y motivaba para el final de la carrera universitaria.

Objetivos de proyecto

Versión Beta: se puede afirmar que el objetivo se cumplió, la versión Beta de SIGAMAS fue creada con los principales requerimientos del sistema. También se aseguró que la versión no posea fallas graves, lo cual se puede evidenciar en el Anexo "Resultados de las Pruebas".

Objetivos de proceso

Aplicar conocimientos adquiridos: para cada parte del proyecto se ha aplicado conocimientos de diferentes materias de la carrera, lo cual garantiza el cumplimiento del objetivo.

Creación de un proceso: se logró crear un proceso el cual se evaluó según distintas métricas. Analizadas las mismas se concluyó que el proceso ayudó para la creación del producto de forma eficaz y eficiente.

Objetivos a futuro

Como objetivo a futuro se anhela poder continuar con el proyecto una vez finalizado el mismo. Se considera que es un producto con mucho futuro y cada uno de los integrantes del equipo quiere continuar trabajando para lograr el éxito del mismo.

2. DESCRIPCIÓN DEL PROYECTO

En este capítulo se describirá el proyecto titulado SIGAMAS. La exposición comprende la descripción del equipo, la motivación del proyecto, los referentes que ayudaron a realizarlo, los desafíos que se presentaron y los objetivos que se pretende a futuro para SIGAMAS.

2.1. Descripción del equipo

El equipo del proyecto está constituido por 4 estudiantes que se encuentran finalizando la carrera de Ingeniería en Sistemas en la Universidad ORT Uruguay. Los diferentes roles y tareas asignadas a cada integrante estuvieron no solo de acuerdo a las necesidades del proyecto sino que también a las preferencias, conocimientos, experiencias y habilidades de cada uno.

2.1.1. Roles

Según el proceso definido, el cual se detalla en el Capítulo "Descripción del Proceso", se cuenta con una lista de roles a ser aplicados durante las ejecuciones del proceso. En este proyecto, los roles definidos se asignaron de la siguiente forma:

ROL	RESPONSABLE
Coordinador de Requerimientos	Ana Silva
Ingeniero de Requerimientos	Todos
Ingeniero de Procesos	Ana Silva
Arquitecto	Andrés Nieves
SQAer	Mauro Varela
SCMer	Clara Youdale
Administrador de tareas	Clara Youdale
Desarrollador	Todos
Tester	Todos

Tabla 2-1 Roles definidos y sus asignaciones

La descripción detallada de estos roles se encuentran en el Capítulo "Definición del proceso".

2.2. Motivación del proyecto

2.2.1. El problema y su contexto

Dado que la agricultura genera aproximadamente el 12% del empleo nacional, siendo este porcentaje significativamente mayor en las áreas rurales (70% del empleo), el desarrollo agropecuario y rural es esencial para el desarrollo económico y los objetivos de inclusión social del Uruguay [4]. Conociendo estos datos y entusiasmado con realizar el proyecto de fin de carrera relacionado con el agro, el equipo decidió ponerse en comunicación con algunos Ingenieros Agrónomos conocidos, los que a su vez hicieron nexo con funcionarios del estado vinculados al sector agrario del Uruguay.

Mediante investigación en el mercado y entrevistas con diferentes Productores e Ingenieros Agrónomos se constató que no existe en Uruguay una herramienta que brinde la posibilidad de planificar, ejecutar y controlar la realización de cultivos. Tampoco lo hay una que indique la fenología que un cultivo debería tener en cada momento según el tipo de cultivar, clima y tiempo que hubo en los días pasados.

2.2.2. Oportunidades detectadas

Los beneficios de este sistema pionero en Uruguay son únicos ya que, luego de un estudio realizado, se notó que actualmente la gestión agrícola se realiza de forma precaria con sistema manuales como ser una hoja de cálculo de Excel.

Por otro lado, cuando se validaba el sistema con nuevos Productores estos hacían saber que cada vez que se acercaba el momento de realizar un nuevo cultivo ellos tenían que acceder y consultar mucha información de diferentes lugares. Esto significa un gran costo ya que les lleva tiempo acceder a la misma. Muchas veces se pierden datos importantes, que luego podrían impactar significativamente en el rendimiento de su cosecha o impedirles tomar mejores decisiones para el futuro de su cultivo. Se observó que les agradaba mucho la idea de tener un sitio donde encontrar toda esta información de manera automática sin necesidad de navegar por diferentes páginas o buscar diferentes sitios de noticias relacionadas con el agro.

También se constató que consideraban de gran utilidad poder analizar el rendimiento de los cultivos con lo realizado durante el mismo, tener toda la trazabilidad del mismo y sacar conclusiones para futuras cosechas. Fue muy bien recibida por los productores la idea de estar notificados del estado actual del cultivo y de las tareas que debían realizarse en cada momento sobre todo por aquellos que tienen varios campos alejados entre sí.

A continuación se presenta un esquema que resume el proceso de la gestión agrícola que manejan la mayoría de los Productores Agrícolas de nuestro país y cómo SIGAMAS va a asistir a sus usuarios en el mismo.

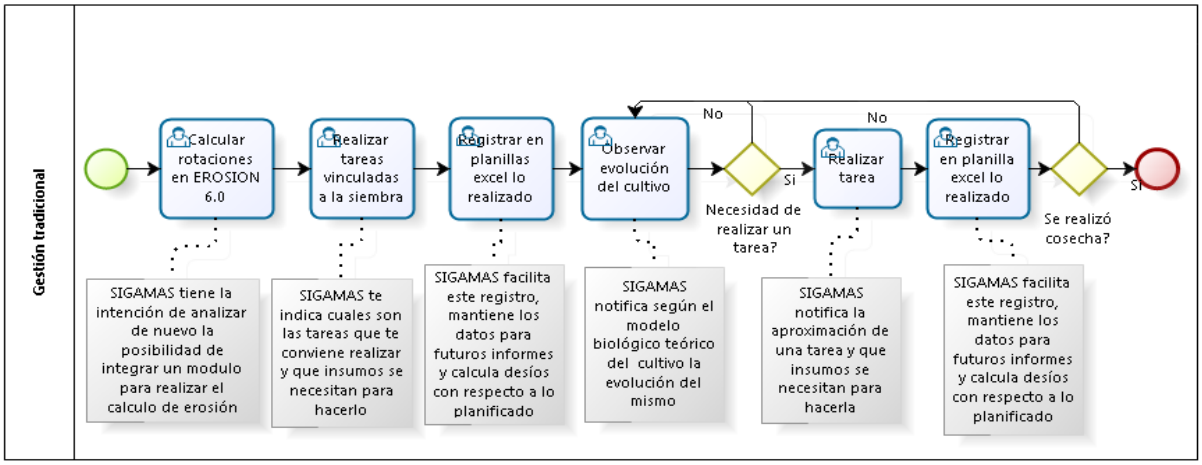


Ilustración 2-1 Proceso actual de gestión de cultivos

2.2.3. Herramientas existentes actualmente

En la actualidad existen herramientas que podrían situarse como posibles competidores, sin embargo, algunas de estas no son nacionales y fueron diseñadas específicamente para ser utilizadas en otros países lo que provoca que no se adapten al mercado nacional. Adicionalmente, estas herramientas no son utilizadas por los productores del Uruguay por manejar diferente vocabulario y distintos cultivares. Aquellas que sí fueron diseñadas para el Uruguay no son estrictamente competidores directos ya que no son herramientas puramente orientadas a la gestión sino que apuntan a aspectos más genéricos del negocio (manejo de stock, facturación, etc.). A continuación se nombran estas herramientas.

Competidores

- OSIRIS Agro
- Physis
- Ñandú
- ViaRural

Productos sustitutos

- Erosión 6.0
- OKARA
- Solapa4

Por una descripción detallada de las mismas, dirigirse al Anexo "Análisis de Herramientas Existentes".

2.2.4. Solución encontrada

El Software tiene como fin brindar una mejora constante en la gestión de la producción agrícola. En un inicio se pretendió que el Software constara con un módulo para el manejo responsable de suelos, es decir, que contuviera la inteligencia e información necesaria para calcular el índice de erosión del suelo y que indicara si este índice se encontraba dentro del permitido por la ley nacional vigente (ver el Anexo "Ley N° 18.564"). Luego de analizado el

mercado y de haberse reunidos con varios expertos del dominio, se decidió retirar este módulo de la solución del proyecto. Esta decisión se tomó no solo porque era muy ambicioso para realizarlo en un año, sino que también los usuarios no lo consideraban útil, no pagarían por algo que lo pueden solucionar con el Software EROSIÓN (brindado gratuitamente por el MGAP – Ministerio de Ganadería Agricultura y Pesca) aunque este no fuera amigable y fácil de usar. De todos modos, el equipo tiene pensado volver analizar su incorporación en alguna futura versión porque es interés del mismo que el sistema sea de ayuda para cuidar el medio ambiente y evitar que los suelos del Uruguay continúen siendo erosionados como hasta ahora (el 30% de los suelos de Uruguay ya se encuentran erosionados y son irrecuperables [3]).

SIGAMAS es una herramienta que ayuda en la toma de decisiones para lograr una gestión de cultivos más eficientes y generando mayores ganancias. Se basa en la información de los modelos fenológicos (relación entre factores climáticos y ciclo de seres vivos) de los diferentes cultivos para sugerir cronogramas que luego serán ejecutados registrando desviaciones y aconsejando oportunidades de mejora.

Las funciones principales están relacionadas con la planificación, ejecución, control y alertas de un cultivo, del cual el sistema calcula las fechas para los diferentes estados fenológicos y las tareas relacionadas con cada uno de estos estados. Permitiendo así notificar y conocer la realidad de cada una de esas chacras. Asimismo, permite personalizar, replanificar, calcular desvíos de cronograma e informar sobre clima, precio de *commodities*, cotizaciones de monedas, precios de insumos, noticias del agro, etc.

Desde un principio se trabajó con distintos expertos en el negocio y posibles usuarios para poder entender las necesidades e implementar una solución que satisfaga con creces dichas necesidades. Esto fue todo un desafío ya que se debía adquirir conocimientos en un rubro totalmente desconocido para los miembros del equipo, pero gracias a los diferentes allegados (expertos del dominio) y el compromiso del equipo surgieron las diferentes funcionalidades del Software. Para una información más detallada de la adquisición del conocimiento en el rubro agro, dirigirse al Anexo “Adquisición del conocimiento”.

Como resultado del esfuerzo de un año, surgió la primera versión de SIGAMAS, la misma es una versión Beta de forma que sea la base sobre la cual se seguirán incorporando funcionalidades para lograr poner en producción una versión operativa. La misma será promocionada con los diferentes vínculos que se han logrado a lo largo del proyecto, los cuales han manifestado su fuerte interés por el mismo.

Una característica de la primera versión es su enfoque a un solo tipo de cultivo. Cuando se inició la investigación se comprendió que había que darle mucha importancia a cada tipo de cultivo y sobre todo las diferencias entre cada uno de ellos. Es por esto, que la primera versión será lanzada solo para el trigo. Se eligió el trigo por diferentes motivos, principalmente porque hace muchos años que se cultiva en el país, por lo tanto sus estados fenológicos y su respuesta en tierras del Uruguay está muy estudiado y registrado.

La razón por la cual no se pudo concretar una versión operativa en producción es que la parte de investigación y adquisición de conocimiento abarcó gran parte del proyecto.

2.2.5. Modelo de negocio de SIGAMAS

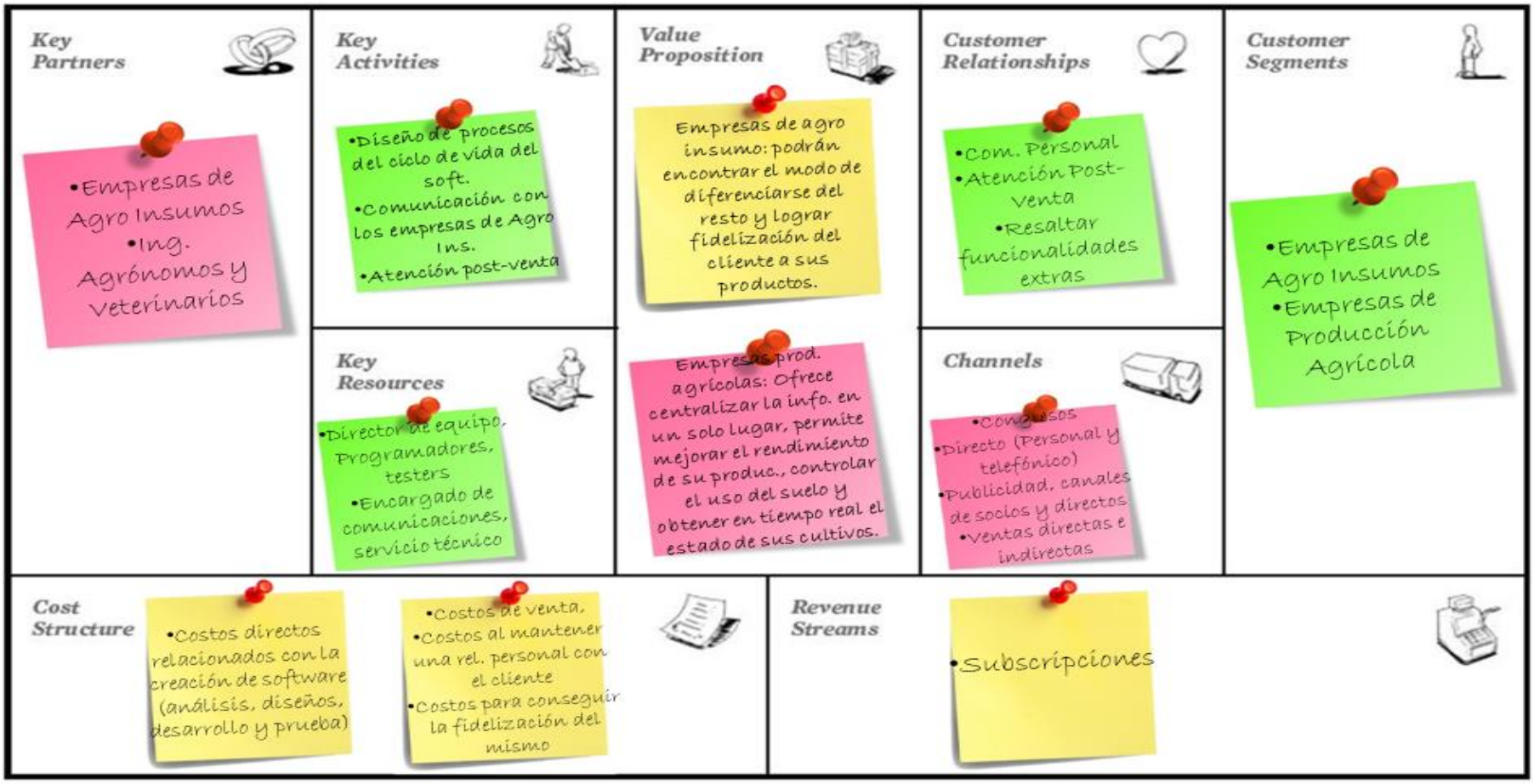


Ilustración 2-2 Modelo de Negocio de SIGAMAS

La explicación detalla del CANVAS se puede encontrar en el Anexo "CANVAS".

2.3. Referentes

Para realizar el proyecto se contó con el Ing. Agrónomo Tabaré Alcorta, quien administra y gestiona los campos de la empresa Nuevo Manantial S.A. [5], y quien actuó como experto del dominio, brindado, ayudando y explicando toda la información relacionada con la gestión de los cultivos.

Además se tuvo la ayuda del Ing. Agrónomo Mario Pereira, funcionario del MGAP (Ministerio de Ganadería, Agricultura y Pesca) en las oficinas que se encuentran en el departamento de Tacuarembó, quien proporcionó información relacionada con los modelos fenológicos de los cultivos.

También se contó con la ayuda de Pablo Montes, biólogo y PhD que trabaja en el MVOTMA (Ministerio de Vivienda, Ordenamiento Territorial y Medio Ambiente), quien aportó todo el material legal y brindó al equipo información vital del Uruguay con respecto al medio ambiente y la importancia e influencia del trabajo del agro sobre el mismo.

2.4. Desafíos del proyecto

A lo largo del proyecto el equipo se encontró con varios desafíos, muchos de los cuales estaban previstos desde el principio y los que no lo estaban no fueron de gran impacto y sirvieron para aprender y crecer.

El primer y gran desafío al que el equipo se enfrentó fue el problema de la comunicación con los expertos del dominio, no solo por no entender su "jerga" sino también por desconocer del tema y del negocio. Para lograr romper esta barrera, fue necesario, después de varios encuentros con expertos del dominio y Productores, dejar en claro que no se diera nada por supuesto y/o entendido porque para ambos lados había barreras de lenguaje que romper correspondientes al área de conocimiento de cada una de las partes. También fue necesario mucho estudio previo sobre el agro, términos, costumbres, fenología vegetal, etc.

El segundo desafío que se encontró fue tecnológico, se necesitó mucha investigación al respecto. Los detalles se encuentran en el Capítulo de "Arquitectura".

Por último y no menos importante, se encontró la necesidad de romper con prejuicios de ambos lados. Por un lado las personas del agro debieron confiar en las propuestas del equipo y en la capacidad del mismo para solucionar sus problemas, creer que el equipo era capaz de entender sus necesidades y problemas aunque sus perfiles sean muy opuestos. El equipo, por otro lado, debió comprender el perfil conservador y reservado de ellos, incluso hallar la manera de superar esto.

2.5. Futuro de SIGAMAS

Como ya se mencionó anteriormente, cuando se finalice el proyecto de grado no se tendrá la versión final de SIGAMAS. Se tendrá un versión Beta, que será lanzada al mercado sin fallas

graves para poder probarlo en casos reales y obtener *Feedback* del mismo. Además, la versión lanzada constará únicamente con la inteligencia necesaria para realizar el cultivo del trigo ya que, según la encuesta agrícola 2014 [6] el trigo se ubica en el segundo lugar, después de la soja, como cultivo principal del Uruguay. El trigo es el principal producto de invierno, con 76 % de las 522.000 hectáreas de plantíos, en verano ocupa el segundo lugar con el 6.9 % de las 1.477.600 hectáreas cultivadas (en verano la soja ocupa el 85.3 %). Pero a diferencia de la soja, el trigo se cultiva en Uruguay desde 1528 [7] (se considera que en el departamento de Soriano se realizaron las primeras plantaciones de trigo en América Latina), mientras que la soja fue introducida recién al final de los años 90' [8]. Esto provoca que, el conocimiento del comportamiento que tiene el trigo en suelos uruguayos es mayor que el de la soja y por lo tanto es más predecible su modelo fenológico.

De todos modos, el equipo tiene como meta para un futuro cercano incorporar la soja y otros cultivos significativos para la agricultura del Uruguay, tales como sorgo, cebada, avena, etc.

También se pretende agregar otros módulos para que el usuario pueda integrar información relevante a su gestión, tales como stock, ventas, etc. Así mismo, se desea estudiar la posibilidad de vincular con un módulo que gestione y controle la erosión de las rotaciones de cultivos (el conjunto de cultivos realizados cronológicamente en un periodo de tiempo).

Se considera que sería útil, hacer una aplicación para *Smartphones* y *Tablets* con el fin que los usuarios puedan usar el sistema en cualquier lugar que se encuentren, inclusive sin tener conexión a Internet, con una base de datos local que luego se sincronice cuando el usuario esté conectado a Internet.

Otro punto que el equipo es consciente que debería rever en un futuro es analizar cómo afecta el cambio climático en SIGAMAS. Según la Organización de las Naciones Unidas para la Alimentación y la Agricultura, el sector agrícola de Uruguay está siendo profundamente afectado por los efectos del cambio climático, que conducen hacia una creciente variación en la producción de cultivos y pasturas, una mayor incidencia de pestes y plagas y una mayor variación en la disponibilidad de agua, entre otros. Inclusive, el Ministerio de Ganadería, Agricultura y Pesca (MGAP), con el apoyo de la FAO implementó el proyecto "Nuevas políticas para la adaptación del sector agropecuario al cambio climático en Uruguay" [9] (noviembre 2010 - marzo 2013) con el objetivo de contribuir a reducir la vulnerabilidad y construir resiliencia de los agro-ecosistemas uruguayos a los efectos del cambio climático sin comprometer el desarrollo potencial del país. Esto implica que, el motor de SIGAMAS que proporciona el modelo fenológico de un cultivo y sus tareas asociadas sea mantenido y actualizado durante toda la vida útil del sistema para que este se mantenga vigente.

2.6. Conclusiones

Todos los integrantes del equipo cuentan con las habilidades y formación necesaria para llevar a cabo con éxito el proyecto.

La solución encontrada es la que mejor se adapta a las necesidades de los futuros usuarios y la evolución del modelo de negocio se fue dando como resultado de un estudio más profundo del mercado. Si bien la solución no es la que inicialmente se deseaba, el equipo fue

consciente desde una etapa inicial que el alcance primario era muy ambicioso y que para avanzar era necesario tener que hacer focos.

Si bien los desafíos que se enfrentaron eran conocidos, fue muy costoso adquirir los conocimientos suficientes y realizar la investigación necesaria para comprender el problema y poder hablar con las personas indicadas de modo correcto.

3. DESCRIPCIÓN DEL PROCESO

Como proceso se entiende el conjunto de pasos, métodos, herramientas y roles, entre otros, que debe seguir el desarrollo de Software. El equipo definió un proceso propio, el cual puede ser ejecutado en este proyecto pero también en futuros emprendimientos o proyectos. A continuación se describe el proceso y las metodologías que se evaluaron para poder crearlo.

3.1. Metodologías evaluadas

Para poder generar un proceso que se pueda ejecutar en este y futuros proyectos, el equipo decidió evaluar distintas metodologías de trabajo que ayudaran a crear la estructura del proceso. Se analizaron comparativamente a las siguientes: Scrum [10], FDD [11], *Extreme Programming* [12], MSF [13], RUP [14] y MSDM [15].

Para el análisis de estas metodologías el equipo decidió analizar las siguientes características. Cada característica se identifica por un nombre, una descripción y una importancia que va de 1 a 5.

- Característica A:
  - El equipo necesitaba estar preparado para cambios de requerimientos durante el proyecto.
  - Importancia 5

- Característica B:
  - No existía un contrato rígido por parte de un cliente
  - Importancia 3

- Característica C:
  - El equipo era un grupo pequeño de menos de 10 integrantes
  - Importancia 5

- Característica D:
  - El ciclo de vida debía ser iterativo y con iteraciones cortas
  - Importancia 4

- Característica E:
  - La metodología debía priorizar la comunicación directa y la colaboración
  - Importancia 4

- Característica F:
  - Se debía contar con ceremonias de planificación, revisión y retrospectiva.
  - Importancia 5

- Característica G:
  - Se debía contar con el rol de Project Manager
  - Importancia 4

- Característica H:
    - La metodología debía ser ágil
    - Importancia 2

Valiéndose de la técnica de *Planning Poker*, el equipo analizó el grado de cumplimiento de estas características que tenían las metodologías analizadas.

Para comparar y evaluar las distintas metodologías, se realizó una tabla comparativa, con las metodologías y las características a cumplir. Si una metodología cumplía con una característica, se le sumaba el valor de la importancia de la misma.

A continuación se muestra la tabla comparativa:

Característica → / **Metodología** ↓	A (5)	B (3)	C (5)	D (4)	E (4)	F (5)	G (4)	H(2)	Total (32)
Scrum	✓	✓	✓	✓	✓	✓	✗	✓	28
FDD	✓	✓	✗	✓	✓	✗	✓	✓	22
Extreme Programming	✓	✓	✓	✓	✓	✗	✗	✓	23
RUP	✓	✗	✗	✓	✓	✗	✓	✗	17
MSF	✓	✓	✗	✓	✓	✗	✗	✗	16
DSDM	✓	✓	✗	✓	✓	✗	✓	✓	19

Tabla 3-1 Comparación de metodologías

Como se puede observar en la Tabla 3.1-1 la metodología que más se adecuaba a las necesidades del equipo para generar un proceso era Scrum. Gracias a este análisis, el equipo generó su primer proceso.

Por más información sobre los cálculos de esta planilla, dirigirse al Anexo de "Comparación de las metodologías", en dónde se detalla el procedimiento de comparación y se definen los resultados.

3.2. Primer proceso definido

El primer proceso generado se basó enteramente en las prácticas de Scrum ya que el análisis realizado anteriormente, sugería que la metodología Scrum era la más adecuada.

3.2.1. Descripción general del primer proceso

A continuación se muestra una descripción general del primer proceso realizado.

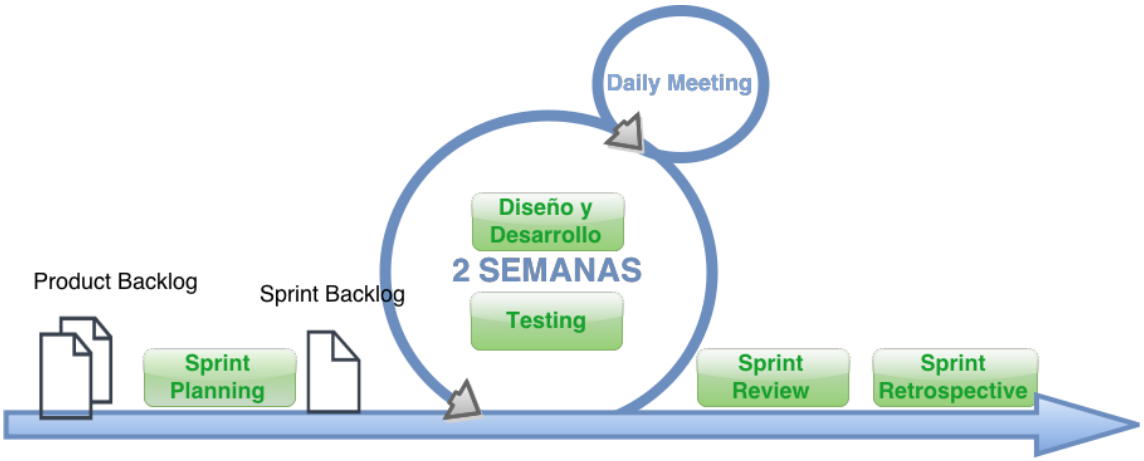


Ilustración 3-1 Descripción general del primer proceso

El mismo se ejecutó en 2 *Sprints* y sus resultados no fueron exitosos. Esto se pudo observar gracias a las métricas recolectadas en dichas iteraciones. Las métricas definidas habían sido:

- *Sprints' Burndowns*
- Velocidad
- Retrabajo

Para ver en detalle porque se consideró que las ejecuciones de este proceso no fueron satisfactorias, dirigirse al Anexo "Registro de las Métricas" y observar los *Sprints* 3 y 4.

3.3. Segundo Proceso definido

Cómo el primer proceso definido no resultó satisfactorio el equipo decidió analizar los errores cometidos para poder definir uno nuevo. Este proceso era el definitivo y el mismo se ejecutó de los *Sprint* 5 a 12 con éxito.

3.3.1. Descripción de las metodologías evaluadas

Al fracasar las ejecuciones del primer proceso, el equipo decidió evaluar otras metodologías que lo ayudaran a definir un nuevo proceso con la esperanza de obtener resultados más satisfactorios. Las metodologías evaluadas la segunda vez fueron Scrum y FDD. Se llegó a la conclusión que evaluar una fusión de ambas metodologías podría llevar a mejores resultados. Se llamó a esta fusión de metodologías *Feature Driven* Scrum (FDS).

Se realizó una lista de elementos que pertenecían a cada una de las metodologías para evaluar cuáles de ellos podían ser útiles al nuevo proceso. Esto permitió formalizar las etapas del nuevo proceso. A continuación se muestran los elementos que se tomaron o no en

cuenta para evaluar Scrum y FDD. Para ver el análisis completo de estas metodologías dirigirse al Anexo "Comparación de las metodologías".

- **¿Qué se evaluó de Scrum?**

Ceremonias *Sprint Planning, Sprint Review y Sprint Retrospective*

Fue muy importante mantener las ceremonias de Scrum ya que permitían realizar reuniones cada 2 semanas para revisar el *Sprint* culminado, planificar el próximo y realizar la retrospectiva.

Iteraciones llamadas *Sprint*

Se debían mantener el nombre *Sprint* para las iteraciones porque era lo que se había realizado desde un principio y cambiar no aporta ningún beneficio. Estas iteraciones debían tener un largo fijo de 2 semanas.

Story Points

Se mantuvieron las *Story Points* de Scrum ya que permitían calificar las funcionalidades a desarrollar, según sus dificultades, para crear un *ranking* de las mismas. Además, generaban la métrica de velocidad, que era importante para medir el ritmo de desarrollo del equipo.

Rol de Scrum *Master*

Cómo no todos los miembros del equipo estaban familiarizados con Scrum, se mantuvo el rol de Scrum *Master*. Este era el encargado de verificar y gestionar las tareas, pero además debía controlar que las ceremonias de Scrum se estuvieran cumpliendo correctamente. Además, en caso de dudas, era el encargado de brindar información y conocimiento sobre la metodología.

- **¿Qué no se evaluó de Scrum?**

Daily Meetings

Para gestionar los *Sprints*, se analizaron varias herramientas *online* de manejo de metodologías ágiles y se eligió AgileFant. Una herramienta *online* de manejo de metodologías ágiles permite gestionar *Sprints*, *User Stories* y tareas, agilizando así el registro de horas y asignaciones de tareas. Por información detallada sobre qué son las herramientas de gestión ágil dirigirse al Anexo de "Comparación de herramientas de gestión de metodologías ágiles".

Cuando se aplicó Scrum puro y se ejecutó el primer proceso, se incluyeron las *Daily Meetings*. Esto resultó ser ineficiente y no aportó ningún beneficio adicional ya que AgileFant permitía saber en qué estaba cada integrante en todo momento. Las *Daily Meetings* eran entonces innecesarias.

- **¿Qué se evaluó de FDD?**

Etapas evaluadas de FDD

En la práctica, se observó en los *Sprints* 1 y 2 que las iteraciones estaban conformadas por la implementación de un conjunto de requerimientos, con un gran parecido a como lo hace FDD (construir por *Feature*). Se formalizó este proceso, construyendo el sistema de *Feature* en *Feature*. Cada una de estas implementaciones se compuso de la siguiente forma:

- *Design* (que se llama Diseño preliminar)
- *Design Inspection* (que se llama Diseño final)
- *Coding* (que se llama Desarrollo)
- *Unit Testing* (que se llama Pruebas unitarias)
- *Integration* (que se llama Liberación)
- *Code Inspection* (que se llama Revisión)

Por más información sobre la ejecución de los *Sprints* 1 y 2, dirigirse al Anexo "Registro de las Métricas" y observar los *Sprints* 1 y 2.

Roles que se mantienen de FDD

Cuando el proyecto arrancó, ya se habían definido ciertos roles. Estos se asimilaban a algunos roles de FDD, sin embargo no se hizo uso de todos ellos. Se descartaron algunos y se mantuvieron los siguientes:

- *Project Manager* (Administrador de Tareas)
- Arquitecto jefe (Arquitecto)
- Experto del dominio
- *Release Manager* (SCMer)
- Tester
- *Developer* (Desarrollador)
- *Deployer* (también el SCMer)

- **¿Qué no se evaluó de FDD?**

Múltiples diseños

Al ser pocos integrantes, presentar múltiples diseños en cada *Feature* era contraproducente. En cada iteración se debió presentar solamente un diseño de solución.

Iteraciones con largo variable

Cómo se había decidido mantener los *Sprints* de Scrum, se optó por eliminar las iteraciones de largo variables que propone FDD y mantener iteraciones fijas de 2 semanas.

- **¿Qué actividades propias se agregaron?**

Revisión

Se establecieron estándares para los documentos propios del equipo. Fue necesario realizar revisiones de dichos documentos para verificar el cumplimiento de estos. Ni Scrum ni FDD realizan esta actividad.

Pruebas cruzadas

Se debían realizar pruebas cruzadas entre los desarrolladores para evaluar los distintos módulos.

3.3.2. Descripción del proceso

Las etapas principales que se destacaron del proceso fueron las siguientes:

Sprint Planning → Definición de *Features* →Diseño y Desarrollo →*Testing* → *Sprint Review* →*Sprint Retrospective*

A continuación se muestra una descripción general del proceso.

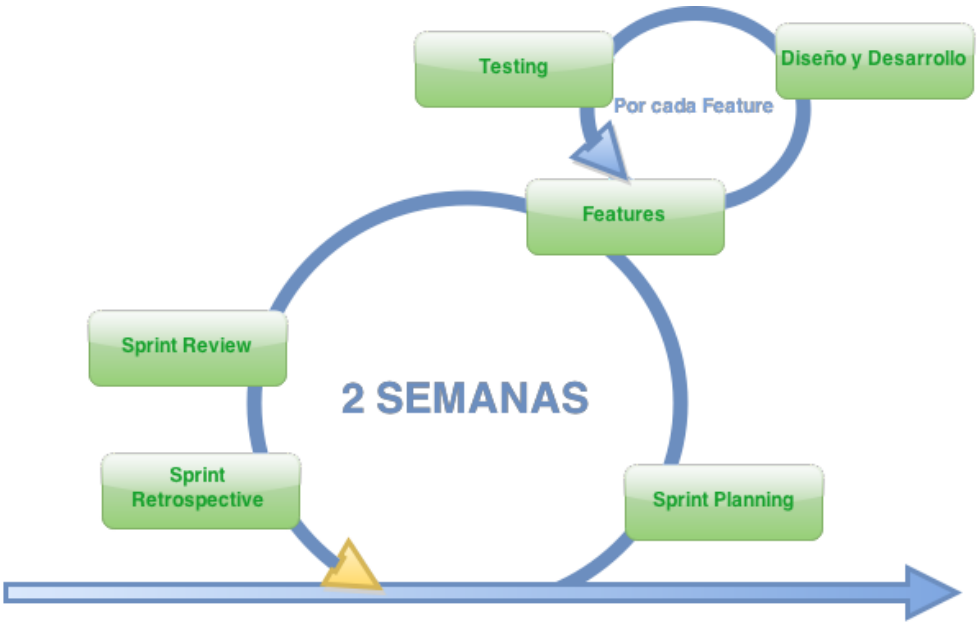


Ilustración 3-2 Proceso.

Cómo se puede ver en la figura anterior, el proceso es iterativo con iteraciones fijas de 2 semanas. Para información detallada del proceso y de sus etapas ver Anexo de "Definición del proceso".

3.3.3. Roles del proceso según etapas

Gracias a las metodologías evaluadas, se definieron los roles que formaron parte del proceso y que fueron necesarios para la ejecución del mismo.

Roles comunes a todas las etapas

Son roles que intervinieron en todas las etapas del proceso.

Administrador de tareas: Figura en el grupo de proyecto que se asemejaba a la de líder. Encargado de coordinar las tareas, reuniones y consultar sobre cómo se encontraba cada integrante. Ese rol era similar al de gerente de proyecto, aunque incluyó al rol de Scrum *Master*.

SCMer: Encargado de las versiones del Software, del control sobre sus cambios y estados, y a su vez de la integración de las partes de Software en un solo producto.

Ingeniero de procesos: Encargado de que todos los procesos que se ejecutaban en el proyecto sean eficientes y eficaces para la producción del sistema.

SQAer: Encargado de definir el plan de aseguramiento de la calidad, las posteriores revisiones y cumplimiento del mismo.

Experto del dominio: Encargado de brindar la información sobre el dominio de la aplicación y el problema específico que necesita resolverse.

Requerimientos

Coordinador de requerimientos: Encargado de redactar y administrar los requerimientos del sistema.

Ingeniero de requerimientos: Encargado de relevar y analizar los requerimientos del sistema.

Diseño y desarrollo

Diseñador: Encargado de realizar el diseño de la solución a desarrollar.

Arquitecto de Software: Encargado de toda la arquitectura del sistema.

Desarrollador: Encargado de generar el código de la aplicación.

Testing

Tester: Encargado de probar las funcionalidades del sistema.

3.3.4. Métricas definidas para evaluar la ejecución del proceso

Las métricas que se realizaron en las ejecuciones del proceso fueron:

Sprint Burndown

La primera métrica, particular de Scrum, era el *Sprint Burndown*. Consistía en marcar la diferencia entre el *Burndown* ideal y el real. El *Burndown* refiere a la medición del trabajo que se ejecuta día a día. Permite medir si el proyecto está retrasado, a tiempo o adelantado.

Velocidad

En Scrum, la velocidad se utiliza para analizar la cantidad de trabajo realizado durante un período de tiempo. Esta cantidad de trabajo puede evaluarse en Horas o *Story Points*.

Indicador de retrabajo

La tercera y última métrica, es la de retrabajo. Consiste en analizar cuánto trabajo de iteraciones anteriores se está realizando nuevamente y permite medir la eficiencia de iteraciones pasadas.

3.3.5. Ingeniería de Requerimientos

La Ingeniería de Requerimientos es el proceso que permite relevar, analizar y especificar requerimientos.

Relevamiento de Requerimientos

El relevamiento permitió descubrir los requerimientos que serán implementados. Para lograrlo, se debía seguir la estrategia que se describe a continuación.

Se definieron las dimensiones del relevamiento de requerimientos, las cuales son, un entorno en el cual se mueve el relevamiento. Relevar es una tarea compleja, ya que muchas veces los clientes no saben exactamente lo que quieren, o muchos tienen opiniones diferentes que pueden generar conflictos. Estas dimensiones permitieron mantener la consistencia de lo relevado y preguntar las preguntas correctas al momento de recolectar información. Estas dimensiones provienen del libro *Requirements Engineering: Processes and Techniques* de G. Kotonya [16] y son las siguientes:

Ilustración 3-3 Dimensiones del relevamiento de Requerimientos

Por información más detallada sobre estas dimensiones, ver Anexo "Definición del proceso".

3.3.6. Testing

Para el *Testing* (prueba del Software) fue importante establecer niveles de prueba que se adaptaran al proyecto en ejecución. En lo posible, debían usarse herramientas de automatización que permitieran profundizar las pruebas.

3.3.7. Gestión de riesgos

Gestión del riesgo se encargó de mantener bajo control los posibles daños y perjuicios que pudieran ocurrir durante la ejecución del proceso.

Descubrimiento de riesgos

Para descubrir los riesgos y "atraparlos" antes de su ocurrencia, se debían analizar los siguientes puntos definidos por V. Kanabar [17] en su libro *MBA Fundamentals Project Management*.

- Especificaciones inconsistentes, incoherentes o incompletas.
- Responsabilidades pobremente definidas
- Pobre eficiencia de un producto
- Comunicación poco efectiva
- Control de cambios débil

Clasificación de respuesta a los riesgos

Para definir la gravedad de los riesgos se realizó un análisis cualitativo de los mismos. El análisis cualitativo consistió en asignarle a cada riesgo, dos parámetros:

1. **Probabilidad de ocurrencia**
   La probabilidad de ocurrencia era un valor que iba de 0 a 1 con los siguientes valores:
   Muy Improbable = 0,1
   Relativamente Probable = 0,3
   Probable = 0,5
   Muy probable = 0,7
   Casi certeza = 0,9

2. **Impacto**
   El impacto era un valor que iba de 1 a 5 con los siguientes valores:
   Impacto muy bajo = 1
   Impacto bajo = 2
   Impacto moderado = 3
   Impacto alto = 4
   Impacto muy alto = 5

Estos valores se basan en el libro PMBOK [18].

3.3.8. Gestión de la configuración

El propósito de la gestión de la configuración (SCM) es establecer y mantener la consistencia de las piezas del Software a lo largo de su ciclo de vida. Se aplicaron procedimientos técnicos y administrativos para identificar, definir las piezas de Software; controlar modificaciones y versiones de estas piezas; registrar y reportar el estado de cada pieza y las solicitudes de modificaciones.

Los elementos de configuración (ECS) son todos aquellos elementos que formen parte de la gestión de configuración. Estos elementos son:

- Códigos fuentes
- Datos
- Ejecutables
- Documentación

Se debe contar con un repositorio para documentos y otro para el código.

El repositorio de documentos debía seguir una estructura que se adapte a las etapas del proceso para facilitar la administración de documentos y la búsqueda de los mismos.

El repositorio de código no debía seguir ninguna estructura especial ya que depende enteramente del proyecto a ejecutar.

3.3.9. Conclusiones

La ejecución del proceso permitió la realización del proyecto SIGAMAS, el cual siguió los pasos y restricciones que el proceso sostiene.

La ejecución de dicho proceso se midió gracias a las métricas definidas. En ese momento se pudo concluir que el proceso ejecutado fue exitoso.

4. INGENIERÍA DE REQUERIMIENTOS

Este capítulo describe el proceso de ingeniería de requerimientos en este proyecto. Se detallarán los actores que participaron de esta actividad, cómo se realizó el relevamiento de requerimientos (la elicitación), cómo se especificaron y cómo fueron validados.

4.1. Roles aplicados

Según el proceso, los roles definidos para llevar a cabo la etapa de Ingeniería de Requerimientos fueron:

- Administrador de tareas
- SCMer
- Ingeniero de procesos
- SQAer
- Experto del dominio
- Coordinador de requerimientos
- Ingeniero de requerimientos

¿Cómo se aplicaron estos roles?

El Administrador de tareas fue el encargado de administrar el esfuerzo invertido en la parte de requerimientos. Esto fue, la asignación de tareas relativas a la ingeniería de requerimientos y además, la validación del registro de horas por parte de los demás integrantes. Este rol fue personificado por Clara Youdale.

El SCMer fue el encargado de administrar los cambios sobre los distintos documentos y artefactos que correspondían a la ingeniería de requerimientos. Estos fueron, el ESRE y los documentos de relevamiento y análisis de relevamiento. Este rol fue personificado por Clara Youdale.

El Ingeniero de procesos fue el encargado de analizar la ejecución de los procesos de relevamiento, análisis y especificación de requerimientos en busca de mejoras o ajustes. Este rol fue personificado por Ana Silva.

El SQAer fue el encargado de revisar los documentos de relevamiento, análisis y Especificación de requerimientos, y analizar su cumplimiento con los distintos estándares. Este rol fue personificado por Mauro Varela.

El experto del dominio, fue el encargado de guiar, aportando su conocimiento en el área para que se puedan detectar quienes iban a ser los usuarios finales y cuáles iban a ser las funcionalidades del sistema a desarrollar. Este rol fue personificado por el Ing. Tabaré Alcorta.

El coordinador de requerimientos fue el encargado de especificar los requerimientos del sistema. Este, también fue el encargado de realizar los prototipos y reunirse periódicamente con el experto de dominio para validarlos. Este rol fue personificado por Ana Silva.

El rol de ingeniero de requerimientos, fue personificado por todos los miembros del equipo, que se encargaron de relevar y analizar los requerimientos relevados.

4.2. Relevamiento de requerimientos

A continuación se describe cómo se ejecutó el relevamiento de los requerimientos.

4.2.1. Estrategia

Gracias a las dimensiones de relevamiento definidas en el proceso se pudo recolectar información sobre lo que iba a ser la aplicación, los problemas específicos que necesitaban ser resueltos y las necesidades de los interesados en el sistema. Esto permitió generar una estrategia de relevamiento propia de este proyecto.

Ilustración 4-1 Estrategia implementada

4.2.2. Estudio del mercado

Para poder identificar quiénes iban a ser los diferentes actores que intervinieran sobre el sistema, se realizó un análisis de negocio, y más específicamente, del nicho agrícola. Se descubrió que existen dos posibles usuarios, los Productores Agrícolas y los Ingenieros Agrónomos.

4.3. Actores identificados

Gracias a las dimensiones de relevamiento definidas en el proceso (Ver Anexo "Definición del proceso"), se pudieron identificar a los siguientes actores:

- Usuario administrador

Ingeniero Agrónomo o Productor Agrícola con una cuenta en SIGAMAS que desea planificar, ejecutar y controlar sus cultivos.

Los Productores Agrícolas son todas aquellas personas que poseen tierras sobre las cuales se cultiva pero que generalmente no son los encargados de administrar su producción.

Los Ingenieros Agrónomos, son todas aquellas personas que administran cultivos que pueden ser propios o de Productores Agrícolas.

- Administrador del sistema

Persona encargada de gestionar los datos de la aplicación, siendo estos datos las cuentas y los usuarios. La persona con este rol debe poder crear nuevos usuarios y cuentas, así como administrarlos.

- Sistema externo

Este actor comprende a todos los sistemas que se pueden consumir desde la aplicación y despliegan datos útiles a los usuarios administradores.

4.3.1. Técnicas de relevamiento

Para relevar los requerimientos en este proyecto se optó por las siguientes técnicas:

Entrevistas
Se realizaron entrevistas a los distintos entes y organizaciones más importantes que se vinculan con el desarrollo agrícola. Estas entrevistas están detalladas en el Anexo de "Encuestas y Entrevistas".

Encuestas
Se crearon encuestas para poder recolectar información útil y la opinión de los posibles usuarios con respecto a las funcionalidades futuras del sistema. Estas encuestas están disponibles en el sitio web de SIGAMAS (www.SIGAMAS.com). Además, cuando el equipo participó del Congreso Uruguayo de Suelos en Colonia, se distribuyeron tarjetas de negocio con los datos del equipo y se realizaron encuestas a los participantes del congreso. Todas estas encuestas (y sus resultados) están detalladas en el Anexo de "Encuestas y Entrevistas". La información detallada del congreso se encuentra en el Anexo "Primer Congreso Uruguayo de Suelos – Agosto 2014". Para obtener más información sobre el sitio web del equipo, dirigirse al Anexo "Sitio Web SIGAMAS".

4.3.2. Listado y Categorización de requerimientos

Una vez obtenidos los resultados de las encuestas y entrevistas, se procedió al listado preliminar de los requerimientos. Para ello se utilizó Google *Drive*, en el cual, se creó un documento de Especificación de requerimientos (ESRE), administrado por el coordinador de Requerimientos. Se empezó por enumerar los requerimientos recolectados que luego se clasificaron. Vale aclarar que luego de realizada esta lista, la misma pasó por una revisión del experto del dominio quien la validó.

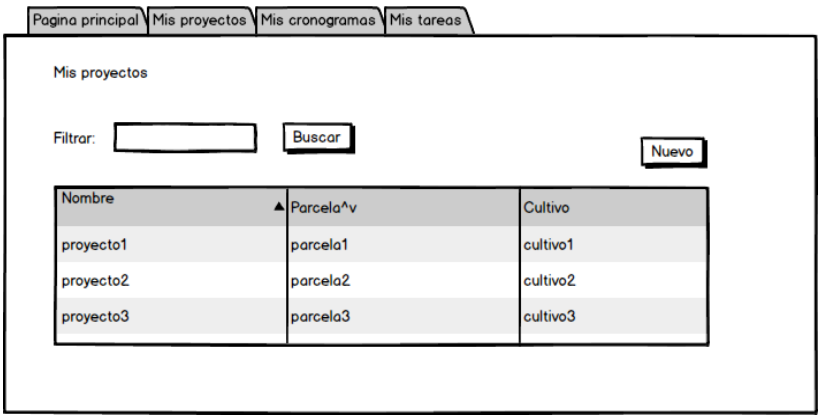
4.4. Especificación de requerimientos

Luego de haberse generado una lista preliminar de requerimientos, fue necesario realizar la especificación de las mismas. Para que la especificación se realice con éxito, los requerimientos fueron escritos siguiendo un estándar, definido para este proyecto.

La especificación se compone esencialmente de una descripción y de un prototipo. El equipo consideró la importancia de especificar con prototipos para poder representar con mayor claridad las funcionalidades del sistema. Esto fue una decisión muy importante ya que permitió validar mejor los requerimientos y mejorar la comunicación con los futuros usuarios para que estos puedan visualizar directamente las funcionalidades.

A continuación se muestra un ejemplo de un requerimiento siguiendo el estándar.

RF21
Nombre: Ver proyectos de cultivos
Actor: Usuario administrador
Descripción: El sistema debe permitir que los usuarios puedan ver los proyectos de cultivos a los cuales están asociados. Los administradores de una cuenta tendrán permiso para modificar todo lo que deseen modificar del proyecto.
Prioridad: Alta
Clasificación: Crítico
Implementado: Si
Story Points: 40
Especificación:



Criterios de aceptación:

Número de	Criterio de	Contexto	Evento	Resultado / Comportamiento

escenario	aceptación			esperado
1	El usuario debe de poder ver los proyectos de cultivos a los cuales está asociado	En caso que el usuario desee ver los proyectos de cultivo a los que está asociado	Cuando el usuario selecciona "Mis proyectos"	El sistema deberá mostrar en pantalla todos los proyectos de cultivos a los cuales está asociado dicho usuario.

Para ver en detalle el estándar, dirigirse al Anexo "Estándar de Especificación de Requerimientos".

Seguir el estándar permitió representar a los requerimientos en forma consistente lo que las vuelve más entendibles.

4.5. Priorización de los requerimientos

En este proyecto se le asoció a cada requerimiento una criticidad, que podía ser "Crítico" o "No Crítico", una prioridad que podía ser "Alta", "Media" o "Baja". Para la cuantificación se optó por los *Story Points*. Estos tomaron los siguientes valores:

- 3: Muy Baja complejidad
- 5: Baja complejidad
- 8: Complejidad Media
- 13: Complejidad Alta
- 40: Complejidad Muy Alta

Estos elementos fueron elegidos por el equipo mediante *Planning Poker*. Se consideró que estos elementos aportaban información útil adicional a los requerimientos los cuales ayudarían al tratamiento de los mismos.

4.6. Validación de los requerimientos

El ESRE (Documento de Especificación de requerimientos) pasó por dos tipos de validaciones.

Validación SQA

El primer tipo de validación, la de SQA, se realizó para corroborar el formato del documento y que los requerimientos cumplieran con el estándar de requerimientos. Esta validación se ejecutó en la etapa de revisión del proceso, por el SQAer.

Validación Experto del Dominio

La segunda validación, la hizo el experto del dominio, para verificar que los requerimientos fueran entendibles y correctos.

4.7. Conclusiones

Se concluye que, implementar una estrategia para relevar los requerimientos y utilizar un estándar permitió generar un ESRE claro, conciso y concreto y que los cambios que se efectuaron sobre el mismo se realizaron fácilmente y rápidamente.

5. ARQUITECTURA

En este capítulo se describirá la arquitectura del producto de Software y las principales decisiones de diseño y tecnológicas que acompañaron a dicha arquitectura. Se presentan también las diferentes vistas para una mejor ilustración.

Para la descripción de la arquitectura se repasarán los atributos de calidad más relevantes a cumplir, derivados de los requerimientos no funcionales definidos, las tecnologías de base seleccionadas para dar soporte a la solución y las tácticas y patrones de diseño arquitectónico que en confluencia con la tecnología permiten cumplir los atributos de calidad y requerimientos no funcionales. Sobre el final del capítulo se ilustra mediante diferentes vistas el diseño final arquitectónico.

Para ver los detalles de las secciones de este capítulo referirse al Anexo "Documento de Arquitectura".

5.1. Atributos de calidad a cumplir

De la fase de requerimientos, se obtuvieron los siguientes requerimientos no funcionales, que definen los principales atributos de calidad a cumplir. Para ver los requerimientos no funcionales relevados dirigirse al Anexo "ESRE - Documento de Especificación de Requerimientos":

Escalabilidad
Dado que es un sistema en fase de adaptación a nuevos requerimientos, es imperativo que sea diseñado pensando en la escalabilidad, de modo de que este pueda crecer en capacidad sin impactar en el desarrollo ya homologado.

Disponibilidad
El sistema debe permitir el acceso de múltiples usuarios al mismo tiempo y mantenerse responsivo, estable y consistente tanto en su operativa como en su manejo de datos.

Seguridad
La seguridad de los datos y el acceso selectivo según usuario a estos debe ser una prioridad para el sistema.

Accesibilidad
El cliente debe poder acceder desde cualquier dispositivo (*Netbook*, *Notebook*, PC, Celular) con el que se pueda conectar a Internet.

Modificabilidad
El sistema va evolucionando a medida que se agregan distintos modelos de cultivos y notificaciones sobre indicadores específicos. Asimismo el desarrollo del proyecto desde sus inicios proyectaba la posibilidad de cambios en cualquier etapa de desarrollo, por lo que este atributo de calidad compromete la calidad del producto.

5.2. Acercamiento a la definición de arquitectura

Dados los atributos de calidad principales a cumplir, y considerando los aspectos del negocio detallados en las secciones precedentes, el análisis de la solución llevó al diseño de un SaaS (Software *As A Service*). La solución propuesta implica un portal web que permita el acceso a los usuarios desde cualquier dispositivo con internet.

5.3. Descripción general del sistema

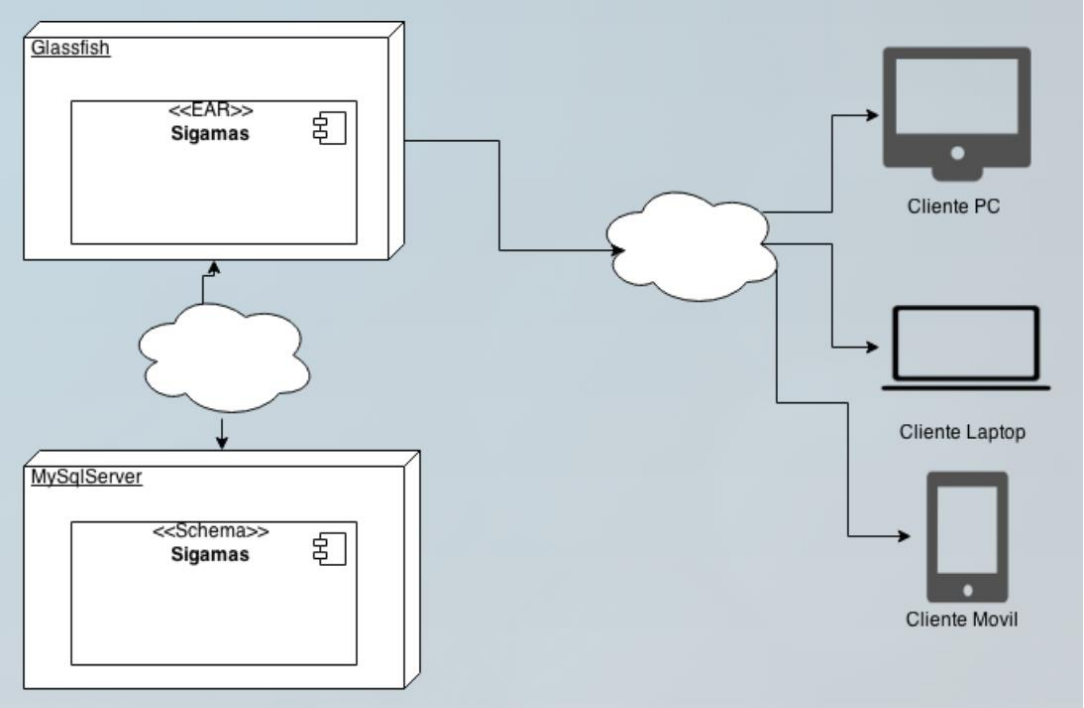


Ilustración 5-1 Diagrama general del sistema

5.4. Decisiones tecnológicas

Al ser un proyecto que cuenta con una porción significativa de investigación, resultó imperativo para el equipo utilizar tecnologías conocidas para el desarrollo de la aplicación, que cumplan tener una curva de aprendizaje baja y permitan un desarrollo ágil y robusto. Las tecnologías elegidas, fueron analizadas por el Arquitecto de tal forma que la utilización de las mismas se contraponga con esta gran carga de investigación.

Fue importante realizar un análisis tecnológico y elegir las tecnologías más convenientes, donde la idea principal se focalizó en generar menos *Sprints* de desarrollo para poder lograr el alcance deseado.

5.4.1. Desafíos tecnológicos

Durante el desarrollo del proyecto se encontraron diferentes desafíos en lo que a tecnologías refiere, desde la elección del lenguaje hasta una propuesta de plan de

despliegue, incluyendo la infraestructura necesaria. Se listan a continuación aquellos desafíos que por su porte fueron más relevantes:

Elección de lenguaje

En el momento en que los requerimientos empezaron a definirse, se tuvieron los elementos como para poder realizar la elección del lenguaje y las tecnologías a utilizar, dado que se esclarecieron puntos claves en cuanto a los requerimientos no funcionales y las prestaciones que debería ofrecer el sistema. Sin embargo, la elección por sí misma no resulta un proceso sencillo dado que se deben sopesar factores no relacionados con el producto objetivo, como ser el tiempo disponible y el conocimiento del equipo de trabajo. En este sentido se ponderaron todos los elementos posibles para llegar a la decisión que resultara más adecuada, como se discute en la próxima sección.

Selección de IDE

Luego de la selección del lenguaje, el entorno de desarrollo es la primera elección compleja que surge. Nuevamente son los factores relacionados a la experiencia del equipo los que terminan pesando más en la decisión, luego de un análisis comparativo de las herramientas más populares.

Elección de *Frameworks*

Tomando en cuenta las restricciones de tiempo y el objetivo final, se debió optar entre distintos *Frameworks* de apoyo para el desarrollo del producto, centrándose la búsqueda en aquellos maduros y bien documentados. La utilización de estos *Frameworks* permite la reutilización de estructuras que agilizan la producción y aportan robustez a la solución.

Definición de arquitectura

Dada la variabilidad posible en los requerimientos por ser un área de conocimiento donde el equipo no tenía experiencia previa, la definición de arquitectura se hizo en forma incremental, e incluso con regresiones importantes, lo que llevó a que se considerará con especial precaución cada decisión determinante, planteando el desafío de que estas pudieran adaptarse y evolucionar a medida que los requerimientos fueran aproximándose a su definición final con el menor impacto posible en el producto ya desarrollado.

Aprendizaje de las tecnologías seleccionadas

Uno de los desafíos más importantes a la hora de aprovechar al máximo el tiempo de desarrollo fue minimizar lo más posible la curva de aprendizaje de los miembros del equipo con menor experiencia en las tecnologías y *Frameworks* seleccionados. Para ello se utilizó la técnica de programación de a pares, sesiones de “training” y ejercicios específicos para que nivelar la capacidad de desarrollo de todos los miembros del equipo.

Integración y Versionado

Durante el desarrollo se encontró que uno de los puntos críticos donde se estaba perdiendo tiempo era en la fase de integración del trabajo realizado por los distintos programadores.

Para poder mitigar esto, se resolvió cambiar el sistema de versionado y mejorar el proceso de SCM.

Visibilidad de la arquitectura

Se entendió como crítico para el éxito del desarrollo del producto que todas las decisiones arquitectónicas y los lineamientos acordados sean totalmente visibles para todos los desarrolladores, de modo que puedan aplicar su juicio en la toma de decisiones tomando en cuenta la estructura final del sistema.

5.4.2. Decisión del lenguaje de desarrollo

Para elegir el lenguaje de desarrollo que fuera más apropiado al producto deseado, se realizó un análisis comparativo entre los lenguajes conocidos por el equipo. Estos eran Java, .Net y Python.

Elegir Java fue decisión de todo el equipo, pero sin embargo, la opinión del arquitecto tuvo más poder que el resto del equipo ya que poseía más conocimientos en temas de arquitectura.

Para poder aliviar la carga de investigación que requirió este proyecto, se optó por el lenguaje de programación más conocido y utilizado por el equipo: Java. Además de ser un lenguaje con un entorno de desarrollo gratis y ligero, Java es un lenguaje de programación orientado a objetos.

Dentro de la variedad de opciones y tecnologías que ofrece el mundo Java, se optó por diseñar una solución JEE, en su versión 7, aprovechando las herramientas que provee el *stack* de tecnologías asociado a esta, como por ejemplo JPA (Java *Persistance* API) para acceso a datos, EJB (*Enterprise Java Beans*) para el *Backend* y JSF (Java Server Faces) para el *Frontend*.

5.4.3. Decisión del servidor de aplicaciones

En el contexto de construir una aplicación JEE, un punto significativo es la elección de un servidor de aplicaciones para ejecutar y contener los artefactos de Software creados.

El mencionado servidor debe poseer, además de las capacidades estándar, algunas características particulares que son prioridad para el equipo de desarrollo:

- Ser Liviano
- De fácil instalación
- De uso intuitivo
- Con posibilidad de despliegue en caliente
- Que permita la configuración de *datasources* y colas sin mayor complicación.

Opciones del mercado

Se listan aquellos servidores con compatibilidad JEE (*Java Enterprise Edition*), preferentemente en versión 6 o más cuya licencia sea libre:

Servidor	Empresa	Versión	Fecha de lanzamiento	Versión JEE
Geronimo	ASF	3.0.1	2013-05-28	6
Glassfish	Oracle	4.0	2013-05	7
JBoss	Red Hat	6.0.2	2013-12-05	6
Jetty	Eclipse	9.1.5	2014-05-05	7(Parcial)
JEUS	TmaxSoft	7	2012-06	6
TomEE	ASF	1.6.0	2013-11-20	6(WEB)

Tabla 5-1 Servidores con compatibilidad JEE

Glassfish y factores que llevan a esta elección

Dentro de la amplia gama de servidores disponibles, Glassfish no solo tiene el soporte de Oracle y una gran comunidad, sino que es un servidor que se utiliza habitualmente en trabajos de materias en la facultad, por lo que existe en el equipo experiencia en el servidor que se puede aprovechar.

Este servidor no tiene soporte comercial, sin embargo se mantiene vigente en la comunidad dada su completa y madura implementación del *stack* JEE.

5.4.4. Decisión del *Framework* de persistencia

Dadas las características del proyecto y las tecnologías utilizadas, el paradigma de programación a usar va a ser Orientación a Objetos y junto con esto y la tecnología de soporte que es *Java Enterprise Edition*, el estándar JPA (Java *Persistence* API) es un enfoque que reúne los siguientes atributos:

- Estándar maduro
- Gran soporte en la comunidad Java
- Gran cantidad de implementaciones
- Desacoplado de las implementaciones
- Soporte integrado en la mayoría de los IDEs

Ventajas de desarrollo

El estándar JPA es de uso muy simple, y permite una gran capacidad de expresión mediante el uso de anotaciones. Dentro de JPA se maneja el concepto de Entidades para denominar un conjunto de datos relacionados, que pude traducirse por ejemplo en una tabla que refleja el estado de diferentes objetos. De esta manera la Entidad se vuelve una suerte de proxy entre la representación como objeto pertinente al mundo POO (Programación Orientada a Objetos) y la representación como tupla en el mundo relacional. Una Entidad no es más que un POJO con anotaciones específicas. Un ejemplo clásico puede ser el siguiente:

```
@Table
public class Empleado {
    @Id
    @Column
    private long pk;
    @Column
    private String nombreEmpleado;
    @Column
    private String direccionEmpleado;
    //Código
 }
```

Mediante estas anotaciones la implementación de JPA traduce el objeto a tabla y viceversa.

5.4.5. Elección de otras tecnologías

Ambiente de desarrollo

Para el desarrollo de la aplicación Web se utilizó NetBeans 8.0.2 [19]. Se comparó con Eclipse y se llegó a la conclusión de elegir NetBeans porque provee más automatizaciones en las tareas que se iban a realizar. Ya viene preparado con *Plugins*, sin la necesidad de agregarlos lo que puede generar más configuraciones.

El equipo no poseía experiencia en otras herramientas populares como ser IntelliJ, por lo que no se exploraron esas opciones.

Librerías de terceros

Se utilizó la librería de PrimeFaces [20] que permite facilitar la creación de aplicaciones Web, dado que provee un conjunto de componentes que extienden las funcionalidades de los componentes básicos de JSF (Java Server Faces) y acelera la tarea de desarrollo.

Asimismo se utilizó la librería Log4J para gestionar *logs* de auditoría del sistema.

Adicionalmente se utilizó el *Framework* JUnit para la automatización de pruebas unitarias.

Base de datos

Para la persistencia de datos se utilizó una base de datos MySql, en su versión 6.2. Se optó por esta base de datos siguiendo la misma línea de elección de tecnologías de desarrollo fundamentada en dos pilares: ser una herramienta *open source*, experiencia del equipo en la tecnología y gran soporte por la comunidad de la misma.

La base estuvo alojada en el mismo servidor que el ambiente de desarrollo, no obstante en un futuro se pretende mover la base a un servidor físico diferente de modo de favorecer la seguridad e integridad del sistema.

5.5. Solución arquitectónica

5.5.1. Definición

La arquitectura definida se corresponde con un diseño separado en tres capas físicas, favoreciendo el bajo acoplamiento y la alta cohesión. Las capas seleccionadas fueron la de persistencia, negocio e interfaz, manteniendo la dependencia en un solo sentido y apoyándose en las APIs del *stack* de JEE (Java *Enterprise Edition*) para mantener esta relación. Asimismo se utilizó el patrón de diseño MVC para bajar a tierra estas tres capas y mapearlas con componentes propios de la tecnología.

Ilustración 5-2 Capas físicas

5.5.2. Cumplimiento de atributos de calidad

Se aplicaron distintas tácticas y patrones arquitectónicos para asegurar el cumplimiento de los atributos de calidad que se definieron, se listan a continuación los más relevantes según atributo de calidad:

Escalabilidad

Para asegurar la escalabilidad del sistema y que la performance se mantenga ante una demanda que se prevé incremental, la arquitectura definida se nutrió de tácticas específicas como incrementar la eficiencia de los recursos en el marco del control de demanda de recursos [21]. Esto se logró mediante un cuidadoso uso de las operaciones críticas del sistema como el acceso a base de datos, minimizando la complejidad de las operaciones llevadas a cabo y optimizando las mismas mediante cálculo de complejidad de algoritmos y consultas a base de datos por ejemplo.

La tecnología de base, JEE (*Java Enterprise Edition*), provee también una gestión de recursos centralizada en los componentes EJB (Enterprise Java Beans), introduciendo concurrencia y manejo de la disponibilidad de recursos.

Disponibilidad

Para asegurar la disponibilidad y la integridad de los datos, se vale principalmente de un API del *stack* JEE llamado JTA (Java *Transaction* API), que se encarga de encapsular en transacciones las operaciones llevadas a cabo en la capa de negocio permitiendo la vuelta atrás si estas fallan. Asimismo, los EJB (Enterprise Java *Beans*) en un contexto gestionado como lo es el de una aplicación JEE son instanciados por el contenedor JEE a demanda según un *pool* de EJBs que puede variar en tamaño para atender más solicitudes.

Seguridad

En materia de seguridad el sistema está provisto con una gestión de usuarios basada en roles. Fue diseñado de esta manera para introducir los beneficios de las técnicas de identificación de Actores, Autenticación de Actores y Acceso Limitado [21].

Se ideó un sistema de seguridad interno compuesto por usuarios, cuentas y roles. Determinado usuario puede tener asociadas varias cuentas y dentro de cada una de ellas tiene un rol particular.

Un usuario operando el sistema está restringido a los contenidos disponibles para el rol que tiene dentro de una cuenta, por lo que sus acciones quedan limitadas.

Accesibilidad

El sistema está basado en el acceso mediante internet, en una arquitectura cliente-servidor. En este marco, se implementan las funcionalidades en la capa de presentación siguiendo buenas prácticas de diseño responsivo, lo que permite el acceso desde cualquier dispositivo con un navegador web.

Modificabilidad

Como se mencionó anteriormente, la modificabilidad fue un atributo considerado crítico para el buen desarrollo del sistema. Se aplicaron varias técnicas así como buenas prácticas para llegar a un producto que permitiera una gran flexibilidad a la hora de extender sus funcionalidades con el menor impacto posible. Se Aplicó táctica de reducción del tamaño de los módulos en conjunto con la incrementación de cohesión para llegar a piezas de Software minimales y auto contenidas, de modo que los cambios en estas se propaguen lo menos posible.

Se aprovechó también que la tecnología de base, JEE (*Java Enterprise Edition*), provee mecanismos para favorecer la técnica "*defer bindings*" dado que los contendores son quienes gestionan la creación y ciclo de vida de los recursos, como por ejemplo los EJB o *Managed Beans* mediante la inyección de dependencias usando la tecnología CDI (*Contexts and Dependency Injection*) de Java, que además favorece la reducción del acoplamiento.

5.5.3. Cumplimiento de los requerimientos no funcionales

El estándar definido en el proyecto para la especificación de requerimientos no funcionales definía la agrupación de estos por atributo de calidad, lo que permite una correspondencia entre las decisiones arquitectónicas que clarifica la trazabilidad entre aquellas que permiten el cumplimiento de un requerimiento no funciona.

De esta forma los requerimientos no funcionales son puntualizaciones en el marco del cumplimiento de los atributos de calidad. A continuación se muestra un resumen de los principales requerimientos no funcionales que soluciona la definición de arquitectura:

Manejo de sesiones

La tecnología de base provee soporte de sesiones mediante la gestión de contextos y contenedores.

Capacidad de procesamiento

Como se mencionó anteriormente, se aplicaron diferentes tácticas que junto con el soporte de la tecnología de base, optimizan la capacidad de procesamiento.

Pruebas unitarias para cada *Feature*

Mediante la utilización del *Framework* JUnit se logró implementar un conjunto de pruebas unitarias para cada *Feature* implementado.

Manejo de errores

Se siguieron buenas prácticas de diseño mediante la utilización de excepciones y manejo de errores para encapsular estos y mantener al sistema responsivo ante fallas no críticas.

5.6. Vistas

5.6.1. Vista lógica

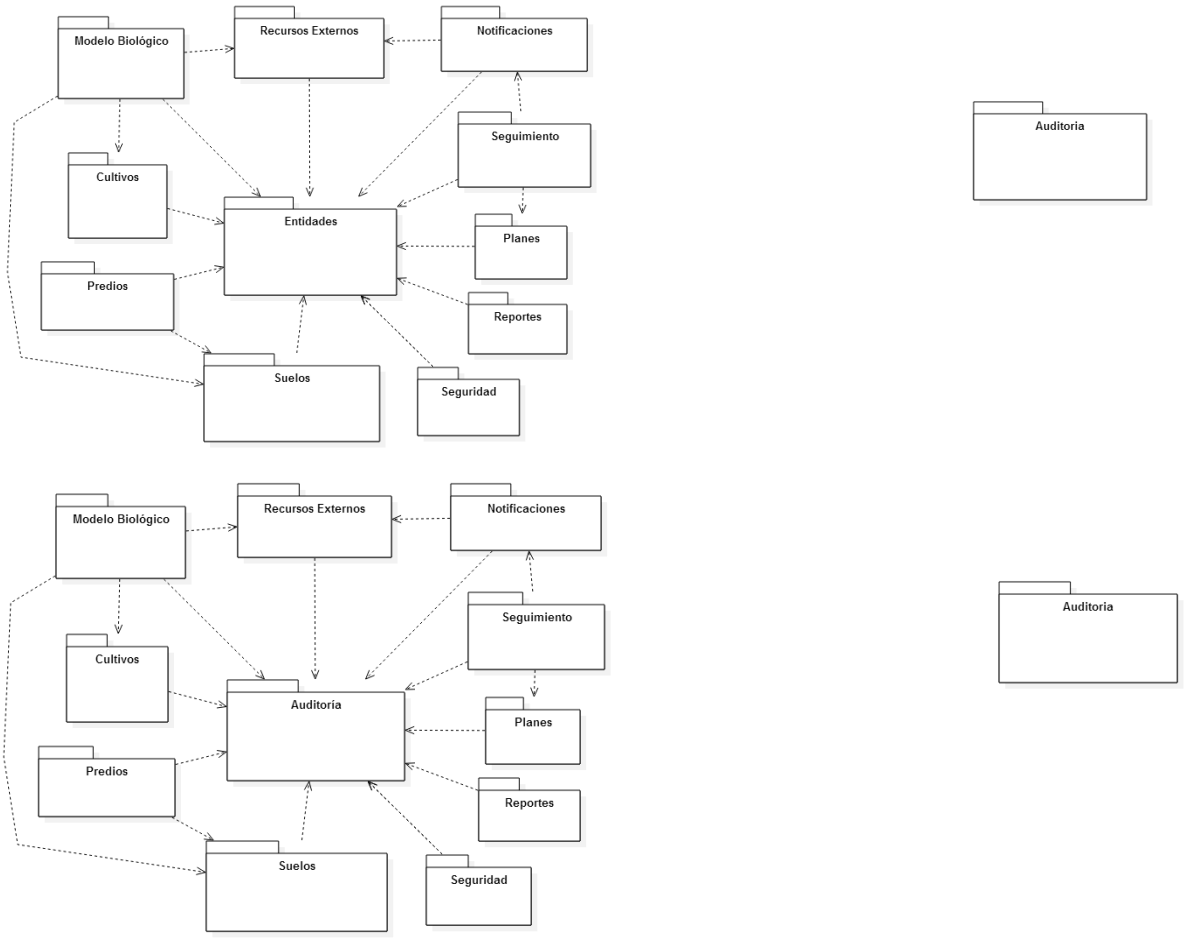


Ilustración 5-3 Vista lógica

5.6.2. Vista de componentes

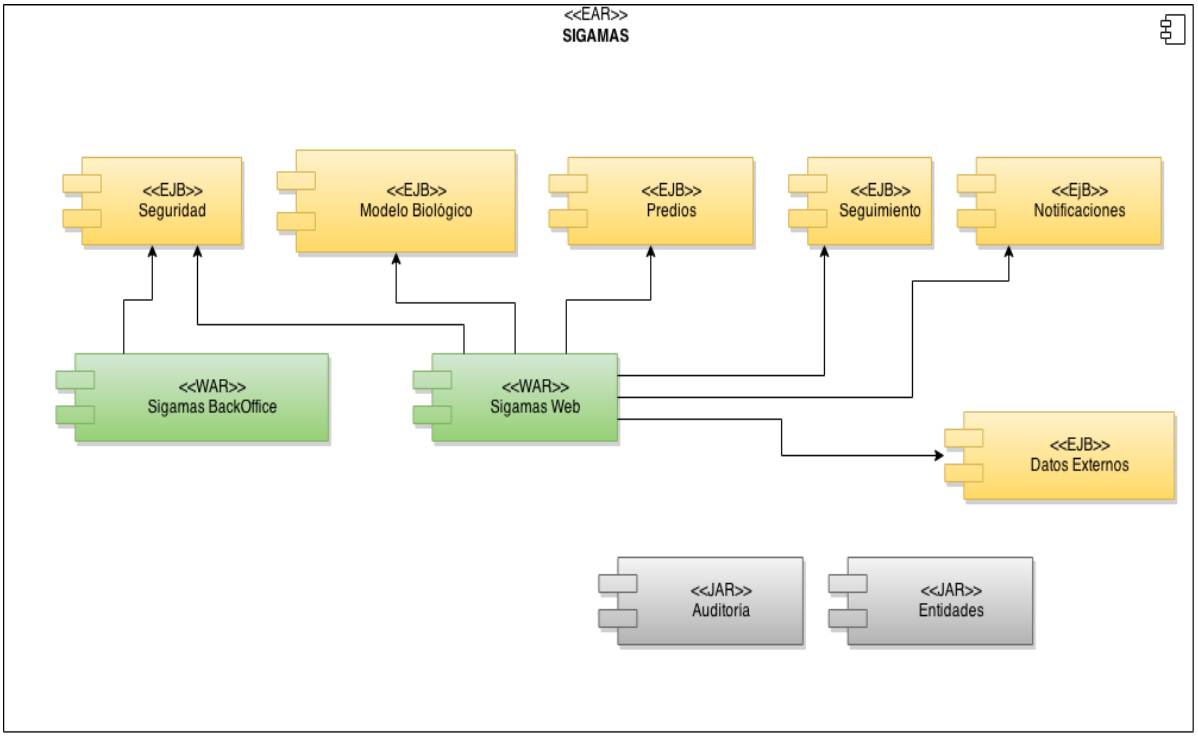


Ilustración 5-4 Vista de componentes

5.7. Evaluación de Arquitectura

La arquitectura se evaluó tanto con el cliente como con un arquitecto experto, quien fue durante todo el proyecto Nicolás Fornaro.

.

6. TESTING

Este capítulo describe las estrategias, procesos, flujos de trabajo, herramientas y metodologías apropiadas que ayudaron a planificar, organizar, ejecutar y gestionar la prueba de Software de SIGAMAS.

Pretende cubrir las áreas de prueba relevantes al sistema, agrupadas según nivel de prueba (pruebas de fábrica, pruebas de integración, pruebas de usabilidad, etc.).

Se describirán las herramientas y metodologías utilizadas para realizar las pruebas, conjuntamente con los formatos para documentar los resultados y las métricas e indicadores relevantes.

6.1. Importancia de las pruebas

En la ingeniería de Software, las fallas pueden aparecer en cualquier etapa del proceso de desarrollo, por lo que es importante revisar y evaluar el producto en todas las etapas de su construcción.

Citando a Myers, “Probar es una tarea extremadamente creativa e intelectualmente desafiante” [22]. El objetivo de las pruebas es encontrar defectos y verificar el comportamiento del producto. La prueba nunca termina. El objetivo es seguir probando hasta la obsolescencia de SIGAMAS.

Para este proyecto, se generaron casos de pruebas específicos para los requerimientos del sistema.

Los casos de prueba y sus resultados se pueden observar en los Anexos “Casos de prueba” y “Resultados de las Pruebas” respectivamente.

6.2. Niveles de prueba

Cómo indica el proceso, es necesario establecer niveles de prueba. Estos niveles deben ajustarse al proyecto siendo ejecutado. Para este proyecto se definieron 6 niveles de prueba que se describirán a continuación. Para todas las pruebas, además de ejecutar casos de prueba, se utilizó la herramienta Selenium que permite la automatización de las pruebas.

6.2.1. Pruebas Unitarias

Las pruebas unitarias permitieron establecer un conjunto de pruebas automatizadas para un fragmento de código, en particular una clase.

Fueron diseñadas e implementadas por el Desarrollador, debiendo estas cubrir toda la lógica nueva incorporada. Si el requerimiento a probar no era crítico, quedaba a criterio del Desarrollador la profundidad de las pruebas unitarias, pudiendo ser verificadas y modificadas por otro desarrollador en otro momento si se consideraba conveniente. En caso

de un requerimiento crítico, el Desarrollador debía seguir un conjunto de casos de prueba previamente establecido.

6.2.2. Pruebas de Fábrica

Pruebas realizadas por el Desarrollador que aseguraron que la pieza de Software que se estaba entregando mantenía un nivel de calidad aceptable.

Se realizaron sobre cualquier pieza de Software completa.

Se consideraba válida una prueba de fábrica si:

- El código compilaba.
- Todo el programa compilaba.
- Se ejecutaron correctamente todas las pruebas unitarias.
- Se ejecutó la pieza de Software sin fallas aparentes, recorriendo al menos un flujo básico completo que utilice la lógica de la nueva pieza de Software.
- Se ejecutó la pieza de Software en conjunto con todos los artefactos con los que interactúa sin que se presentaran fallas aparentes, recorriendo al menos un flujo básico completo que utilizara la lógica de la nueva pieza de Software.

6.2.3. Pruebas cruzadas

Las pruebas cruzadas se llevaron a cabo por un Desarrollador no involucrado en la construcción de la pieza de Software creada, quien tomaba para esta actividad el rol de Tester.

Las pruebas cruzadas fueron el producto de:

- Diseñar casos de prueba en base a los requerimientos y prototipos.
- Repetir los pasos de la prueba de fábrica, ejecutando la pieza de Software en base a los casos de prueba diseñados.
- Generar resúmenes de prueba
- Validar la correctitud de la pieza de Software en base a los criterios de Prueba dispuestos en ese documento.
    - En caso que no se validara la correctitud debía reportarse, según el caso, con los criterios y mecanismos descritos en este documento.
- Validar la correctitud de la pieza de Software en base a los criterios de Prueba de iteraciones anteriores.

6.2.4. Pruebas de usabilidad

Las pruebas de usabilidad estaban a cargo del Tester, quien no tenía por qué ser el mismo que realizó las pruebas cruzadas, y de los Expertos del dominio.

Las pruebas de usabilidad se llevaron a cabo tomando en cuenta los criterios y herramientas pautados en este documento, y se realizaron sobre un entregable completo.

Quedaba a cargo del Tester designado y del SQAer acordar el criterio con el cual cada entregable de Software quedaba validado en cuanto a usabilidad.

6.2.5. Pruebas de regresión

Las pruebas de regresión estaban a cargo del Tester, quien además de realizar los otros tipos de prueba previamente mencionados, debía ejecutar casos de prueba de las iteraciones anteriores, para verificar la consistencia del sistema. Esto se realizaba durante la prueba cruzada cómo se mencionó previamente. El resultado de estas pruebas, no se registró en ningún documento. Si un resultado no es el deseado, se le notificaba al Desarrollador del módulo con falla y ese debía arreglarlo.

6.2.6. Prueba de liberación

Las pruebas de liberación fueron realizadas por todos los integrantes, que en dicho momento tomaron el rol de Tester. Estas pruebas se realizaron sobre la versión final del producto, este una vez liberado al final de cada iteración.

6.3. Automatización de pruebas

Para este proyecto, se hizo uso de la herramienta Selenium, que permite ejecutar pruebas en un entorno especialmente diseñado para aplicaciones web, de modo de poder acelerar el proceso de prueba.

6.4. Clasificación de fallas

A cada falla encontrada se le asignaba una clasificación que dependía de la gravedad de la misma.

La clasificación era la siguiente:

Graves
Fallas en las cuales se genera una caída del sistema, y el usuario se encontraba inhabilitado para seguir utilizándolo.

Se consideraba que una falla era grave cuando:

- El sistema se caía por completo, inhabilitando cualquier operación
- Se perdían datos que fueron guardados
- Se generaban inconsistencias en intercambios de datos

Medias

Fallas en las cuáles se podía observar un error al realizar alguna actividad pero no se generaba ni pérdida de datos ni inhabilitación del sistema.

Se consideraba que una falla era media cuando:

- El usuario realizaba una acción y el resultado no era el esperado
- El usuario no podía realizar una acción

Leves

Fallas en las cuales el sistema seguía funcionando y no había pérdida de información. Se consideraba que una falla era leve cuando:

- Había fallas en la página (visuales, faltas de ortografía, UI mal generada)
- Cuando el sistema no le notificaba al usuario que se estaba procesando algo (rueda de espera)
- Cuando no se tomaban en cuenta campos obligatorios
- Cuando se trataban a campos comunes cómo obligatorios

6.5. Documentación de las pruebas

Para este proyecto, se documentó en cada ejecución de pruebas, las fallas detectadas, la versión del Software en la que fueron detectadas, de manera de admitir el análisis causal de las mismas.

Además se registraron los pasos realizados hasta la aparición de las fallas, el entorno y la versión en la que ocurrieron, de manera de poder reproducir posteriormente estas fallas para el análisis y búsqueda de soluciones pertinentes.

Para ver los detalles de esta documentación, dirigirse al Anexo de “Resultados de las Pruebas”.

6.6. Conclusiones

Se tiene claro que, aunque se haya invertido tiempo en las pruebas, los niveles definidos no son suficientes para un proyecto de tal magnitud. Cómo se decidió entregar una versión Beta, se establecieron estos 6 niveles pero a futuro se seguirá escalando para poder tener toda la prueba exhaustiva del producto que lo vuelva apto para ser liberado al mercado.

7. GESTIÓN DEL PROYECTO

En esta sección se pretende explicar cómo se realizó la gestión del proyecto. En primer lugar se describirá cómo se adaptó el proceso en el momento de ejecutarlo. Luego se explicará el ciclo de vida aplicado y finalmente se desarrollarán las etapas de planificación, ejecución y el seguimiento realizado.

7.1. Adaptación al proceso

Cómo se detalló en la descripción del proceso (véase Anexo "Definición del proceso"), el mismo está compuesto por una serie de actividades claves relacionadas a la vez con Scrum y FDD. Estas actividades se ejecutaron exitosamente a lo largo de las iteraciones. El largo fijo de 2 semanas se respetó siempre y no generó ningún inconveniente. Esto se debe a la fase de planificación que se describe más adelante. Para poder administrar las etapas del proceso recurrimos a las herramientas AgileFant y Google *Drive*.

Dada la actividad laboral del equipo, se tenía una disponibilidad horaria particular. Esto implicaba que no se podía asegurar una dedicación diaria constante al proyecto. Esto respaldó la decisión de no aplicar las *Daily Meetings* ya que la organización de las mismas resultaba ser muy compleja y sin resultados significativos. En los *Sprints* en los cuales se aplicó Scrum puro, las *Daily Meeting* se hicieron vía *Whatsapp*. Esto no resultó ser eficiente ya que en general, no todos podían estar conectados al mismo tiempo.

Las ceremonias de FDS (*Feature Driven* Scrum) se aplicaron todas con éxito y fueron muy útiles para realizar el seguimiento y generar los resúmenes de los *Sprints*.

Para ver los resultados de estas ceremonias y resúmenes dirigirse a los Anexos de "Resultados de las ceremonias" y "Registro de las Métricas" respectivamente.

7.2. Ciclo de vida del proyecto

El ciclo de vida del proyecto fue Incremental. Esto significa que los requerimientos fueron relevados, identificados, especificados y validados antes de empezar las iteraciones.

Es importante aclarar que en cada iteración, los requerimientos volvieron a ser analizados por si surgían cambios o algunas funcionalidades llegaban a volverse obsoletas. Además, en esta etapa se realizó la elección de requerimientos a desarrollar. Esto se ejecutó en la etapa "Definición de *Features*" definida en el proceso.

Se utilizó un ciclo de vida incremental porque al principio del proyecto, ORTsf pidió listar las funcionalidades que iba a ofrecer el producto, para que se valide el emprendimiento por ORTsf. Después de realizar esta lista, (que se podría haber asociado a una lista de *Epics* si se habría seguido con la metodología Scrum), se especificaron todos los requerimiento. El objetivo era priorizar una arquitectura estable y poco cambiante y no relevar requerimientos en cada iteración. Este último punto fue importante porque, cómo se especifica en el Capítulo de "Ingeniería de requerimientos", relevar no fue una tarea fácil. Realizarla en cada iteración tenía cómo riesgo retrasar el cronograma.

El ciclo de vida incremental se puede observar en el Anexo de "Definición del proceso".

7.3. Planificación

Según el proceso, la planificación es la primera etapa que se repite iteración tras iteración. Consiste en preparar el *Sprint* de principio a fin. Para poder realizarla, el equipo se reunía físicamente al empezar cada una de las iteraciones (un *Sprint*) y se ejecutaron las actividades detalladas en el siguiente punto.

Estas reuniones cumplían con la ceremonia de *Sprint Planning* de Scrum definida en el proceso.

7.3.1. Proceso de planificación - *Sprint Planning*

En cada iteración se repitieron las siguientes actividades durante la *Sprint Planning*:

- Cada integrante presentaba su disponibilidad horaria para las próximas 2 semanas
- A partir de estas horas disponibles, se calculaba un total de horas de trabajo que se iban a invertir en el *Sprint*
- Listar *Features* a desarrollar: se seleccionaba una lista de requerimientos elegidos a partir de sus *Story Points* y prioridad. Se recuerda que una *Feature* es un conjunto de requerimientos que son dependientes entre sí.
- Revisión de los requerimientos: para mantener la consistencia y realizar ajustes necesarios
- Listar tareas: se dividen las tareas en dos tipos, las sub tareas de los requerimientos y las tareas no asociadas a ninguna *Feature*, como por ejemplo tareas de investigación o documentación. Las tareas no superan las 3 horas ya que esta era la cantidad promedia diaria que podía dedicar cada integrante.
- Estimar el esfuerzo de cada tarea mediante *Planning* Poker.
- Asignar tareas: El Administrador De Tareas asigna las tareas a los integrantes, basándose en la disponibilidad horaria de cada uno.

Es importante destacar que uno de los objetivos de usar FDS era priorizar la colaboración y comunicación. Por eso en muchas ocasiones, las tareas fueron asignadas a varios responsables para trabajar en grupo sobre un mismo tema.

Para la ejecución de los *Sprint*s, se aplicó una política propia: se necesitan completar las *Features* planificadas al 100%, esto se detalla en la parte de métricas.

Herramientas de planificación

Para poder gestionar la *Sprint Planning*, se utilizó la herramienta AgileFant. Se hizo uso también de la técnica de *Planning* Poker. Esta técnica consiste en utilizar cartas de Poker para tomar decisiones. Por más información sobre *Planning* Poker y su aplicación en el proyecto dirigirse al Anexo de "*Planning* Poker y su aplicación".

7.3.2. Gestión del ESRE

En cada iteración, al generarse la lista de *Features*, se necesitaba gestionar el ESRE (Documento de Especificación de Requerimientos). La elección de *Features* a desarrollar dependía de la categorización (Crítico o No Crítico) de cada *Feature* y de su prioridad (Alta, Media y Baja). Los requerimientos fueron elegidos en cada iteración en el siguiente orden:

1. [Crítico, Alto]
2. [No Crítico, Alto]
3. [Crítico, Medio]
4. [No Crítico, Medio]
5. [Crítico, Bajo]
6. [No Crítico, Bajo]

7.4. Ejecución del proyecto

En esta sección se describe de qué forma se ejecutaron las iteraciones del proceso.

7.4.1. Ejecución de los *Sprints*

Cada integrante tenía la responsabilidad de administrar su tiempo en orden de poder cumplir con sus asignaciones. Al realizar una actividad, era obligatorio registrar en AgileFant el esfuerzo dedicado en cada una de nuestras tareas. A continuación se muestra de qué manera se registraban las horas en la herramienta:

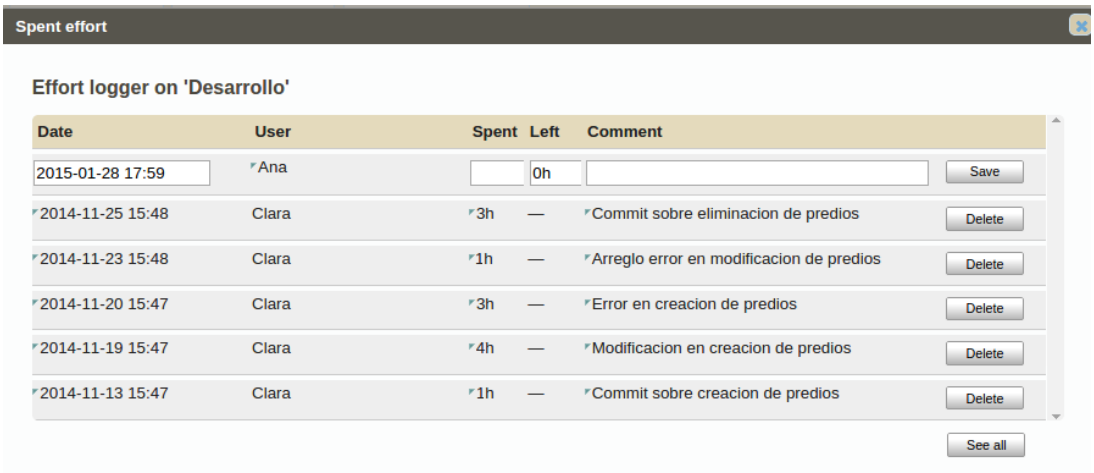


Ilustración 7-1 Registro de horas en AgileFant

En muchas otras ocasiones, un integrante podía encontrarse frente a un problema el cual no podía resolver y verse en la necesidad de pedir ayuda. Un segundo integrante debía entonces, ayudar al primero, y al final registrar el esfuerzo de su ayuda en la tarea del primer integrante.

7.5. Seguimiento

Para poder realizar y analizar el seguimiento del proyecto mientras el mismo se ejecutó, se utilizaron la herramienta definida anteriormente, AgileFant, que permitió registrar el esfuerzo en las distintas *Features* y tareas especificadas por *Sprints*. Estos registros permitieron luego componer las métricas que midieron el avance del proyecto. Al final del proyecto se pudo extraer el resumen completo del avance en función de *Story Points* y tiempo. El mismo está representado en el punto de este capítulo "Registro de las Métricas".

7.5.1. Evaluación y Revisión de los *Sprints* - *Sprint Review*

Al final de cada iteración se ejecutaba la ceremonia, *Sprint Review*. Todo el equipo formaba parte de la reunión, y si era posible, también participaba el Experto del dominio. El propósito de esta ceremonia, era evaluar el progreso realizado en el *Sprint*. Se realizaba mediante reuniones informales y se revisaba lo siguiente:

Features completadas

Se analizaban las *Features* completadas y el resultado de las pruebas sobre las mismas. En caso de no completarse una *Feature*, se discutía la razón y se evaluaban formas de resolver los problemas encontrados. Vale aclarar que esto solamente sucedió en los *Sprints* 3 y 4, al ejecutar el primer proceso basado enteramente en Scrum. En la ejecución del proceso definitivo, esto no sucedió. En el Anexo de "Registro de las Métricas" se detallan estos acontecimientos.

Demo de la nueva versión estable

La demo permitía observar la nueva versión del producto, pudiendo analizar directamente cómo se integraban las nuevas *Features*.

Generación de resúmenes

Gracias a AgileFant, se podían obtener datos numéricos de la ejecución del *Sprint* para generar estadísticas de rendimiento que se transformaron luego en métricas.

Revisión SQA

SQAer era el encargado de realizar una etapa de revisión al finalizar un *Sprint*. Debía analizar los documentos generados o actualizados, el código, las pruebas realizadas para ver si seguían correctamente los estándares y poder generar métricas del producto y por último, era el encargado de revisar las métricas generadas. Si el SQAer consideraba que los documentos no seguían los estándares o tenían algún defecto, registraba esa anomalía en su planilla correspondiente para que se pueda arreglar. A continuación se muestra un ejemplo de estos registros:

Sprint	Priorización	Fecha	Nombre	Descripción	Status
3	Planificar	31/07/2014	Documento de arquitectura	No se actualizó la tabla de revisión	Done
	Planificar	31/07/2014	Plan SCM	No se actualizó la tabla de revisión	Done
	Planificar	31/07/2014	Como documentar el desarrollo	Necesita la tabla de revisión	Done

Tabla 7-1 Ejemplo de planilla de revisión SQA

Además, era el encargado de revisar las métricas generadas por las iteraciones y analizar la eficiencia de la ejecución del proceso. Dependiendo del resultado de las métricas, el equipo se podía reunir para evaluar las ejecuciones no satisfactorias y desarrollar formas de mejorarlas.

A continuación se muestra un ejemplo de la planilla de revisión de métricas.

Sprint	Fecha	Nombre	Lectura de la métrica	Decisión tomada
4	19/08/14	*Sprint* 4 *Burndown*	Siguen habiendo atrasos, aunque estos se disminuyeron	El equipo debe ir mirando los *Burndowns* mientras se ejecuta el ciclo y no solamente al final.
	19/08/14	Velocidad *Sprint* 4	No se pudieron completar todas las *Story Points*.	Aumentar compromiso del equipo aún más. Evaluar otras metodologías ágiles ya que Scrum no está produciendo los resultados esperados.
	19/08/14	Retrabajo	Se generó un gran porcentaje de retrabajo debido a la mala ejecución del *Sprint* 3	No se realizarán acciones adicionales. Se considera que aumentando el compromiso del equipo y analizando otras metodologías esto se podría solucionar.

Tabla 7-2 Ejemplo de los resultados de evaluación de las métricas generadas

Para ver en detalle estos resultados, dirigirse al Anexo "Resultados de las Revisiones SQA".

7.5.2. Retrospectiva de los Sprints - Sprint Retrospective

Después de realizada la *Sprint Review*, se efectuaba la *Sprint Retrospective*. Esta reunión tenía los siguientes objetivos [23].

- Mejorar continuamente la calidad del producto y del proceso.
- Mejorar la productividad
- Mejorar la capacidad
- Aumentar la capacidad

Todos participaban de ella, y se actualizaba un documento de retrospectiva, con la respuesta a las siguientes preguntas:

¿Qué se hizo y se debe seguir haciendo?

Define cuáles son las actividades que se realizaron y tuvieron éxito. A cada actividad que se encontrara en esta categoría se le asignaba el atributo *Continue*, que significaba que está actividad era positiva y se debía continuar haciéndose.

¿Qué se hizo y debe dejar de hacerse?

Define cuales son las actividades que se realizaron y no tuvieron éxito. Fueron actividades previstas o que surgieron al improviso. A las mismas, se les asignaba el atributo *Stop Doing* que significa "Dejar de hacer".

¿Qué se hizo pero debe mejorarse?

Define cuales son las actividades que se realizaron, tuvieron éxito pero podrían tener aún más. Se les asignaba el atributo *More Of* que significa "Más de".

¿Qué se hizo pero debe hacerse menos?

Define cuales son las actividades que se realizaron, pero en las cuales se invirtieron demasiado tiempo comparado al beneficio de sus resultados. Se les asignaba el atributo *Less Of* que significa "Menos de".

¿Qué no se hizo y debe hacerse?

Define cuales fueron las actividades que no se realizaron, pero habrían sido muy beneficiosas para aumentar la calidad del producto y/o del proceso. Se les asignaba el atributo *Start Doing* lo que significa "Empezar a hacer".

A continuación se muestra un ejemplo de la tabla de retrospectiva utilizada por el equipo, en la cual se registraron las actividades sobre las cuales actuar.

Sprint	*What?*	*What to do?*	*Why?*
5	Mejorar organización horaria	*More of*	Se consideró mejor la disponibilidad pero hay que mejorar
	Mejorar división de tareas	*More of*	Se dividió aún más pero faltaron tareas
	Buena Comunicación	*Continue*	Segundo *Continue*
	Estimación de horas	*More of*	Hay que profundizar la estimación porque hubo una gran diferencia entre lo estimado y lo real en este *Sprint*
	Daily Meetings	*Stop Doing*	Las *Daily Meetings* no se están respetando pero esto no está afectando el desempeño, por lo que no son necesarias
	Horas de *Features*	*Less of*	En este *Sprint* hubieron demasiadas horas invertidas en *Features*, debe bajar

Tabla 7-3 Retrospectiva del Sprint 5

Cómo se puede observar, el documento se fue creando incrementalmente, con el objetivo tener un estado *Continue* en todas las actividades. Por eso no solamente se revisó lo del *Sprint* recién terminado, sino lo que había sido reportado anteriormente. En la figura estas actividades aparecen marcadas en verde claro en la columna *What* que significa "Que".

Por último, durante la *Retrospective*, pero cada 2 *Sprints*, se realizaba la gestión de riesgos cómo se describe a continuación.

Para ver la lista completa de actividades, dirigirse al Anexo "Resultados de Ceremonias".

7.6. Gestión de riesgos

Durante la ejecución del proyecto, fue importante realizar la "Gestión de riesgos" para controlar la posible ocurrencia de los riesgos definidos durante la *Retrospective*.

Para ello, se identificaron los riesgos, que luego se analizaron y se registraron en la misma planilla de revisión de *Sprints*, indicando la fecha de comienzo del mismo (en qué momento se definió), quién lo definió y su grado de impacto. El grado de impacto se calculó gracias a la clasificación de respuesta de los riesgos definida en el proceso.

Esto permitió observar su evolución y poder ordenarlos de mayor a menor impacto para saber cómo actuar. Gracias a esto, se controló y aplicó un proceso de seguimiento sobre los riesgos. Vale aclarar que solamente se consideraron los riesgos con consecuencias negativas. Los riesgos con consecuencias positivas no fueron analizados pero sí tomados en cuenta informalmente.

A continuación se muestra la evolución de todos los riesgos definidos en el proyecto y luego se detallarán los puntos importantes de la gestión de los mismos.

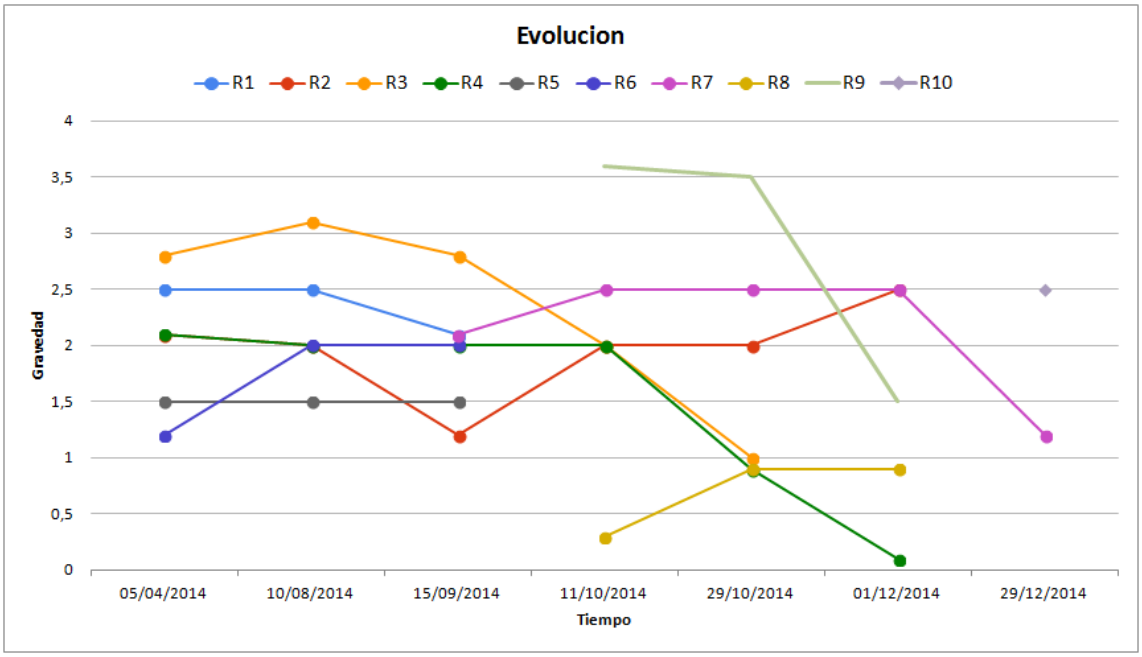


Ilustración 7-2 Evolución de la gravedad de los riesgos a lo largo del proyecto

R1: No tener la información necesaria para poder realizar los planes de tareas

R2: Definir un alcance demasiado grande para la entrega del proyecto

R3: No aceptación por parte de los usuarios

R4: Conflicto en el equipo

R5: No poder enfrentarnos a las tecnologías y acceso a datos necesarios

R6: Sistema poco *User Friendly*

R7: Cambios en el proceso

R8: Cambios en los requerimientos

R9: No tener un proceso definido

R10: No tener las métricas adecuadas

7.6.1. Descubrimiento de Riesgos

Todos los meses (cada 2 *Sprints*) se revisó la lista especificada en el proceso, para detectar eventuales nuevos riesgos.

- Especificaciones inconsistentes, incoherentes o incompletas.
- Responsabilidades pobremente definidas
- Pobre eficiencia de un producto
- Comunicación poco efectiva
- Control de cambios débil

Se decidió realizar esta actividades cada 2 *Sprints* para aliviar la carga de *Sprint Planning*.

7.6.2. Acción sobre los riesgos

En este proyecto, según los datos obtenidos en el análisis cualitativo de los riesgos, se le atribuyó a cada uno, una estrategia. Estas estrategias fueron:

Aceptar
Cuando un riesgo es de bajo impacto y probabilidad de ocurrencia, no se debía realizar acción alguna sobre él.

Evitar
Cuando un riesgo debe ser evitado, implica eliminar por completo su amenaza. Es decir, lograr que no ocurra.

Mitigar
La acción de mitigar, reduce el impacto y/o la probabilidad de ocurrencia de un riesgo. No lo elimina. Se realizan acciones tempranamente, así, si el riesgo ocurre, su impacto no es tan importante o simplemente, tiene menos oportunidad de ocurrir. La mitigación se realiza porque se considera que es menos costoso mitigar un riesgo que luego corregir su ocurrencia.

A continuación se muestra la tabla de Acción sobre los riesgos

Prob. de Ocurrencia **Impacto**	**0,1**	**0,3**	**0,5**	**0,7**	**0,9**
1	Aceptar(0,1)	Aceptar (0,3)	Aceptar (0,5)	Aceptar (0,7)	Mitigar (0,9)
2	Aceptar (0,2)	Aceptar (0,6)	Aceptar (1,0)	Mitigar (1,4)	Evitar (1,8)
3	Aceptar (0,3)	Aceptar (0,9)	Aceptar (1,5)	Mitigar (2,1)	Evitar (2,7)

4	Aceptar (0,4)	Aceptar (1,2)	Mitigar (2,0)	Evitar (2,8)	Evitar (3,6)
5	Aceptar (0,5)	Mitigar (1,5)	Mitigar (2,5)	Evitar (3,5)	Evitar (4,5)

Tabla 7-4 Acción sobre riesgos

7.6.3. Tipos de planes de contingencia definidos

Para cada tipo de respuesta a riesgo, se definió una medida de contingencia.

Para los riesgos que necesitaban ser mitigados se elaboró un plan de contingencia que redujera el impacto y/o probabilidad de impacto de los mismos.

Para los riesgos que necesitaban ser evitados, se elaboró un plan de contingencia que evitara la ocurrencia y/o el impacto de los mismos. Estos planes se encuentran en el Anexo "Plan de Gestión de Riesgos".

7.7. Métricas del proyecto

Realizar métricas a lo largo del desarrollo del proyecto fue muy importante para tener medidas cuantitativas que pudieran mejorar el producto, proceso y proyecto. Posibilitaron realizar un seguimiento de los *Sprints* y asignarles valores que ayudaran a controlar sus desempeños y mejorarlos continuamente.

Se siguieron las 3 métricas definidas en el proceso, estas fueron la Velocidad por *Sprint*, los *Sprints Burndowns* e Indicador de retrabajo.

En el Anexo "Registro de las Métricas" se muestran los resultados de estas métricas.

Gracias a AgileFant que brinda la posibilidad de exportar los resultados de cada *Sprint*, se generaron las métricas. Para ello se utilizó una planilla Excel con los datos exportados y a partir de los mismos se produjeron las métricas. Esto se realizaba al final de cada *Sprint*, en la *Sprint Review*.

La primera métrica, fue el *Sprint Burndown*. Esta consiste en marcar la diferencia entre el *Burndown* ideal y el real.

Para ilustrar esta idea se muestra el *Burndown* del *Sprint* 9 (Los *Burndowns* de los otros *Sprints* se encuentran en el Anexo "Registro de las Métricas"):

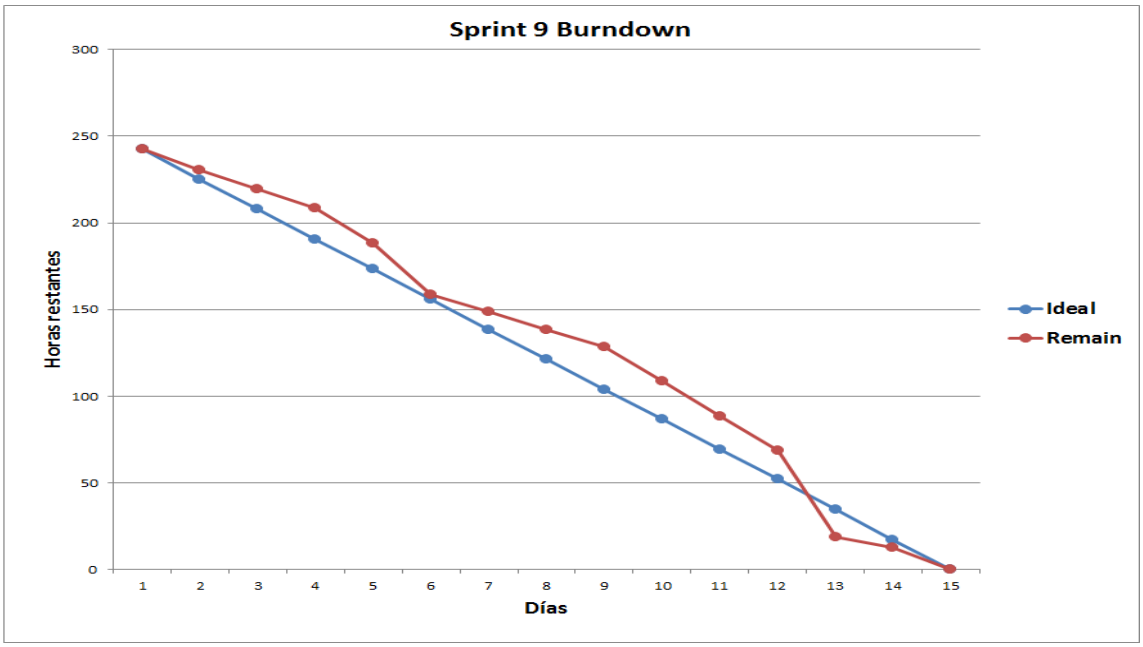


Ilustración 7-3 Sprint 9 *Burndown*

El eje de las X representa la cantidad de días del *Sprint*.

El eje de las Y representa la cantidad de horas planificadas para ese *Sprint*. Se realizaron los *Burndowns* a partir de las horas y no de las *Story Points* porque resultó ser un método más directo y rápido al momento de registrar, ya que cada tarea tenía asignado una cantidad de horas.

Cómo se puede observar, el *Burndown* cuenta con dos líneas.

La primera línea, la azul, es el *Burndown* ideal. A partir de las horas estimadas al principio del *Sprint* y hasta llegar a 0, se muestra cómo gastar estas horas diariamente (si se trabajara la misma cantidad de horas por día).

La segunda línea, la roja, es el *Burndown* real. Muestra la cantidad de horas estimadas restantes. Empieza en el mismo punto que la línea real y va decreciendo a medida que se van completando tareas. Al completarse una tarea, se resta su valor en horas, a las horas estimadas restantes, es decir con el siguiente cálculo:

Punto = Horas estimadas restantes – Suma (Cantidad de Horas Estimadas de tareas completadas ese día).

Esta gráfica permitió saber lo siguiente:

- Si la línea roja se encontraba por encima de la azul, significaba estar en una zona de peligro, ya que representaba estar retrasado.
- Si la línea roja se encontraba por debajo de la azul, significaba estar en una zona segura, ya que representaba estar adelantado. En este caso, se podía avanzar en tareas futuras. Sin embargo, se decidió invertir ese tiempo de sobra en revisión y documentación.

7.7.1. Velocidad

Cómo se definió en el proceso, la velocidad puede evaluarse en Horas o *Story Points*. Para este proyecto, la velocidad se analizó en *Story Points*. Los resultados fueron más concretos en este caso, por qué registrar la velocidad en horas puede ser menos representativo de la cantidad de trabajo realmente concretado. Por ejemplo, se podrían haber invertido muchas horas en un *Sprint* pero habiendo completado un pequeño porcentaje de las *Story Points* planificadas, y en un gráfico de velocidad con horas esto puede presentar falsas conclusiones.

Para ilustrar esta idea, se muestra un ejemplo. Al principio del proyecto, al no tener proceso definido todavía, se había decidido optar por la métrica "diferencia entre horas estimadas y horas reales por *Sprint*". Sin embargo, en una revisión, el revisor preguntó si esta métrica realmente representaba al trabajo realizado, y la respuesta fue no. Sugirió cambiarla ya que no aportaba resultados significativos con respecto al desempeño. Se combinaron estos datos, con los datos de la velocidad, para saber si se estaba estimando acertadamente pero que además estas horas estaban siendo invertidas correctamente.

A continuación se muestra dicha gráfica:

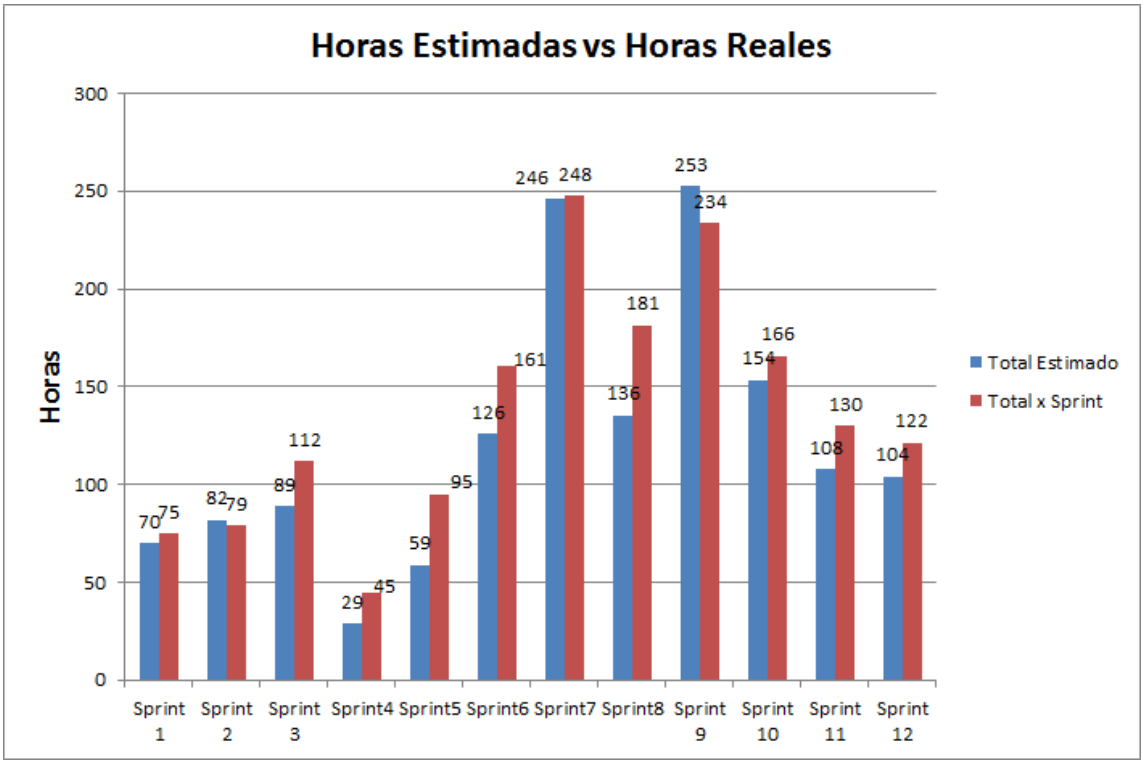


Ilustración 7-4 Horas estimadas vs Horas reales

Se siguieron completando los valores de esta métrica luego de la revisión, simplemente para marcar el error del equipo y se cambió a la gráfica de velocidad.

Vale aclarar, que la meta principal en cada *Sprint* era cumplir con todos los *Story Points* definidos, lo cual se cumplió en todas las iteraciones a partir del *Sprint* 5. Esto se debe a que la política del equipo en comprometerse a completar todas las *Features* en cada *Sprint* (mencionada en el punto de este Capítulo de Planificación). Si una *Feature* se veía atrasada

(información obtenida gracias a los *Burndowns*), se invertían más horas en el *Sprint* para lograr alcanzar la finalización de la(s) *Feature*(s). Por eso en la gráfica de Horas Estimadas vs Horas Reales, en algunos *Sprints* se invirtieron más horas de lo estimado.

7.7.2. Indicador de retrabajo

En cada *Sprint* se registraba el retrabajo gastado en horas. En AgileFant era una tarea llamada "Retrabajo" sin cantidad de horas estimadas. Con retrabajo, se entiende por la necesidad de realizar ajustes sobre tareas que pertenecían a *Sprints* anteriores. Esto permitió saber si los *Sprints* se ejecutaban correctamente.

7.8. Conclusiones

La gestión del proyecto permitió poner a prueba el proceso definido y ver si la ejecución resultaba exitosa.

Afortunadamente, esto resultó ser cierto y se concluye entonces que las metodologías evaluadas y el proceso ingeniado fueron exitosos en su aplicación. Se puede decir que se podrán seguir aplicando en el futuro, para el crecimiento de SIGAMAS u otros proyectos nuevos.

Se aprendió mucho de la ejecución del proyecto, sobre todo en tema de organización. Tener un proceso, y que su ejecución sigua los pasos definidos, ayudan a obtener mejores resultados y poder controlar el desarrollo del producto.

Para el futuro, se mejorará la eficiencia de las reuniones, para que estas sean más cortas y concisas. Algunas reuniones se extendían demasiado por discusiones varias. Para más detalles sobre la distribución de estas reuniones dirigirse al Anexo "Resultados de las ceremonias".

7. GESTIÓN DE LA CALIDAD

Para producir un sistema de la mejor calidad, es necesario que se cumpla y si se puede, se supere todas las expectativas de los usuarios sobre el sistema. El usuario final medirá dicha calidad respecto a lo que tenga o no, y se dependerá de cómo lo juzgue.

A continuación se detallan los principales aspectos para el aseguramiento de la calidad del producto, que se ejecutaron durante este proyecto según lo establecido en el proceso.

7.1. Objetivos de la calidad

Para la obtención de un producto de calidad fue necesaria la utilización de metodología o procedimientos estándares para el análisis, diseño, programación y prueba de Software que permitieron uniformar la forma de trabajo, de forma de lograr un aumento en la confiabilidad, mantenibilidad, productividad, tanto para el desarrollo como para el control de la calidad del Software.

De esta forma se buscó lograr cumplir el objetivo principal el cual era cumplir con todas las necesidades y requerimientos de los usuarios en el sistema.

7.2. Definición del plan de calidad

El plan de calidad es un documento en el cual se buscó establecer los responsables, fases, herramientas, técnicas, indicadores y documentación que se usaron para asegurar la calidad del producto.

En el mismo se especificó de forma clara cuál es el propósito, la organización y los estándares que se usaron.

7.3. Definición del plan de aseguramiento de calidad

Aseguramiento de la calidad: Conjunto de actividades planificadas y sistemáticas necesarias para aportar la confianza en que el producto (Software) satisfaga los requisitos dados de calidad.

7.3.1. Actividades de SQA

Roles y responsabilidades: se identificaron las personas responsables del aseguramiento de la calidad indicando nombres, el rol que tienen y las responsabilidades que tienen en el proyecto, ya que así fue establecido en el proceso.

A continuación se detallan los roles que se definieron para este proyecto y las actividades correspondientes a cada uno:

SQAer

Fue el encargado de realizar el plan de aseguramiento de la calidad y las revisiones correspondientes para velar por el cumplimiento de dicho plan. Sobre las revisiones y auditorías, indicó en qué momentos y qué elementos se revisarían durante el desarrollo del proyecto. Realizó las tareas de especificación de estándares a seguir para lograr la calidad. Definió las métricas y cómo se obtendrían, especificando los indicadores junto con sus parámetros de interpretación. Indicó que metodología se utilizaría y qué herramientas se usarían junto con las técnicas que apoyen el cumplimiento de la metodología de trabajo.

SCMer

Fue el encargado de administrar los documentos de SQA, los documentos resultados de las revisiones y aquellos donde se registraban los incidentes. La documentación se debía especificar de manera clara y objetiva con toda la información necesaria que se generaría y utilizaría en cada fase del proyecto.

Administrador de tareas

Fue el encargado de asegurar que SQAer registrara todo su esfuerzo en la herramienta AgileFant, para luego poder obtener las métricas.

Experto del dominio

Fue el encargado de validar las funcionalidades del sistema de modo de asegurar la calidad del sistema.

7.3.2. Estándares

Los estándares son los niveles mínimo y máximo deseados, o aceptables de calidad que debe tener el resultado de una acción, una actividad, un programa, o un servicio. En otras palabras, el estándar es la norma técnica que se utiliza como parámetro de evaluación de la calidad.

Una vez programadas las actividades de solución al problema de gestión, los círculos de calidad se debieron definir los estándares de calidad del resultado, o los resultados esperados.

Se utilizaron los siguientes estándares para lograr un producto de calidad:

FASE	ESTÁNDARES
Ingeniería de requerimientos	Documentos 302, 303, 304, 306 y 307 Estándar de Especificación de Requerimientos (propio)
Diseño y Arquitectura	Arquitectura "*Views and Beyond*"

	Patrones "*Gang of Four*" www.dofactory.com/net/design-patterns
Desarrollo	Buenas prácticas de ORACLE Estándar de Codificación (propio)
Testing	Documento de plan de *testing*, guía generada por el equipo
Aseguramiento de la calidad	IEEE 2008 std. 1028-1008, IEEE *Standard of* Software *Reviews and Audits.* IEEE Std 730.1-1995, IEEE *Guide for* Software *Quality Assurance Planning* IEEE Std 1012-1998, IEEE *Standard for* Software *Verification and Validation* Documento 302, 303, 304, 306 y 307

Tabla 7-1 Estándares utilizados

Para ver en detalle los estándares de Especificación de requerimientos y codificación dirigirse a los Anexos "Estándar de Especificación de Requerimientos" y "Estándar de Codificación" respectivamente.

7.3.3. Métricas del producto

Para poder evaluar la calidad del producto, se definieron métricas. Las mismas se pueden ver en detalle en el Anexo "Registro de las Métricas". Estas se basaron en las pruebas del sistema cuyo objetivo principal fue el de comprobar que el sistema operara de la manera deseada y cumpliera todos los requerimientos pautados. Si bien en esta instancia se generó una versión Beta, fue importante probar el sistema para que no poseyera fallas importantes, es decir de categoría grave. En el capítulo de *Testing* se profundiza en la categorización de fallas. La aparición de fallas graves impedía ejecutar correctamente las principales funcionalidades del sistema.

Se seleccionaron las siguientes métricas ya que se cree que fueron las que mejor se adaptaron al tipo de producto, el cual apuntó a un mercado objetivo particular y no hacía solo un cliente.

Nivel de Usabilidad

Se realizó una encuesta de usabilidad a expertos del dominio, se analizaron sus resultados para determinar el nivel de usabilidad de las diferentes funcionalidades del sistema. La encuesta fue realizada cada dos semanas al finalizar cada *Sprint*. Esta encuesta se empezó a realizar a partir del *Sprint* 5 ya que fue en este momento que el equipo la definió. Se le

transfería al Experto del Dominio, una demo del producto que él probaba y luego llenaba la encuesta. La usabilidad se midió con una nota del 0 al 5, siendo 0 una muy mala usabilidad y 5 una muy buena usabilidad. Además de estos valores, todos los *Sprints*, el equipo evaluaba cuál era el valor mínimo aceptable de los resultados de las encuestas. Estos fueron los resultados.

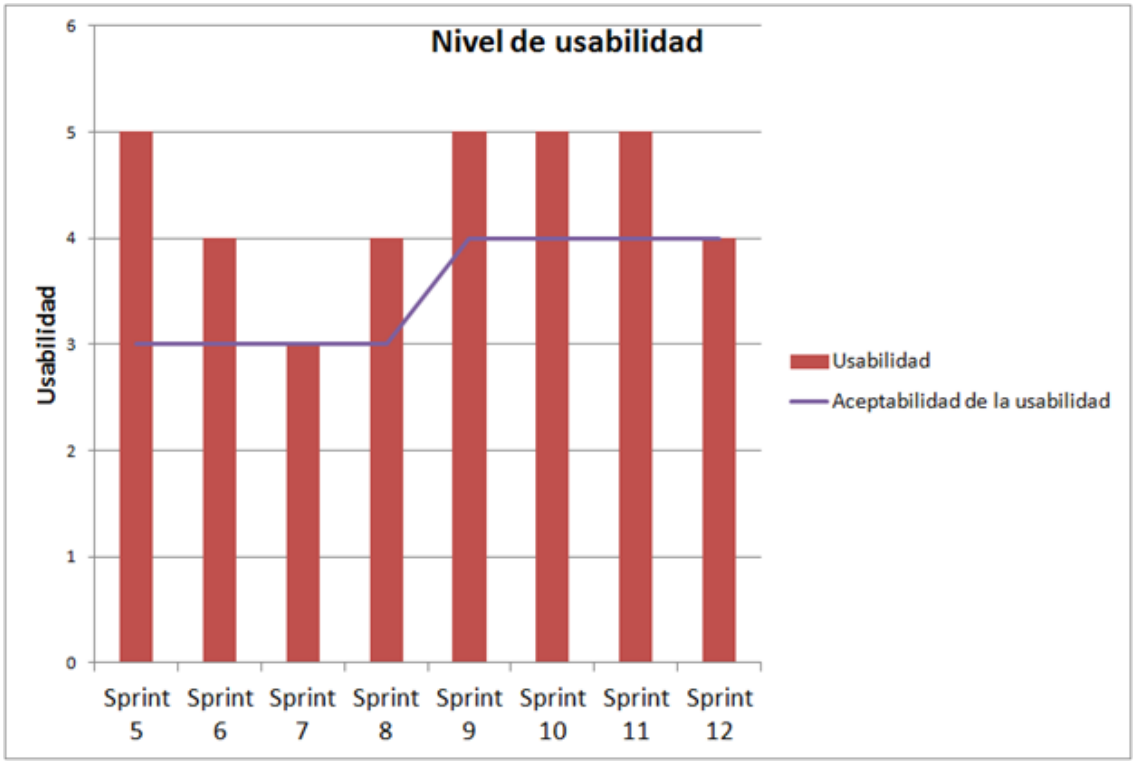


Ilustración 7-1 Resultados encuesta Nivel de Usabilidad

Para obtener más detalles sobre los resultados de esta encuesta, dirigirse al Anexo "Resultados de las Revisiones SQA".

Aceptación del producto

Esta métrica al igual que la usabilidad, surgió como resultado de evaluar los resultados de una encuesta a los expertos del dominio. También fue realizada cada dos semanas y se analizó haciendo un promedio de la aceptación general a lo largo de todo el proyecto. Al igual que la usabilidad, la aceptación se midió con valores del 0 al 5, siendo el 0 "Muy poca aceptación" y el 5 "Muy bien aceptado", y además el equipo definía en cada *Sprint* el valor mínimo a cumplir. Estos fueron los resultados:

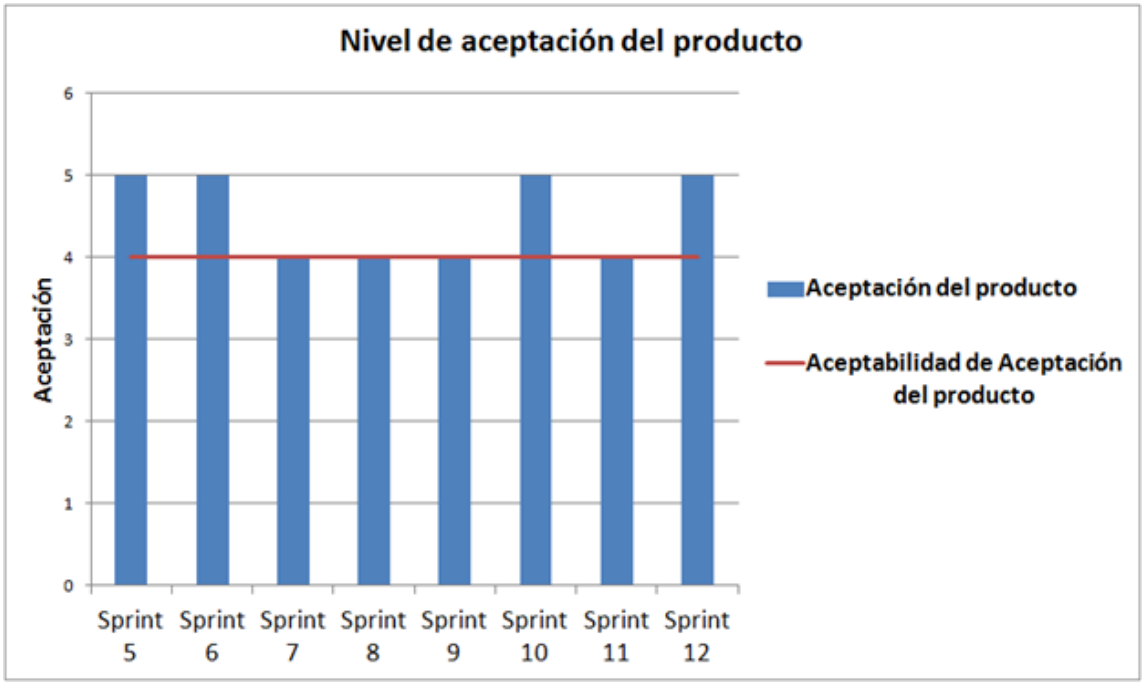


Ilustración 7-2 Resultados encuesta Nivel de Aceptación del producto

Para obtener más detalles sobre los resultados de esta encuesta, dirigirse al Anexo "Resultados de las Revisiones SQA".

Fallas por Sprint

Se calculaba la cantidad de fallas que poseía determinado *Sprint* y se mostraba el porcentaje respecto a la cantidad de casos de prueba iniciales. Luego se analizaba en un gráfico como fueron evolucionando las mismas. A continuación se muestran estos resultados:

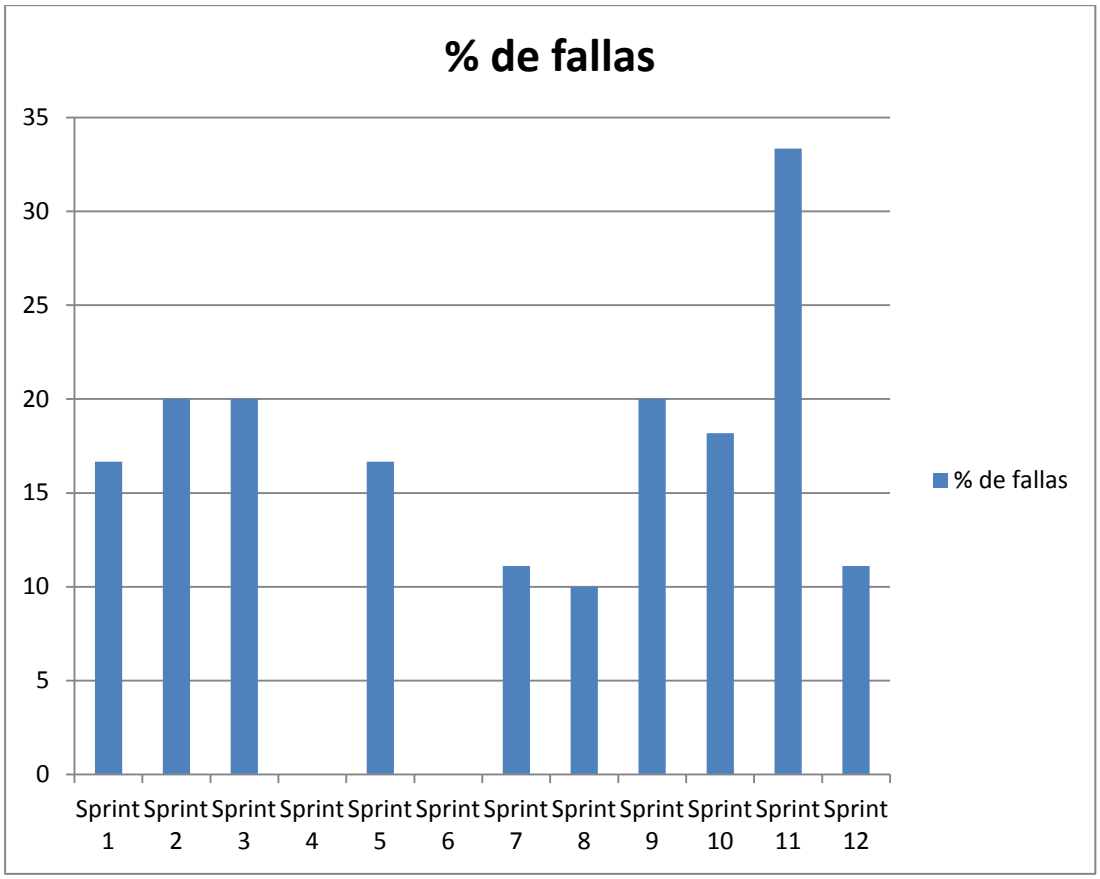


Ilustración 7-3 Porcentaje de Fallas por Sprint

Para obtener más detalles sobre estos resultados, dirigirse al Anexo "Resultados de las Revisiones SQA".

7.4. Registro de incidentes

Un incidente es un comportamiento anómalo que se presenta al realizar el proyecto, lo cual puede producir desviaciones en la planificación. Se debe saber cómo actuar frente a la aparición de éstos hechos.

Lo primero que se realizaba era reportar la incidencia. La cual se comunicada al SQAer y luego al resto del equipo. Luego se realizaba un análisis del impacto de dicha incidencia. Se presentaba la solución al incidente y por último se dejaba registro por escrito del acontecimiento.

7.5. Revisiones de ORTsf

Se planteó la existencia de dos tipos de revisiones durante el proyecto. Las mismas se dividen en las revisiones internas al proyecto y las externas. Las revisiones externas son tres y se realizan por un profesor de la Universidad, el cual luego de presenciar una muestra de

los avances, aconseja y asiste brindando posibles mejoras desde sus conocimientos y experiencia.

El detalle de la devolución de estas revisiones se encuentra detallado en el Anexo "Devolución de las Revisiones".

7.5.1. Primera revisión

La primera revisión, se realizó con el Ing. Nicolás Fornaro el 1ero de Julio 2014. En esta revisión se presentó al primer avance del proceso, producto y proyecto. En esta versión se contaba con prototipos para presentar un panorama general del producto. Además, en este momento el equipo utilizaba el primer proceso definido, el cual fue presentado en esta revisión. En esta revisión se marcaron las fortalezas y aspectos a mejorar del equipo, proceso, producto y proyecto.

Por más detalles sobre resultados de esta revisión dirigirse al Anexo "Devolución de las Revisiones".

7.5.2. Segunda revisión

La segunda revisión la realizó el Lic. Marcelo Cagnani el 9 de setiembre del 2014. En esta revisión se presentó al segundo avance del proceso, producto y proyecto. En este avance, se presentó al proceso definitivo del equipo. Al igual que en la revisión anterior, se anotaron las fortalezas y aspectos a mejorar. Por más detalles sobre resultados de esta revisión dirigirse al Anexo "Devolución de las Revisiones".

7.5.3. Tercera revisión

La tercera y última revisión, la realizó el Ing. Rafael Bentancur el 16 de diciembre del 2014. Para esta revisión el equipo ya contaba con el producto en sus etapas finales, funcionando y con detalles por arreglar. Al igual que las revisiones anteriores, el revisor marcó las fortalezas del equipo y los aspectos a mejorar.

Por más detalles sobre resultados de esta revisión dirigirse al Anexo "Devolución de las Revisiones".

7.6. Reunión con expertos

Debido a que el proyecto tuvo como temática central un área en la cual se carecía de conocimiento, se decidió realizar reuniones con expertos del dominio. Varias reuniones fueron necesarias mantener, para lograr un producto de calidad.

En el plan de calidad se detallan cuáles fueron las reuniones realizadas a lo largo de todo el proyecto. Para ver estos detalles, dirigirse al Anexo "Plan de Aseguramiento de la Calidad (SQA)".

8. GESTIÓN DE LA CONFIGURACIÓN

El propósito de la gestión de la configuración (SCM) fue de establecer y mantener la consistencia de las piezas del Software a lo largo de su ciclo de vida. Se aplicaron procedimientos técnicos y administrativos para identificar, definir las piezas de Software; controlar modificaciones y versiones de estas piezas; registrar y reportar el estado de cada pieza y las solicitudes de modificaciones.

8.1. Identificación de los elementos de configuración

Según el proceso, los elementos de configuración (ECS) son todos aquellos elementos que formen parte de la gestión de configuración, siendo estos elementos:

- Códigos fuentes
- Datos
- Ejecutables
- Documentación

A continuación se describe cuáles son estos elementos que se administraron en este proyecto:

Elemento	Tipos
Códigos fuentes	Clases Java: .class Elementos web: .xhtml, .css, .js, .config
Datos	Base de datos Imágenes y recursos web
Ejecutables	Java: .jar, .war, .ear
Documentación	Documentos word: .docx Documentos excel: .xls Documentos de Google Drive Imágenes: .jpg, .png, .ico

Tabla 8-1 Elementos de configuración identificados

8.2. Repositorios

Para poder almacenar los distintos tipos de ECS, se evaluaron 3 herramientas para el repositorio de documentos y 3 herramientas para el repositorio de fuentes. Después de un análisis comparativo entre las distintas herramientas, que se basó en usabilidad, completitud y posibilidad de concurrencia sobre documentos, se llegó a la conclusión de utilizar el repositorio Google *Drive* [24] para los documentos y GitHub [25] para el manejo del código.

A continuación se muestra el resumen comparativo entre los distintos repositorios.

Repositorio	Concurrencia	Gratis	Permite comentarios
Google Drive	✔	✔	✔
	✘	✔	✘
OneDrive	✔	✔	✘

Tabla 8-2 Repositorios para los documentos

Repositorio	Gratis	Manejo de problemas	Interfaz amigable	Protocolo Git
GitHub	✔	✔	✔	✔
Google code	✔	✔	✘	✘
assembla	✘	✔	✔	✔

Tabla 8-3 Repositorios para el código

Para la documentación del proyecto, el equipo decidió utilizar Google *Drive* ya que permite la concurrencia en la visualización y edición de documentos publicados en el mismo, lo que permitió trabajar en forma colaborativa y simultánea. Esto fue un punto muy importante porque permitió una comunicación constante y poder realizar correcciones cruzadas instantáneamente. Además, Google *Drive* permite la inserción de comentarios dentro de los documentos lo que agilizó la resolución de fallas y realización de mejoras (eficientemente). Para ver la estructura y descripción del repositorio en Google *Drive* ver el Anexo de "Plan de SCM".

8.2.1. Repositorios de fuentes

Para el desarrollo de la aplicación se decidió optar por GitHub ya que utiliza el protocolo Git. El equipo decidió utilizar un repositorio que implementara el protocolo Git porqué era una forma de trabajo desconocida por todos por lo que era interesante aprender algo nuevo. Para más detalles sobre esta decisión, dirigirse al Anexo "SVN vs Git".

En la página principal del repositorio online, se puede acceder a la lista de *Branches*, *Commits* y visualizar las diferentes carpetas y el último *Commit* que se realizó sobre la misma. A continuación se muestra el repositorio online en GitHub.

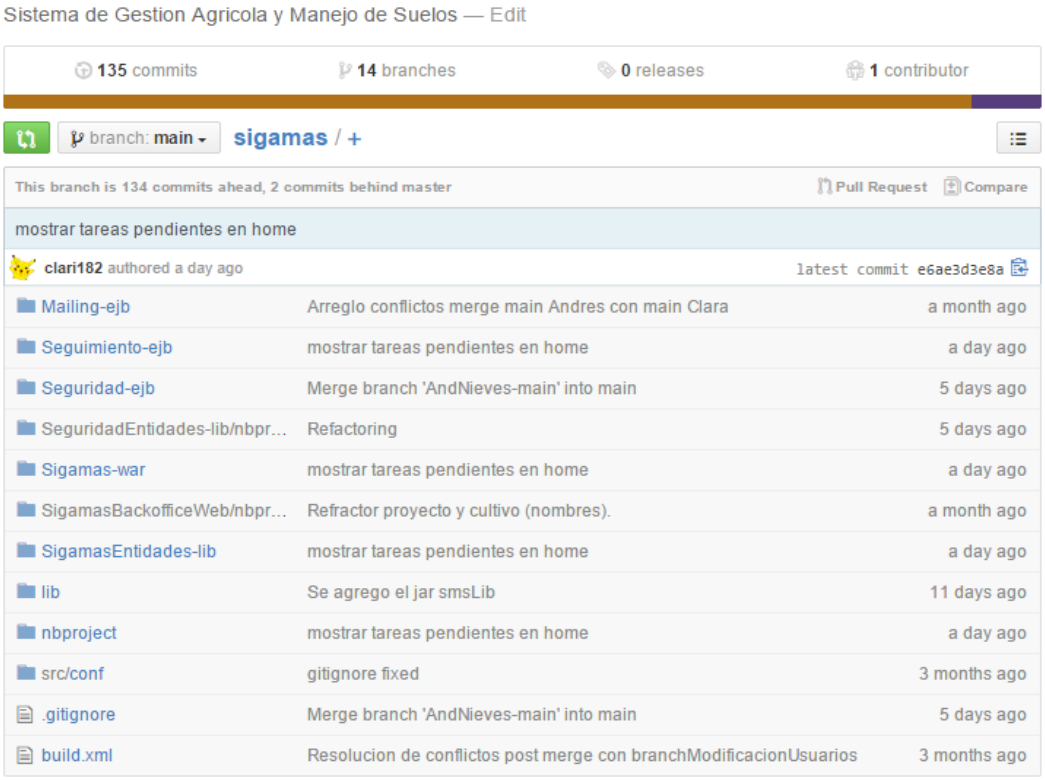


Ilustración 8-1 Repositorio GitHub

Para ayudar al equipo a manejar el repositorio en Github se creó un manual con los pasos a seguir. Por más detalles de cómo se utiliza Github en Netbeans, ver el Anexo "Manual Netbeans para ejecutar el Plan SCM".

8.3. Control de cambios

A medida que se iba construyendo el producto, surgía la necesidad de realizar cambios sobre el mismo. Estas peticiones provenían del equipo o de los validadores al momento de revisarlo (el producto).

Según Pressman, controlar los cambios que surgen es una actividad muy importante, ya que los cambios no controlados pueden convertirse en caos [26].

Cada cambio se calificó según su importancia y su urgencia. En un documento Google *Sheet* (llamada "Control de Cambios") se registraron los cambios, y la acción que se debía tomar en cada uno de ellos.

A continuación se muestra la tabla que se utilizó para determinar la acción a realizar sobre una petición de cambio. Para realizar esta tabla, se evaluaron dos formas de valorar los cambios:

1. Atribuyéndoles una prioridad (Leve, Moderado, Alto, Urgente)

2. Clasificándolos con dos parámetros: Urgencia e Importancia

Estas dos posibilidades fueron discutidas por el equipo y finalmente se decidió optar por la segunda opción (mediante *Planning* Poker). Cada nuevo cambio podía ser Urgente o No urgente e Importante o No importante. Para cada par de clasificación se decidió realizar una acción apropiada.

A continuación se muestra la acción para cada atributo.

	URGENTE	NO URGENTE
IMPORTANTE	Actuar	Planificar
NO IMPORTANTE	Delegar	Rechazar

Ilustración 8-2 Priorización de los cambios

A continuación se muestra el proceso que se utilizó para controlar las peticiones de cambios.

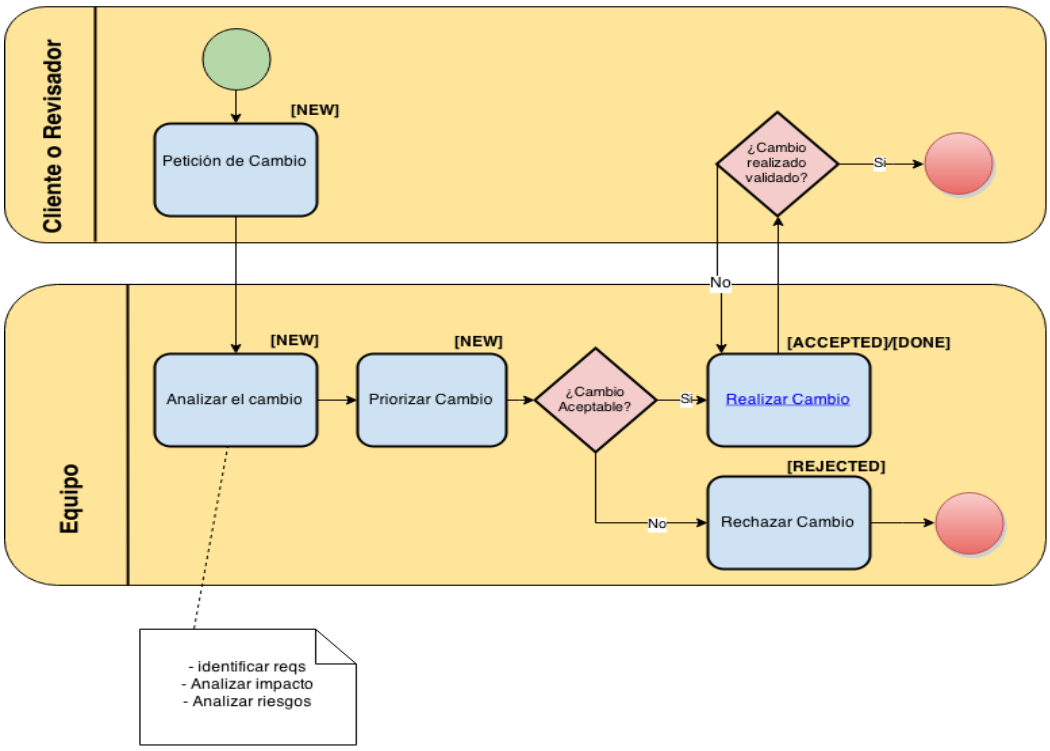


Ilustración 8-3 Proceso de control de Cambios

Por una explicación más detallada de este proceso de cambios dirigirse al Anexo "Plan de SCM".

Para ver la bitácora de cambios realizados durante el proyecto, dirigirse al Anexo "Bitácora de Cambios del Proyecto".

9. CONCLUSIONES

En esta sección se describe el resultado que obtuvimos respecto a los objetivos que nos hemos planteado al comenzar el proyecto. Se especifica para cada uno de los objetivos, si el mismo se ha cumplido o no y su justificación. A su vez también se detallan las lecciones aprendidas en distintas áreas del proyecto y una conclusión final que engloba de manera resumida lo acontecido durante el año.

9.1. Lecciones aprendidas

Definición temprana del modelo del servicio

Una definición temprana del modelo de producto o servicio permite definir más fácilmente quienes son los clientes y quienes los usuarios del sistema. Al haber tenido la convicción de que el proyecto sería un *SaaS* desde un comienzo, simplificó las decisiones de diseño y arquitectura del mismo. Álvaro siempre nos alentó a tomar dicha decisión de manera temprana.

Definición de estrategia

Se debe definir una estrategia para la definición del producto de software.

Definición de objetivos

Es imprescindible definir objetivos a corto plazo a modo de poder tener una lectura temprana de cómo se va avanzando hacia ellos. De esta manera poder realizar ajustes a tiempo en caso que los resultados de los objetivos no sean los buscados.

Metodología
Se obtienen mejores resultados con una metodología ideada especialmente para las necesidades actuales de un equipo de trabajo pero que también sea robusta para ser usada en el futuro. Crear una metodología propia permitió tener un proceso bien definido, lo cual se detalla a continuación.

Proceso bien definido

Un proceso bien definido y adaptado a las necesidades del equipo permite un proyecto exitoso. Esto se vio reflejado en la ejecución del proceso. Gracias a la metodología definida, se pudo elaborar un proceso que se ejecutará hasta finalizar SIGAMAS.

Investigación
Al haber elegido un área totalmente desconocida para el equipo, invertir tiempo en investigación permitió elevar la calidad del producto y conocer nuevas tecnologías y rubros. Eso resultó en un gran compromiso y dedicación por parte de cada uno de los miembros que

no habría sido posible sin el apoyo incondicional del tutor y de los Expertos del Dominio que guiaron y ayudaron en momentos difíciles.

9.2. Objetivos cumplidos

Una vez finalizado el proyecto, el equipo está conforme por haber logrado gran parte de los objetivos.

9.2.1. Objetivos de Equipo

Proyecto en equipo

Aprender a realizar un proyecto de cero con 4 integrantes y lograr terminarlo con éxito ya que al principio del proyecto, ninguno de los integrantes había realizado un proyecto académico u obligatorio conjuntamente con otras 3 personas.

El objetivo estará cumplido si todos los integrantes finalizan con una buena relación luego de aprobado el proyecto.

Este objetivo se vio cumplido ya que se pudo realizar un proyecto emprendedor en 12 meses cómo se había deseado.

Aceptación del tema elegido

Lograr un consenso y aceptación en el tema elegido, todos los integrantes deben estar motivados con la idea general del proyecto y que el proyecto satisfaga todas las expectativas personales y académicas. Se realizó una reunión entre los integrantes del equipo donde se evaluaron cada una de las propuestas.

Este objetivo se considera cumplido si el tema es aceptado por todos los integrantes.

Entre todas las propuestas de proyecto que cada integrante del equipo presentó, SIGAMAS fue elegido por unanimidad siendo la opción más motivadora para el equipo.

9.2.2. Objetivos de Proyecto

Versión Beta

Se deseaba obtener una versión Beta sobre la cual se pudiera seguir avanzando en el futuro.

Para medir el éxito de este objetivo se utilizará la métrica “Cantidad de fallas”, las cuales deberían tener cantidad cero de fallas graves como resultados, es decir se aceptan solamente fallas leves o medias.

Este objetivo se cumplió y la versión actual de SIGAMAS es una versión Beta que cumple con los requerimientos esenciales para tener un producto estable y sobre el cual se podrá seguir desarrollando para lograr cumplir con todos los requerimientos especificados inicialmente. La actual versión de SIGAMAS no posee falla de categoría grave.

9.2.3. Objetivos del proceso

Aplicar conocimientos adquiridos

Aplicar los conocimientos adquiridos a lo largo de toda la carrera, es decir aplicar Ingeniería de Software. Todas las materias que se han visto por separado y que enriquecen en determinados aspectos de la Ingeniería de Software, debieron ser realizadas para cada aspecto del proyecto.

Este objetivo se considerará cumplido si el proceso creado se ejecuta con éxito (que las métricas del mismo tengan resultados satisfactorios).

A lo largo de la carrera, se adquirieron todos los conocimientos relacionados a la ingeniería en sistemas. Gracias a este proyecto, estos conocimientos pudieron ser aplicados y medidos. Analizadas las métricas de proceso se han obtenido resultados satisfactorios.

Creación de un proceso

Crear un proceso de desarrollo de software el cual se pueda aplicar para futuros proyectos. Para medir el éxito del proceso, se utilizarán las métricas que se muestran en el Capítulo "Descripción del proceso".

Se considerará cumplido este objetivo si dichas métricas presentan resultados satisfactorios.

Evaluadas las métricas, hemos constatado resultados satisfactorios de nuestro proceso, es decir que el objetivo ha sido cumplido.

9.3. Proyecciones a futuro

Sabiendo que el proyecto es actualmente una versión Beta, la proyección a futuro más importante es de convertir el producto en una versión final, pronta para salir a producción y poder ser comercializada cómo planificado. Se cuenta con el apoyo de nuestros Expertos del Dominio que prometieron no abandonar al equipo y ayudarlo en su futuro y el de SIGAMAS.

9.4. Conclusiones generales

El proyecto es una experiencia única la cual hizo crecer a todos tanto en lo profesional como en lo personal. Fue el primer emprendimiento para todos los integrantes en el cual se tuvo que sortear diferentes obstáculos pero sin duda todo el esfuerzo que se ha invertido ha dado su resultado.

En el futuro el equipo se enfrentará nuevamente con situaciones similares en las cuales ya se habrá adquirido experiencia sobre qué acciones tomar.

10. REFERENCIAS BIBLIOGRÁFICAS

[1] Universidad ORT Uruguay, “Laboratorio ORT Software Factory”, 2013. [En línea]. Disponible:http://fi.ort.edu.uy/innovaportal/v/3393/5/fi.ort.front/inicio.html

[2] Universidad ORT Uruguay, “Laboratorio ORT Software Factory - Visión, Misión y Políticas de Calidad”, 2013. [En línea]. Disponible: http://fi.ort.edu.uy/2012/5/mision_y_vision_.html

[3] Comunidad del Sur, “Impactos de la revolución verde, agricultura convencional”. [En línea]. Disponible: http://www.ecocomunidad.org.uy/coeduca/artic/impactos_verde3.htm

[4] D. Yuravlivke, *Uruguay NOTAS DE POLITICA. Desafíos y oportunidades 2010-2015*, D.C, USA: World Bank, 2010.

[5] Nuevo Manantial S.A. [En línea]. Disponible: http://www.nuevomanantial.com/

[6] Ministerio de Ganadería, Agricultura y Pesca, "El Ministerio de Ganadería, Agricultura y Pesca a través de Estadísticas Agropecuarias (DIEA) comunica: Resultados de la Encuesta Agrícola Invierno 2014”, 2014. [En línea]. Disponible: http://www.mgap.gub.uy/portal/afiledownload.aspx?2,5,3,O,S,0,9373%3BS%3B3%3B36

[7] Wikipedia, “Dolores (Uruguay)”, marzo 2015. [En línea]. Disponible: http://es.wikipedia.org/wiki/Dolores_%28Uruguay%29

[8] Noticias Rurales, "En Uruguay la mayoría de la soja y maíz es transgénica", 2013. [En línea]. Disponible: http://www.noticiasrurales.com.uy/general/nacionales/agricultura/en-uruguay-la-mayoria-de-la-soja-y-maiz-es-transgenica/

[9] FAO, "Nuevas políticas para la adaptación del sector agropecuario al cambio climático en Uruguay", 2010. [En línea]. Disponible: http://www.fao.org/climatechange/80141/es/

[10] Scrum Guide, “The Scrum Guide™”, 2013. [En línea]. Disponible: http://www.Scrumguides.org/docs/Scrumguide/v1/Scrum-Guide-US.pdf#zoom=100

[11] Universidad ORT Uruguay, “Metodología FDD”, 2003. [En línea]. Disponible: http://fi.ort.edu.uy/innovaportal/file/2021/1/metodologiafdd.pdf

[12] Extreme Programming, a gentle Introduction. [En línea]. Disponible: http://www.extremeprogramming.org/

[13] M. Keeton, *Microsoft Solutions Framework (MSF): A Pocket Guide*, USA: Van Haren Publishing, 2006.

[14] Rational Unified Process, “What is Rational Unified Process?”. [En línea]. Disponible: http://www.ibm.com/developerworks/rational/library/content/RationalEdge/jan01/WhatIstheRationalUnifiedProcessJan01.pdf

[15] DSDM, “What is DSDM?”. [En línea]. Disponible: http://www.dsdm.org/content/what-dsdm

[16] G. Kotonya, *Requirements Engineering: Processes and Technique*, USA: John Wiley & Sons Ltd., 1998.

[17] V. Kanabar, *MBA Fundamentals Project Management*, New York, USA: Kaplan Publishing, 2008.

[18] P. M. Institute, *A Guide to the Project Management Body of Knowledge: PMBOK(R) Guide*, 5th Ed., USA: Project Management Institute, 2013.

[19] NetBeans. [En línea]. Disponible: https://netbeans.org/

[20] PrimeFaces. [En línea]. Disponible: http://primefaces.org

[21] L. Bass, P. Clements, R. Kazma, *Software Architecture in Practice*, USA: Addison-Wesley, 2012.

[22] G. Myers, *The Art of Software Testing,* USA: Word Association Inc., 2012.

[23] E. Derby, D. Larsen, *Agile Retrospectives Making good teams Great*, USA: The Pragmatic Bookshelf, 2006.

[24] Google Drive. [En línea]. Disponible: www.google.com/Drive

[25] GitHub. [En línea]. Disponible: http://www.github.com

[26] R. S. Pressman, *Ingeniería Del Software*, 2nd.ed., USA: McGraw-Hill/ Interamericana de México, 2005.

11. ANEXOS

11.1. RESULTADOS DE LAS PRUEBAS

A continuación se lista la ejecución de los casos de prueba establecidos formalmente o por el desarrollador.

Se reagrupan los resultados de las pruebas en una tabla, indicando a que Requerimiento funcional hace referencia (RF), un nombre, la entrada de la prueba, la salida esperada, la salida real y si su comportamiento es aprobado o no. Se presentan separados por Sprint.

Cómo se puede observar, no se detectaron fallas de categoría grave.

Sprint 1

RF	Nombre	Entradas	Salida esperada	Salida	Aprobado	Cat.
Login	E01	Usuario y contraseña existentes y seleccionar "Ingresar"	*Login* correcto	OK	Si	
	E02	Usuario invalido, contraseña valida y seleccionar "Ingresar"	Mensaje de error	OK	Si	
	E03	Usuario valido, contraseña invalida y seleccionar "Ingresar"	Mensaje de error	OK	Si	
	E04	El usuario ingresa un nombre de usuario y contraseña válidos, luego selecciona la tecla "*Enter*".	*Login* correcto	No ingresa	No	Media
	E05	Usuario invalido, contraseña valida	Mensaje de error	OK	Si	
	E06	Usuario valido, contraseña invalida	Mensaje de error	OK	Si	

Sprint 2

RF	Nombre	Entradas	Salida esperada	Salida	Aprobado	Cat.
ABM Parcela	E01	Se selecciona “Crear nueva parcela”	El sistema abre la creación de parcela	OK	Si	
		Se ingresa un nombre, padrón y un departamento				
		Se selecciona “Crear”	El sistema crea una nueva parcela	OK	Si	
	E02	Se selecciona “Crear nueva parcela”				
		Se elige un departamento del combo	Se centra el departamento en el mapa	OK	Si	
	E03	Se selecciona “Crear nueva parcela”				
		No se ingresa padrón				
		Se selecciona “Crear”	El sistema indica que el padrón es obligatorio	OK	Si	
	E04	Se selecciona “Crear nueva parcela”				
		No se ingresa nombre				
		Se selecciona “Crear”	El sistema indica que el nombre es obligatorio	OK	Si	
	E05	Se selecciona una parcela existente				

		Se modifica su nombre				
		Se selecciona “Guardar”	El sistema actualiza la parcela	El sistema actualiza la parcela pero vuelve a la lista de parcelas	No	Leve
	E06	Se selecciona una parcela existente				
		Se modifica su nombre y se lo deja vacio				
		Se selecciona “Guardar”	El sistema indica que el nombre es obligatorio	OK	Si	
	E07	Se ingresa a la lista de parcelas	Mensaje de error	OK	Si	
		Se selecciona una parcela	Se elimina la parcela	OK	Si	
		Se selecciona “Eliminar”	Se despliega un mensaje para confirmar eliminación	No OK, el sistema elimina directamente	No	Media
		Al confirmar se elimina la parcela	El sistema elimina la parcela de la lista	OK	Si	
Invitar usuario por correo	E01	Se ingresa a la sección de invitación por mail	El sistema despliega los elementos para invitar por mail	OK	Si	
		Se ingresa un mail valido				
		Se selecciona “Enviar”	El sistema despliega información sobre el correcto envío del mail	No despliega información pero lo envía	No	Leve

			El sistema envía un mail a la dirección especificada	OK	Si
	E01.1	Se completa con dirección sin formato de correo	El sistema despliega un mensaje de error indicando que el mail ingresado es invalido	OK	Si

Sprint 3

RF	Nombre	Entradas	Salida esperada	Salida	Aprobado	Cat.
Logout	E01	Se selecciona la opción "cerrar sesión"	El sistema cierra sesión	OK	Si	
	E02	Se cierra el navegador				
		Se entra al sistema sin ingresar	La sesión está cerrada	OK	Si	
Ver Información del cultivo	E01	Se ingresa a la lista de cultivos	El sistema muestra la lista de cultivos	OK	Si	
		Se filtran los cultivos por nombre	El sistema solo despliega los cultivos que contienen este nombre	OK	Si	
		Se selecciona un cultivo	El sistema despliega la información del cultivo	Se muestra la información correctamente pero aparece un cartel de error sin información. El mismo no afecta el funcionamiento y se puede cerrar.	Si	Leve

Sprint 4

RF	Nombre	Entradas	Salida esperada	Salida	Aprobado
Elegir cronograma	E01	Se ingresa a la lista de proyectos	El sistema despliega la lista de proyectos del usuario	OK	Si
		Se selecciona “Crear”	El sistema despliega la creación de proyecto	OK	Si
			El sistema permite elegir el cronograma por defecto	OK	Si

Sprint 5

RF	Nombre	Entradas	Salida esperada	Salida	Aprobado	Cat.
Iniciar Proyecto de cultivo	E01	Se ingresa a la lista de proyectos	Se despliega la lista de proyectos	OK	Si	
		Se selecciona “Crear”	El sistema despliega la creación de un nuevo proyecto	OK	Si	
		Se ingresa un nombre valido				
		Se ingresa una fecha de inicio valida				
		Se elige el cronograma por defecto				
		Se selecciona “Guardar”	El sistema genera un nuevo proyecto con sus correspondientes tareas	OK	Si	
	E01.1	No se ingresa nombre	Mensaje de error	OK	Si	
		Se selecciona “Guardar”	El sistema despliega información que el nombre es	No crea el proyecto pero no	No	Leve

			obligatorio	despliega mensaje		
	E01.2	Se ingresa un nombre				
		Se ingresa una fecha inválida				
		Se selecciona “Guardar”	El sistema indica que la fecha seleccionada no es valida	OK	Si	

Sprint 6

RF	Nombre	Entradas	Salida esperada	Salida	Aprobado	Cat.
Visualización de *tabs*	E01	Se ingresa con usuario/contraseña existentes	Las *tabs* están correctamente implementadas	OK	Si	
Cambiar de *tabs*	E01	Se selecciona la *tab* de proyectos	El sistema muestra la lista de proyectos	OK	Si	
	E02	Se selecciona la *tab* de tareas	El sistema muestra la lista de tareas pendientes	OK	Si	
	E03	Se selecciona la *tab* de parcelas	El sistema muestra la lista de parcelas	OK	Si	
	E04	Se selecciona la *tab* principal	El sistema despliega la información de la *tab* principal	OK	Si	

Sprint 7

RF	Nombre	Entradas	Salida esperada	Salida	Aprobado	Cat.
Modificar Planificación de Cronograma	E01	Se selecciona un cronograma	Se abre el cronograma seleccionado	OK	Si	
		Se modifica una tarea del cronograma	Se ingresan datos visibles	OK	Si	
		Se selecciona guardar	Se guarda el cronograma	OK	Si	
	E02	Se selecciona un cronograma	No se guarda el cronograma	OK	Si	
		Se modifica una tarea del cronograma				
		Se selecciona cancelar				
	E03	Se selecciona un cronograma	Se muestra un mensaje de error	El Sistema no guarda la tarea pero no muestra mensaje de error	No	Leve
		Se ingresa una fecha incorrecta en una tarea				
		Se selecciona guardar				
	E04	Se selecciona un cronograma	No se guarda el cronograma	OK	Si	
		Se ingresa una tarea con fecha pasada				
		Se selecciona cancelar				
	E05	Se selecciona un cronograma	Se guarda el cronograma y se recalcula	OK	Si	
		Se modifica la predecesora de una tarea				
		Se selecciona				

		cancelar				
	E06	Se selecciona un cronograma	No se guarda el cronograma	OK	Si	
		Se elimina la predecesora de una tarea				
		Se selecciona cancelar				
	E07	Se selecciona un cronograma	No se guarda el cronograma	OK	Si	
		Se asigna una tarea a un modelo fenológico				
		Se selecciona cancelar				

Sprint 8

RF	Nombre	Entradas	Salida esperada	Salida	Aprobado	Cat.
Administrar notificaciones	E01	El usuario ingresa a la gestión de notificación	El sistema despliega la lista de notificaciones activas y no activas	OK	Si	
		El usuario selecciona el link "Agregar" correspondiente a la tarea que desea obtener notificación	El sistema despliega un *popup* con la información a proveer de la notificación	OK	Si	
		El usuario ingresa el mensaje que desea recibir	--			
		El usuario ingresa la anterioridad de cuando desea recibir la	--			

	notificación				
E01.1	El usuario selecciona guardar	El sistema cierra el *popup* y guarda la notificación	El sistema guarda la notificación pero no cierra el *popup*	No	Leve
E01.2	El usuario selecciona cancelar	El sistema cierra el *popup* y no guarda la notificación	OK	Si	
E02	El usuario selecciona un mensaje para modificar				
	El usuario deja el mensaje en blanco				
E02.1	El usuario selecciona guardar	El sistema guarda la notificación sin mensaje	OK	Si	
E02.2	El usuario selecciona cancelar	El sistema no guarda la notificación	OK	Si	
E03	El usuario ingresa a la gestión de notificación				
	El usuario selecciona "Ver" de una notificación	El sistema despliega un *popup* con la información de la notificación	OK	Si	
	El usuario selecciona la opción "No notificar" de la lista de anterioridades				

	E03.1	El usuario selecciona la opción "Guardar"	El sistema guarda los cambios	OK	Si	
	E03.2	El usuario cierra el *popup*	El sistema no guarda los cambios	OK	Si	
Notificar hitos del cronograma	E01	El usuario ingresa al sistema	El sistema muestra las notificaciones correspondientes a las administradas	OK	Si	

Sprint 9

RF	Nombre	Entradas	Salida esperada	Salida	Aprobado	Cat.
Ver lista de proyectos	E01	El usuario ingresa a la sección de proyectos	El sistema despliega la lista de proyectos	OK	Si	
	E01.1	El usuario ordena los proyectos por nombre	El sistema ordena los proyectos por nombre	No los ordena	No	Leve
	E01.2	El usuario ordena los proyectos por fecha	El sistema ordena los proyectos por fecha	OK	Si	
	E01.3	El usuario ingresa el nombre de un proyecto en la caja de texto de filtro	El sistema filtra la lista de proyectos mostrando solamente los que coinciden con el ingreso	OK	Si	
Ver un proyecto	E01	El usuario selecciona un proyecto de la lista	El sistema despliega la información del proyecto seleccionado	OK	Si	

Sprint 10

RF	Nombre	Entradas	Salida esperada	Salida	Aprobado	Cat.
Insertar una nueva tarea a un cronograma en ejecución	E01	El usuario ingresa a un proyecto activo	El sistema despliega la información del proyecto	OK	Si	
		El usuario inserta una nueva tarea dentro del proyecto, sin asignarle tarea predecesora	El sistema ingresa la tarea a la lista	OK	Si	
		El usuario inserta una nueva tarea dentro del proyecto, asignándole una tarea predecesora	El sistema ingresa la tarea a la lista	OK	Si	
			El sistema recalcula las tareas sucesoras	OK	Si	
	E02	El usuario modifica una tarea existente cambiando su nombre	El sistema modifica la tarea	OK	Si	
	E03	El usuario modifica una tarea existente cambiando su fecha	El sistema modifica la tarea	OK	Si	
			El sistema recalcula las tareas sucesoras	No recalcula	No	Media
	E04	El usuario ingresa un insumo a una tarea existente	El sistema actualiza la tarea y el presupuesto del cronograma	OK	Si	
	E05	El usuario cambia la fecha de un estado fenológico	El sistema actualiza el estado	OK	Si	
			El sistema actualiza los siguientes	OK	Si	

		estados			
		El sistema actualiza las tareas dependientes de este estado	No actualiza	No	Media

Sprint 11

RF	Nombre	Entradas	Salida esperada	Salida	Aprobado	Cat.
Reajustar plan	E01	El usuario ingresa a un proyecto activo	El sistema reajusta el plan tomando en cuenta los pronósticos climáticos	OK	Si	
			El sistema reajusta el plan tomando en cuenta proyectos de otros años	OK	Si	
			El sistema reajusta el plan tomando en cuenta cambios de precios en *commodities*	No toma en cuenta estos cambios	No	Media

Sprint 12

RF	Nombre	Entradas	Salida esperada	Salida	Aprobado	Cat.
Informe gastos	E01	El usuario ingresa a la gestión de informes		OK	Si	
	E02	El usuario selecciona un proyecto terminado	El sistema despliega la lista de posibles informes	OK	Si	
	E03	El usuario selecciona el informe de presupuesto	El sistema despliega la diferencia entre el presupuesto planificado y el real	OK	Si	
	E04	El usuario selecciona el informe de tiempo	El sistema despliega la diferencia entre las fechas iniciales/finales estimadas y las fechas	OK	Si	

			reales			
Ver noticias	E01	El usuario ingresa al sistema	El sistema despliega abajo a la derecha las principales noticias del día de hoy	OK	Si	
	E02	El usuario ya ingresado, vuelve a la página principal	El sistema recarga las noticias para actualizarlas	No recarga	No	Media
	E03	El usuario recarga la pagina	El sistema recarga las noticias para actualizarlas	OK	Si	
	E04	El usuario selecciona una noticia	El sistema abre una nueva pestaña con el contenido de la noticia	OK	Si	
	E05	El usuario ingresa al sistema	El sistema despliega el clima del día de hoy	OK	Si	

11.2. ANÁLISIS DE HERRAMIENTAS EXISTENTES

A continuación se detallará una breve descripción de los competidores y productos sustitutos de SIGAMAS.

Competidores

OSIRIS Agro [1]

Procedencia: Argentina

Descripción: Software cuyo principal objetivo es brindarle al pequeño y mediano productor agrícola una herramienta de fácil manejo que le permita organizar y controlar su actividad productiva para hacerla más rentable.

Conclusiones: Es un producto más completo que SIGAMAS en cuanto a sus funcionalidades pero al ser de procedencia extranjera no posee los datos climáticos de nuestro país ni información de los suelos

Physis [2]

Procedencia: Argentina

Descripción: Es la solución para administraciones agropecuarias que permite automatizar sus circuitos de información. El diseño del Software de gestión agropecuaria Physis facilita la integración de los procesos de compra, producción, venta, fondos y contabilidad e impuestos.

Conclusiones: Se focaliza más en los procesos de logística, es de procedencia extranjera y no es utilizado en Uruguay.

Ñandú [3]

Procedencia: Argentina

Descripción: Es un sistema integral de operación, control y gestión para la administración de Empresas Agropecuarias, que vincula todas las actividades del campo con lo administrativo. Seguimiento exhaustivo de los movimientos de actividades de las siembras. Basado en los asientos de datos en partes diarios de todas las labores e insumos utilizados en cada potrero. Obteniendo reportes permanentes de avances y costos de los cultivos.

Conclusiones: Producto de procedencia Argentina que no posee datos de Uruguay, gran parte del sistema se focaliza en la Ganadería.

ViaRural [4]

Procedencia: Uruguay

Descripción: Es un portal agroindustrial con un listado ordenado de insumos y maquinarias para las actividades productivas en Uruguay como la agricultura.

Conclusiones: No posee las funcionalidades de SIGAMAS, sin embargo es muy utilizado para la obtención de maquinaria. Se puede agregar funcionalidad similar para el futuro.

Productos sustitutos

Erosión 6.0 [5]

Procedencia: Uruguay

Descripción: Software gratuito para la estimación de pérdidas de suelo por erosión.

Conclusiones: Herramienta muy precisa para la cálculo de la erosión. Actualmente es muy utilizado en Uruguay.

OKARA [6]

Procedencia: Uruguay

Descripción: Software que brinda ayuda para la gestión y planificación de la producción agrícola. Propone soluciones a partir del estudio de datos recolectados.

Conclusiones: Producto utilizado en Uruguay, posee varias de las funcionalidades de SIGAMAS y agrega otras como desarrollo inmobiliario (inversión estratégica en activos rurales).

Solapa4 [7]

Procedencia: Argentina

Descripción: Sistema para mejorar los procesos de toma de decisiones para incrementar la productividad. Se puede acceder en múltiples plataformas como *Tablet*, Celular o PC. Posee toda la información de los suelos de Argentina para una óptima utilización de los recursos.

Conclusiones: Es el Software más completo que se encontró. No tiene información de los suelos de Uruguay. SIGAMAS en un futuro tendrá las funcionalidades que posee Solapa4.

REFERENCIAS BIBLIOGRÁFICAS

[1] Herramienta Osiris. [En línea]. Disponible: http://www.osirisagro.com.ar/

[2] Herramienta Physis. [En línea]. Disponible: http://www.physis-agro.com/

[3] Herramienta Ñandu. [En línea]. Disponible: http://agronandu.com/

[4] Herramienta Via Rural. [En línea]. Disponible: http://www.viarural.com.uy/

[5] Herramienta Erosión 6.0. [En línea]. Disponible: http://www.cebra.com.uy/renare/planes-de-uso-y-manejo-de-suelos/herramientas-de-apoyo/

[6] Herramienta Okara. [En línea]. Disponible: http://okara.com.uy/

[7] Herramienta Solapa4. [En línea]. Disponible: http://www.solapa4.com/

11.3. LEY N° 18.564

Publicada D.O. 7 oct/2009 - Nº 27831

Ley Nº 18.564

CONSERVACIÓN, USO Y MANEJO ADECUADO DE LOS SUELOS Y LAS AGUAS

NORMAS

El Senado y la Cámara de Representantes de la República Oriental del Uruguay, reunidos en Asamblea General,

DECRETAN:

Artículo 1º.- Sustitúyese el artículo 2º del Decreto-Ley Nº 15.239, de 23 de diciembre de 1981, por el siguiente:

"ARTÍCULO 2º.- Todas las personas tienen la obligación de colaborar con el Estado en la conservación, el uso y el manejo adecuado de los suelos y de las aguas.

Los titulares de explotaciones agropecuarias, cualquiera sea su vinculación jurídica de los mismos con el inmueble que les sirve de asiento, o tenedores de tierras a cualquier título, quedan obligados a aplicar las técnicas que señale el Ministerio de Ganadería, Agricultura y Pesca para evitar la erosión y la degradación del suelo o lograr su recuperación y asegurar la conservación de las aguas pluviales.

De constatarse el incumplimiento en la aplicación de las técnicas aludidas en el inciso anterior, erosión o degradación del suelo, esa Secretaría de Estado, a través de la División Servicios Jurídicos, aplicará las sanciones establecidas en la normativa vigente y en todos los casos será solidariamente responsable el propietario del predio".

Artículo 2º.- La División Servicios Jurídicos del Ministerio de Ganadería, Agricultura y Pesca, en el ejercicio de sus potestades sancionatorias desconcentradas, cuando se trate de

incumplimiento a las normas que regulan el uso y el manejo de los suelos y de las aguas, podrá aplicar las siguientes sanciones:

A) Multa que será fijada entre 10 UR (diez unidades reajustables) y 10.000 UR (diez mil unidades reajustables).

   En caso de que la misma sea aplicada contra un propietario de inmuebles que no lo estuviere explotando en forma directa, a los efectos de la graduación de la multa se tendrá en cuenta la conducta de éste en relación al control que hubiere efectuado en cuanto al manejo de los suelos y de las aguas.

B) Suspensión por hasta un año de habilitaciones, permisos o autorizaciones para la actividad respectiva.

Artículo 3º.- En los contratos que se otorguen a partir de la vigencia de la presente ley por los cuales una de las partes se obliga a conceder a otra el uso y el goce de un predio rural con destino a cualquier explotación agropecuaria, las partes podrán establecer una cláusula en la cual se estipule que se depositará una suma de dinero en el Banco de la República Oriental del Uruguay, la cual servirá de garantía a efectos de cubrir una eventual multa por el mal manejo del uso de suelos y aguas a que se alude en el literal A) del artículo 2º de la presente ley.

El Poder Ejecutivo reglamentará las disposiciones del presente artículo.

Artículo 4º.- Quedan derogadas todas las normas que tácita o expresamente contravengan la presente ley.

Sala de Sesiones de la Cámara de Representantes, en Montevideo, a 27 de agosto de 2009.

ROQUE ARREGUI,

Presidente.

José Pedro Montero,

Secretario.

MINISTERIO DEL INTERIOR

MINISTERIO DE RELACIONES EXTERIORES

MINISTERIO DE ECONOMÍA Y FINANZAS

MINISTERIO DE DEFENSA NACIONAL

MINISTERIO DE EDUCACIÓN Y CULTURA

MINISTERIO DE TRANSPORTE Y OBRAS PÚBLICAS

MINISTERIO DE INDUSTRIA, ENERGÍA Y MINERÍA

MINISTERIO DE TRABAJO Y SEGURIDAD SOCIAL

MINISTERIO DE SALUD PÚBLICA

MINISTERIO DE GANADERÍA, AGRICULTURA Y PESCA

MINISTERIO DE TURISMO Y DEPORTE

MINISTERIO DE VIVIENDA, ORDENAMIENTO TERRITORIAL Y MEDIO AMBIENTE

MINISTERIO DE DESARROLLO SOCIAL

Montevideo, 11 de setiembre de 2009.

Cúmplase, acúsese recibo, comuníquese, publíquese e insértese en el Registro Nacional de Leyes y Decretos, la Ley por la que se dictan normas relacionadas con la conservación, el uso y el manejo adecuado de los suelos y de las aguas.

TABARÉ VÁZQUEZ.

RICARDO BERNAL.

PEDRO VAZ.

ANDRÉS MASOLLER.

GONZALO FERNÁNDEZ.

MARÍA SIMON.

VÍCTOR ROSSI.

RAÚL SENDIC.

JULIO BARÁIBAR.

MARÍA JULIA MUÑOZ.

ERNESTO AGAZZI.

HÉCTOR LESCANO.

JACK COURIEL.

MARINA ARISMENDI.

11.4. ADQUISICIÓN DEL CONOCIMIENTO

Reuniones con expertos del dominio

A lo largo de todo el proyecto se realizaron reuniones con diferentes expertos del dominio que guiaron tanto en aspectos del negocio como en el conocimiento específico de agricultura.

Como ya fue nombrado anteriormente, el principal experto del dominio es el Ing. Tabaré Alcorta, él fue el primero en comentar la dificultad que sufre la agricultura al no existir la herramienta necesaria para gestionar la misma sin estar vinculada al modelo fenológico de los cultivos.

El equipo se reunió con Tabaré todo el año, él fue quien le enseñó todo lo relacionado con la gestión de los cultivos y aportó material y guías para extraer más conocimiento útil.

Además, en las reuniones participaba Santiago Alcorta, estudiante avanzado de la carrera de Ingeniero Agrónomo, él fue, principalmente quien orientó al equipo con todo lo vinculado a los modelos fenológicos, es decir, a los ciclos de vida de los cultivos. Al igual que Tabaré, proporcionó material e información para obtener un mayor conocimiento sobre el tema.

En un comienzo, cuando el equipo estaba más motivado con el módulo que controlaría la erosión del suelo en las rotaciones, se reunió repetida veces con el Biólogo Pablo Montes, funcionario de MVOTMA (Ministerio de Vivienda, Ordenamiento Territorial y Medio Ambiente), quien brindó el enfoque relacionado con la erosión de suelo que generan los cultivos al Uruguay y la concientización de la importancia de prestar especial cuidado a salvar este recurso. También fue muy importante el nexo que realizó para que el equipo pudiera presentarse en el Congreso de Suelos del Uruguay a principios de agosto, donde se obtuvo el mayor *Feedback* de todo el proyecto.

Por otro lado, Pablo también puso en contacto al equipo con el Ingeniero Mario Pérez, catedrático de la cátedra de suelos de la Facultad de Agronomía de la UDELAR (Universidad DE LA República), quien explicó al equipo la fórmula para calcular la erosión de los suelos a través de la rotación de los cultivos y compartió material sobre este tema con el equipo.

Como ya se mencionó anteriormente, el equipo participó del Congreso de Suelos del Uruguay, donde tuvo la oportunidad de exponer el proyecto en el cual se estaba trabajando, pudiendo conocer y hablar con varios Ingenieros Agrónomos y Productores Agrícolas, a quienes además se les pidió la realización de las encuestas de la página web del proyecto.

Este Congreso sirvió para conocer y relacionarse con Mónica Barbazán, la encargada de la realización del Congreso y directora del SUCS (Sociedad Uruguaya de Ciencia del Suelo), quien invitó a presentar el proyecto frente a la cámara de oleaginosos del Uruguay donde se obtuvo un buen *Feedback* y se pudo conocer con mayor precisión la realidad de los Productores Agrícolas.

También, en la segunda mitad del año, se obtuvo la ayuda del Ingeniero Mario Pereira, funcionario del MGAP (Ministerio de Ganadería, Agricultura y Pesca) responsable de RENARE (Dirección de Recursos Nacionales Renovables) en Tacuarembó, quien aportó conocimiento, información y material para realizaron los modelos fenológicos del trigo.

Aplicación de la investigación en SIGAMAS

El siguiente proceso indica cómo sería una rutina normal usando SIGAMAS, es decir cómo funciona el sistema desde que se crea un proyecto hasta que el mismo se finaliza indicando la fecha de cosecha.

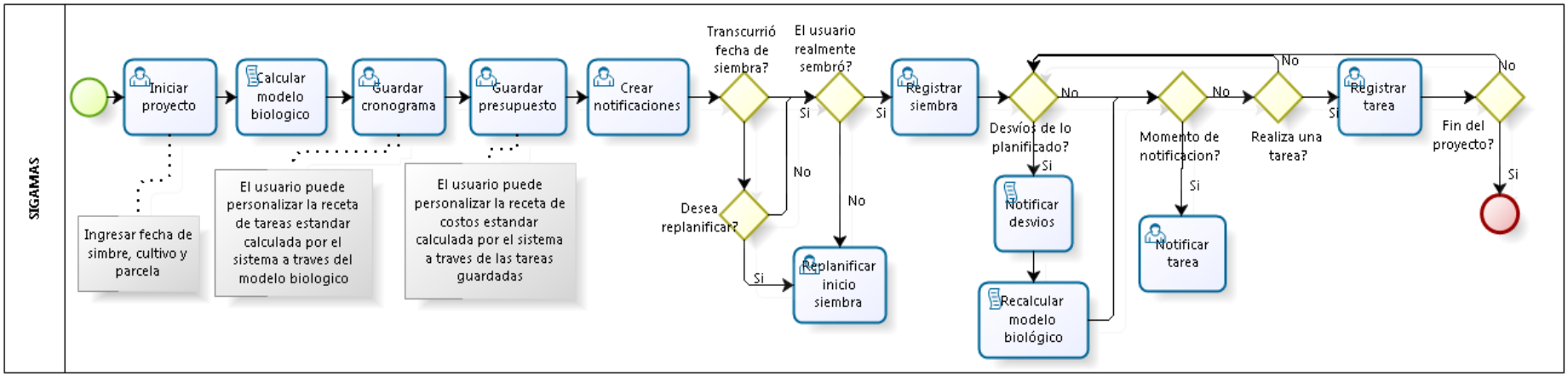


Ilustración 11-1 Proceso de Ejecución de SIGAMAS

11.5. CANVAS

Segmentos del mercado (*Costumer Segments*)

El mercado es segmentado dado que el modelo de negocio que se cree más conveniente estará definido por varios segmentos con necesidades y problemas diferentes. Pero relacionadas entre sí, ya que ambos se encuentran en el sector agrícola de Uruguay.

- Empresas que ofrecen agro insumos (venta de materia prima -semillas y agroquímicos): estas empresas podrán brindar el servicio a sus clientes con una configuración adaptable a cada tipo de cultivar (según la compra realizada por su cliente) y en base a los requerimientos de este cultivar se ofrecen productos (herbicidas, fungicidas, fertilizantes) que la misma empresa tiene a la venta. Es decir, estas empresas tendrán disponible para ofrecer a sus clientes una configuración del sistema de acuerdo a la compra realizada por estos.

- Empresas productoras agrícolas: los productores que anteriormente accedieron al Software por medio de la empresa "semilleras" podrán acceder a paquetes pagos con mayores funcionalidades y membrecías (soporte, asistencia, etc.). Estas deben ser grandes y medianas empresas y tener predisposición a adoptar la tecnología como herramienta de trabajo que les otorgará beneficios extras.

Se definieron a las empresas que venden agro insumos, como empresas "semilleras", dado que la configuración del sistema depende de las semillas del cultivo que estás le vendan al productor.

Propuesta de valor (*Value Proposition*)

A los diferentes segmentos del mercado se les ofrece el Software como un servicio (*SaaS*). Los servicios esenciales que el Software ofrecerá a cada uno de ellos son:

- Empresas "semilleras": Tienen el problema de vender un producto estandarizado sin poder lograr diferenciación en su mercado. Ofreciendo el valor extra que les proporcionará SIGAMAS a sus clientes, podrán encontrar el modo de diferenciarse del resto y lograr fidelización del cliente a sus productos. En resumen, se le ofrece valor diferencial en relación a su competencia. No sólo venderán las semillas sino que le venderán la herramienta para gestionar el cultivo de las mismas.

- Empresas productoras agrícolas: (mejora el rendimiento, personalización, reducción de riesgos, comodidad/utilidad) Ofrece a sus clientes la posibilidad de centralizar, en pocos pasos, la información en un solo lugar, además permite mejorar el rendimiento de su producción y controlar el uso del suelo en conjunción con las leyes vigentes. (También permite que el cliente personalice fácilmente sus planes según su experiencia y las recomendaciones del sistema - Flexible)

Se ofrece un servicio post venta que cuenta con: soporte técnico, manuales de uso, guías de uso, contacto telefónico y electrónico; tanto para productores como para empresas que venden insumos agrarios.

A través del sitio (http://SIGAMAS.com/noticias.html), se realizaron 2 encuestas diferentes para cada uno de los segmentos anteriormente nombrados dónde además de validar el producto se busca conocer la opinión de futuros clientes sobre el valor que ellos perciben que SIGAMAS les podría ofrecer. (Ver Anexo de "Encuestas y Entrevistas")

Canales (*Channels*)

Aquí se definió como se hará para que el producto le llegue a los clientes.

- Empresas de agro-insumos: con la presencia del equipo en congresos del área agraria y realizando llamadas telefónicas directamente a empresas del rubro para luego establecer reuniones. Con ellas se tendrá un trato personal y directo, ya que no solo se espera que estas vendan el producto como uno suyo, sino que se desea aprender de ellos a cómo vender mejor y mejorar el producto con su input y cualquier interrogante que surja al ir poniendo en práctica el proyecto.

- Empresas de producción agrícola: Al ser estos los clientes finales, la forma que se ha decidido adoptar es comunicar el producto a través de las empresas que han sido encontradas en el paso anterior (empresas que ofrecen agro-insumos), es decir, estas además de ser clientes de SIGAMAS también funcionarán como canales. Luego de haberles transmitido correctamente las ganancias y valores que aporta el Software para los productores agrícolas, las empresas "semilleras" harán lo mismo con sus clientes para que estos usen el Software. También se cuenta con la web, la cual al principio no se tiene asumido que tenga publicidad para potenciar la llegada de clientes sino que su enfoque sería que la comunicación sea a través de las empresas de agro-insumos. Una vez que el cliente ha accedido al Software, se entrará en contacto con él mediante llamadas electrónicas (Skype), email y teléfono.

Relación con los clientes (*Customer Relationships*)

La relación con clientes va a ser de manera personal y siempre apoyándose en ellos para comunicar el producto y a la vez aprender de ellos y evolucionar juntos. Este tipo de relación va a ayudar en los 3 ítems definidos como "Relación con nuestros clientes":

- Captar nuevos clientes: Este tipo de relación es la que más cuesta en el mercado, y por ello el foco se basa en comunicar el producto, con sus características, funcionalidades y beneficios que otorga a sus usuarios. A las empresas "semilleras" se le deben comunicar todos los aspectos claves, para que luego adopten el producto y estén convencidas de los beneficios que le brinda a los consumidores finales y a ellos mismos al ofrecerlo como un extra en sus ventas.

- Fidelización de los mismos: Fundamental que luego de haber conseguido a los clientes, es mantenerlos felices y que al utilizar el sistema, quieran seguir usándolo. Se presta atención al servicio de post-venta, el cual ofrece asistencia on-line cuando el cliente la solicite, guías y manuales de uso.

- Estimulación de compra de los clientes: Luego de haber utilizado el sistema en una zafra y observar los resultados que ha aportado el mismo, el usuario puede convencerse de la potencialidad que el sistema tiene y los beneficios que le brinda. Si

desea puede obtener un *Plugin* que se adapte mejor a sus necesidades, o quizás aumentar las funcionalidades para su tipo de cosecha. En el mismo sistema se le indicará las limitaciones que tiene la versión que está usando y las posibilidades de obtener un producto aumentado haciéndose de nuevos *Plugin*.

Fuente de Ingresos (*Revenue Streams*)

La estrategia que se aplicará para generar efectivo de parte de los clientes es "Subscripciones" dado que es un SaaS, los clientes podrán pagar por usar el "servicio".

- Empresas "semilleras": Las mismos tendrán una cuenta especial con permisos para crear 20 cuentas de usuarios a sus clientes en el sistema. Una vez superado este número, la empresa deberá abonar por cada cuenta de usuario que quiera crear a un nuevo cliente o comprar paquetes con una capacidad mayor para crear más cuentas de usuario.

- Productores agrícolas: Los mismos deberán pagar si desean acceder a un producto más completo o a herramientas específicas de un cultivo en particular que no ha sido brindada en la versión del sistema que la empresa "semillera" tendrá para ofrecerles. La compra la van a poder realizar en la misma web, o personalmente si así lo desean.

Recursos Claves (*Key Resources*)

Aquí se realizó una división en dos áreas principales y que definen la estructura del proyecto.

Por un lado se tienen los recursos para que el proyecto funcione desde su base, sea creado de manera correcta y luego mantenido para poder crecer con la demanda:

- Director de equipo.

- Programadores y Testers, al principio, al ser un equipo joven que trabaja con metodologías ágiles, no se tiene tan definidos y separados los roles. Luego que el sistema evolucione y el equipo con él, los roles también lo harán.

- Servidores, los cuales también al principio su demanda no va a ser grande, pero si se necesita que sean confiables y se encuentren disponibles las 24 horas del día, se desea una pronta recuperación en caso de existir un problema con los mismos.

Por otro lado, para que el proyecto sea exitoso se necesita:

- Encargado de comunicación con empresas vendedoras.

- Área de servicio técnico.

Socios Claves (*Key Partners*)

- Aquí se tienen a las mismas empresas "semilleras", que van a ofrecer el producto como uno suyo y a la vez trabajar juntos para ir mejorando el negocio y producto. Aprendiendo cómo llegarles a los clientes finales y mejorando la atención hacia ellos.

Estas empresas van a aportar conocimiento en el área, tanto en lo técnico hasta en cómo se debe vender el producto y ayudar a entender mejor a los usuarios finales.

- Se desea, más adelante, hacer asociaciones con Ingenieros Agrónomos y Veterinarios que trabajan con diferentes Productores Agrícolas, ofreciéndoles comisiones por ventas de cuentas de usuarios que realicen.

Actividades Claves (*Key Activities*)

Aquí también se pudo dividir en dos grandes áreas las actividades claves.

Inherentes a la programación y gestión del proyecto:

- Aquí se encuentran varias actividades que definen si el proyecto es exitoso o no. Algunas son implícitas por ser un desarrollo de Software profesional y ético. Consisten en que el Software haya sido realizado siguiendo los principios de la programación, para brindarle al cliente la mejor experiencia y a su vez hacer que el Software sea actualizable y se pueda adaptar a futuros cambios.

- Diseño de los procesos del ciclo de vida del Software. (análisis, diseños, desarrollo, mantenimiento y prueba).

Inherentes al negocio:

- Comunicación eficiente y clara con los vendedores de semillas, para que ellos puedan transmitir el producto como suyo y así explicar y venderlo de la mejor manera.

- Atención post-venta: Actividad de servicio al cliente que ya adquirió el producto y precisa conocimiento sobre cómo usar el sistema, y a su vez personalizar el mismo si en necesario. Se debe cuidar al cliente que obtiene el producto, ya que si éste es cuidado en un futuro va a adquirir una licencia superior y esto va a generar ingresos a la empresa.

Estructura de costos (*Costs Structure*)

- Los costos más importantes son aquellos costos directos relacionados con la creación de Software (análisis, diseños, desarrollo y prueba).

- También se considera importante los costos variables que se desprenden de actividades ya mencionadas anteriormente, tales como costos de venta, costos al mantener una relación personal con el cliente y costos para conseguir la fidelización del mismo.

En un principio se pensó en llegar al cliente final de manera masiva, lo cual fue descartado por el alto costo que esto requiere, se decidió tomar el camino de presentar el proyecto en congresos relacionados al área agrícola, presentaciones de productos también relacionados al área y reuniones personales con empresas del rubro. Estos costos no son tan elevados y para la empresa y brindan resultados rápidamente, ya que se han obtenido llamados de interesados al presentar SIGAMAS en eventos del rubro.

11.6. COMPARACIÓN DE LAS METODOLOGÍAS

El objetivo de este anexo es explicar cómo se realizó el análisis entre las distintas metodologías, detallando por qué se eligió comparar Scrum, FDD, *Extreme Programming*, RUP, MSDM y MSF y bajo qué criterios se compararon.

1. **Elección de las metodologías.**

Para elegir cuáles iban a ser las metodologías que iban a ser analizadas, el equipo se reunió y realizó una tormenta de ideas de la siguiente forma:

Cada uno tenía que escribir 5 metodologías en un papel sin que los demás lo vieran. Luego se juntaron los papeles y se hizo una lista de todas las metodologías ingresadas, ordenándolas de mayor a menor repetición. La lista resultó ser la siguiente:

1. Scrum (4 repeticiones)
2. RUP (4 repeticiones)
3. *Extreme Programming* (4 repeticiones)
4. FDD (3 repeticiones)
5. MSF (2 repeticiones)
6. MSDM (2 repeticiones)
7. ASD (1 repetición)

Se descartó la última metodología ya que solo un integrante la había elegido. Este descarto fue acordado por todos los miembros del equipo.

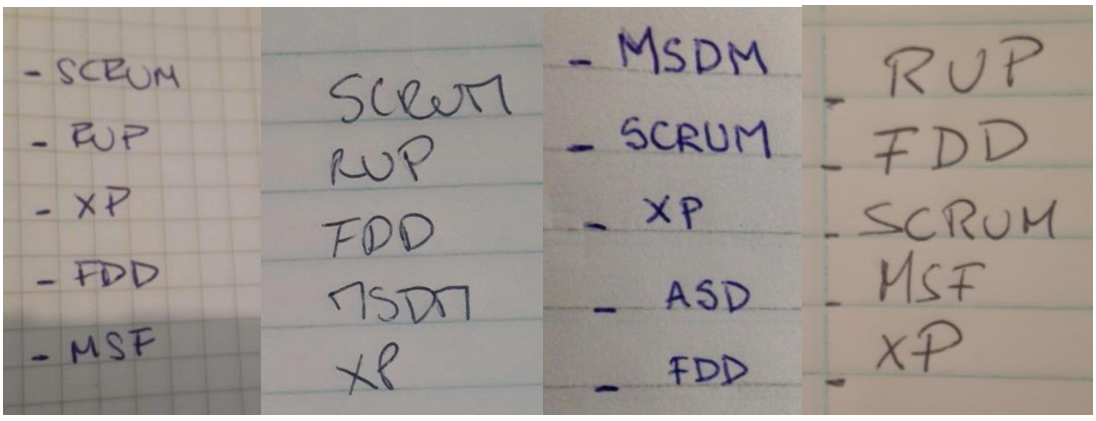


Ilustración 11-2 Lista de metodologías de cada integrante

Elección de los puntos a cumplir

La elección de los puntos se realizó de la misma forma. Cada uno eligió 5 puntos que el proyecto necesitaba cumplir y luego se listaron de mayor a menor cantidad de repetición. La lista es la siguiente:

1. Punto A - El equipo necesita estar preparado para cambios de requerimientos durante el proyecto (4 repeticiones)
2. Punto B - No existe un contrato rígido por parte de un cliente (4 repeticiones)
3. Punto C - El equipo es un grupo pequeño de menos de 10 integrantes (3 repeticiones)

4. Punto D - El ciclo de vida debe ser iterativo y con iteraciones cortas (3 repeticiones)
5. Punto E - La metodología debe priorizar la comunicación directa y la colaboración (2 repeticiones)
6. Punto F - Se debe contar con ceremonias de planificación, revisión y retrospectiva (2 repeticiones)
7. Punto G - Se debe contar con el rol de *Project Manager* (1 repeticiones)
8. Punto H - La metodología debe ser ágil (1 repeticiones)

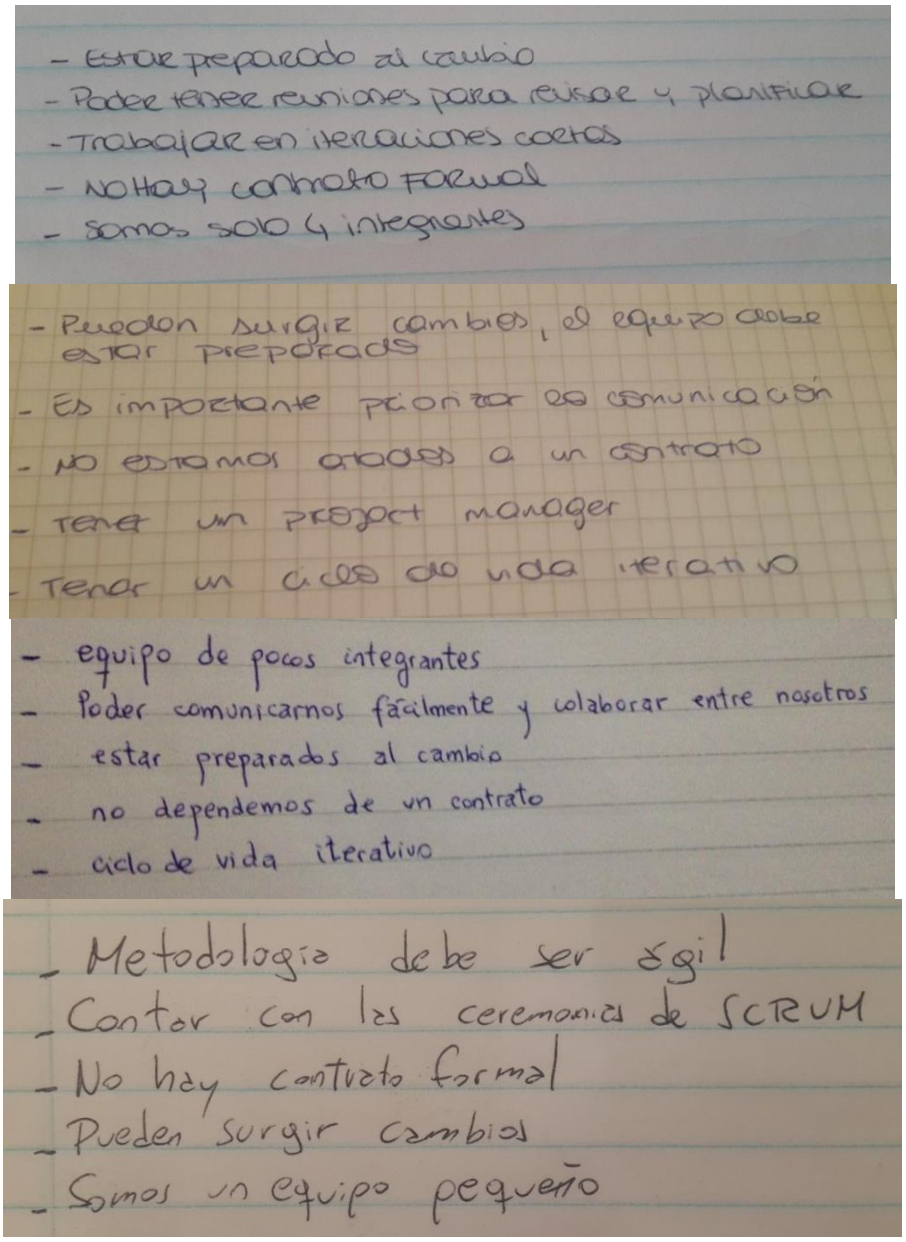


Ilustración 11-3 Lista de puntos a cumplir de cada integrante

Cómo última etapa, se le asignó a cada punto una importancia del 1 al 5, que fue elegida por el equipo mediante *Planning* Poker.

- Punto A: Importancia 5
- Punto B: Importancia 3

- Punto C: Importancia 5
- Punto D: Importancia 4
- Punto E: Importancia 4
- Punto F: Importancia 5
- Punto G: Importancia 4
- Punto H: Importancia 2

2. Resultados

Habiendo definido las metodologías a comparar, los puntos sobre los cuales se iban a comparar y sus respectivas importancias, se realizaron los cálculos para definir cuál era la metodología que mejor se adaptara a todas las necesidades del equipo y del proyecto.

Cada metodología se evaluó sobre 32 puntos, siendo ese número la suma de los Puntos a evaluar. 5 + 3 + 5 + 4 + 4 + 5 + 4 + 2 = 32

Si la metodología cumplía con el Punto evaluado, se le sumaba su importancia.

Scrum: 5 + 3 + 5 + 4 + 4 + 5 + 2 = 28

FDD: 5+ 3 + 4 + 4 + 4 + 2 = 22

Extreme Programming: 5 + 3 + 5 + 4 + 4 + 2 = 23

MSF: 5 + 3 + 4 + 4 =16

RUP: 5 + 4 + 4 + 4 = 17

DSDM: 5 + 2 + 4 + 4 + 4 + 2 = 21

3. Análisis realizado

A continuación se detalla la justificación del cumplimiento de cada punto.

Punta A - El equipo necesita estar preparado para cambios de requerimientos durante el proyecto

Scrum: Al ser una metodología ágil, está especialmente ideado para ejecutarse en proyectos con requerimientos cambiantes. **+5**

FDD: Al ser una metodología ágil, está especialmente ideado para ejecutarse en proyectos con requerimientos cambiantes. **+5**

Extreme Programming: Al ser una metodología ágil, está especialmente ideado para ejecutarse en proyectos con requerimientos cambiantes. **+5**

MSF: “Mantenerse ágil, esperar cambios y adaptarse a ellos.”[4] **+5**

RUP: Aunque RUP no sea una metodología ágil, su proceso está pensado para adaptarse al cambio. **+5**

DSDM: La metodología es ágil por lo que implica un proceso que se adapta al cambio. [6] **+5**

Punto B - No existe un contrato rígido por parte de un cliente

Scrum: Al ser una metodología ágil, puede adaptarse a proyectos con o sin contrato. **+3**
FDD: Al ser una metodología ágil, puede adaptarse a proyectos con o sin contrato. **+3**

Extreme Programming: Al ser una metodología ágil, puede adaptarse a proyectos con o sin contrato. **+3**

MSF: MSF es una metodología que puede adaptarse a muchos tipos de proyectos, dentro de los cuales se encuentran proyectos informales. **+3**

RUP: RUP no es una metodología ágil por lo que su aplicación se adapta mejor a proyectos con contratos existentes por parte de los clientes. **+0**

DSDM: No se encontró información que indique que deba existir un contrato pre establecido. **+2**

Punto C - El equipo es un grupo pequeño de menos de 10 integrantes

Scrum: Scrum está especialmente pensado para adaptarse a proyectos de entre 3 y 9 integrantes. [1] **+5**

FDD: FDD es una metodología ágil pero sin embargo está orientada a equipos más grandes que los de Scrum y *Extreme Programming*. **+0**

Extreme Programming: Al igual que Scrum, *Extreme Programming* está orientado a equipos pequeños. **+5**

MSF: No se encontró información que indique que los equipos deban ser pequeños. **+0**

RUP: No se encontró información que indique que los equipos deban ser pequeños. **+0**

DSDM: No se encontró información que indique que los equipos deban ser pequeños. **+0**

Punto D - El ciclo de vida debe ser iterativo y con iteraciones cortas

Scrum: Emplea un ciclo de vida iterativo [1] **+4**

FDD: Emplea un ciclo de vida iterativo [2] **+4**

Extreme Programming: Emplea un ciclo de vida iterativo [3]**+4**

MSF: "La práctica del desarrollo iterativo es un tema constante en MSF. Los documentos, diseños, planes y otras entregas se desarrollan de manera iterativa. Como cabe esperar, el modelo de gobierno de MSF tiene un enfoque iterativo." [4] **+4**

RUP: Emplea un ciclo de vida iterativo [5] **+4**

DSDM: Emplea un ciclo de vida iterativo [6] **+4**

Punto E - La metodología debe priorizar la comunicación directa y la colaboración

Scrum: Las ceremonias de Scrum más sus políticas de colaboración prioriza la comunicación directa y la colaboración [1]. **+4**

FDD: Al ser una metodología ágil, prioriza la comunicación **+4**

Extreme Programming: Las prácticas de *Extreme Programming* promueven el trabajo en equipo y por ende la colaboración y la comunicación. [3] **+4**

MSF: "Fomentar una comunicación abierta. Para que el equipo sea eficaz y eficiente, tanto usted como su equipo deben compartir niveles de información apropiados entre los miembros del equipo y en toda la empresa."[4] **+4**

RUP: Mismo si RUP no es considerado una metodología ágil, prioriza la comunicación entre los miembros del equipo. **+4**

DSDM: Al ser una metodología ágil, prioriza la comunicación **+4**

Punto F - Se debe contar con ceremonias de planificación, revisión y retrospectiva

Scrum: "Scrum define 4 eventos formales para la inspección y adaptación: *Sprint Planning*, *Daily* Scrum, *Sprint Review* y *Sprint Retrospective*. [1] **+5**

FDD: Estas ceremonias no forman parte de la metodología FDD. **+0**

Extreme Programming: Estas ceremonias no forman parte de la metodología *Extreme Programming*. **+0**

MSF: Estas ceremonias no forman parte de la metodología MSF. **+0**

RUP: Estas ceremonias no forman parte de la metodología RUP. **+0**

DSDM: Estas ceremonias no forman parte de la metodología DSDM. **+0**

Punto G - Se debe contar con el rol de Project Manager

Scrum: No se encontró un rol en Scrum que se adaptara exactamente a un *Project Manager*. **+0**

FDD: "*Project Manager* Es el líder administrativo y financiero del proyecto. Una de sus tareas principales es proteger al equipo de distracciones externas y permitir que el equipo de pueda trabajar en las condiciones apropiadas. En el FDD el *Project Manager* tiene la última palabra sobre temas referidos al alcance, tiempo y personal."[2] **+4**

Extreme Programming: No se encontró un rol que se asimilara a un *Project Manager*. **+0**

MSF: No se encontró un rol que se asimilara a un *Project Manager*. **+0**

RUP: Cuenta con un rol de *Project Manager*. [5] **+4**

DSDM: Cuenta con el rol de *Project Manager*. **+4**

Punto H - La metodología debe ser ágil

Scrum: Es una metodología ágil. **+2**

FDD: Es una metodología ágil: **+2**

Extreme Programming: Es una metodología ágil. **+2**

MSF: Puede aplicarse cómo una metodología ágil pero por definición no lo es: **+0**

RUP: No es una metodología ágil: **+0**

DSDM: Es una metodología ágil: **+2**

REFERENCIAS BIBLIOGRÁFICAS

[1] Scrum Guide, "The Scrum Guide™", 2013. [En línea]. Disponible: http://www.Scrumguides.org/docs/Scrumguide/v1/Scrum-Guide-US.pdf#zoom=100

[2] Universidad ORT Uruguay, "Metodología FDD", 2003. [En línea]. Disponible: http://fi.ort.edu.uy/innovaportal/file/2021/1/metodologiafdd.pdf

[3] Extreme Programming, a gentle Introduction. [En línea]. Disponible: http://www.extremeprogramming.org/

[4] M. Keeton, *Microsoft Solutions Framework (MSF): A Pocket Guide*, USA: Van Haren Publishing, 2006. [5] *What is Rational Unified Process*? [En línea]. Disponible: http://www.ibm.com/developerworks/rational/library/content/RationalEdge/jan01/WhatIstheRationalUnifiedProcessJan01.pdf

[5] Rational Unified Process, "What is Rational Unified Process?". [En línea]. Disponible: http://www.ibm.com/developerworks/rational/library/content/RationalEdge/jan01/WhatIstheRationalUnifiedProcessJan01.pdf

[6] DSDM, "What is DSDM?". [En línea]. Disponible: http://www.dsdm.org/content/what-dsdm

11.7. COMPARACIÓN DE HERRAMIENTAS DE GESTIÓN DE METODOLOGÍAS ÁGILES

Para poder gestionar las tareas del proyecto, fue importante encontrar una herramienta de gestión de metodología ágil. Al principio del proyecto, el equipo planificaba y registraba las horas mediante una planilla Excel. Esto resultaba ineficiente ya que se invertía demasiado tiempo en esta actividad.

Se decidió entonces investigar sobre las herramientas existentes y probarlas a ver cuál era la mejor para el equipo. La herramienta debía cumplir las siguientes características:

- A: Gestión por iteraciones
- B: Permitir la creación de tareas al estilo *User Story* o *Feature* con sub tareas asignadas
- C: Que las tareas puedan ser asignadas a múltiples usuarios
- D: Generar reportes
- E: Gratis

Estos requisitos fueron definidos mediante tormenta de ideas en una reunión y el equipo decidió mediante *Planning* Poker utilizar los 5 puntos anteriormente mencionados. Debían ser requisitos que se relacionaran con la definición de nuestro proceso y con el paradigma de metodología ágil.

A continuación se detalla la tabla de decisión para las metodologías.

Características → / **Herramienta** ↓	A	B	C	D	E
AgileFant[1]	✓	✓	✓	✓	✓
ScrumDesk[2]	✓	✓	✗	✓	✓
***Acunote*[3]**	✓	✗	✗	✗	✗
***AgileWrap*[4]**	✓	✓	✗	✓	✓
***Flying Donnut*[5]**	✓	✗	✗	✓	✗
Herogami[6]	✗	✗	✗	✓	✗
QuickScrum[7]	✓	✗	✗	✓	✓

Tabla 11-1 Comparación de metodologías

Según el análisis, AgileFant era el más apropiado a los criterios del equipo. Se decidió usarlo y el resultado fue muy positivo ya que la herramienta cumplió con todas las necesidades.

Además cuenta con una versión para Android (Beta) que permite visualizar las tareas asignadas de cada uno desde cualquier lugar.

REFERENCIAS BIBLIOGRÁFICAS

[1] AgileFant [En línea]. Disponible en: http://www.AgileFant.com

[2] ScrumDesk [En línea]. Disponible en: http://www.Scrumdesk.com

[3] Acunote [En línea]. Disponible en: http://www.acunote.com

[4] AgileWrap [En línea]. Disponible: http://www.agilewrap.com

[5] Flying Donnut [En línea]. Disponible en: https://www.flying-donut.com

[6] Herogami [En línea]. Disponible en: https://www.herogami.com

[7] QuickScrum [En línea]. Disponible en: http://www.quickScrum.com/

11.8. REGISTRO DE LAS MÉTRICAS

Sprints sin proceso definido (*Sprint* 1 a 2)

Burndowns

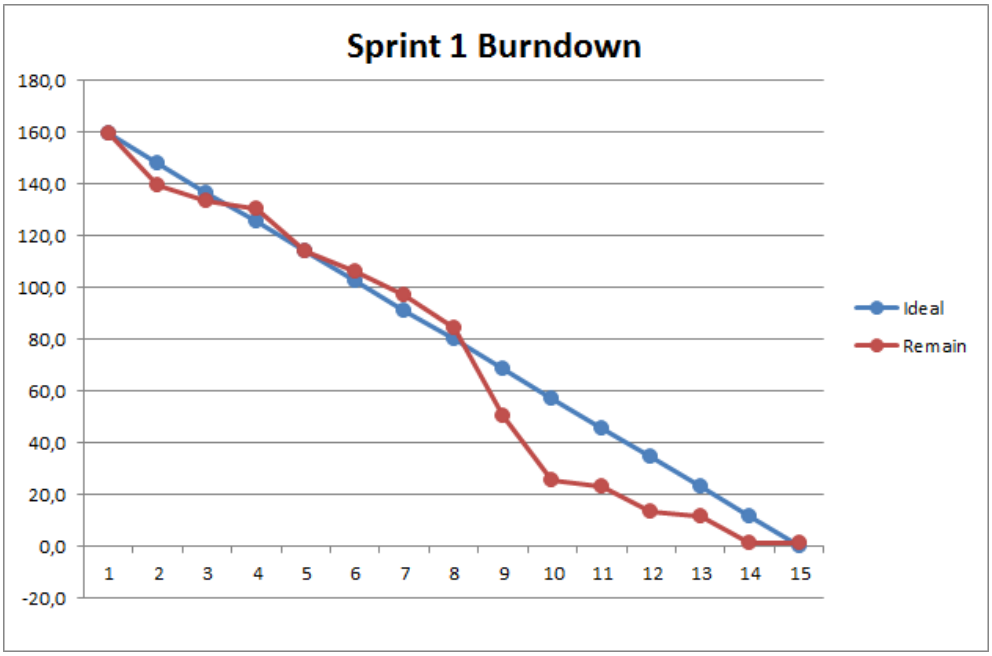


Ilustración 11-4 Sprint 1 Burndown

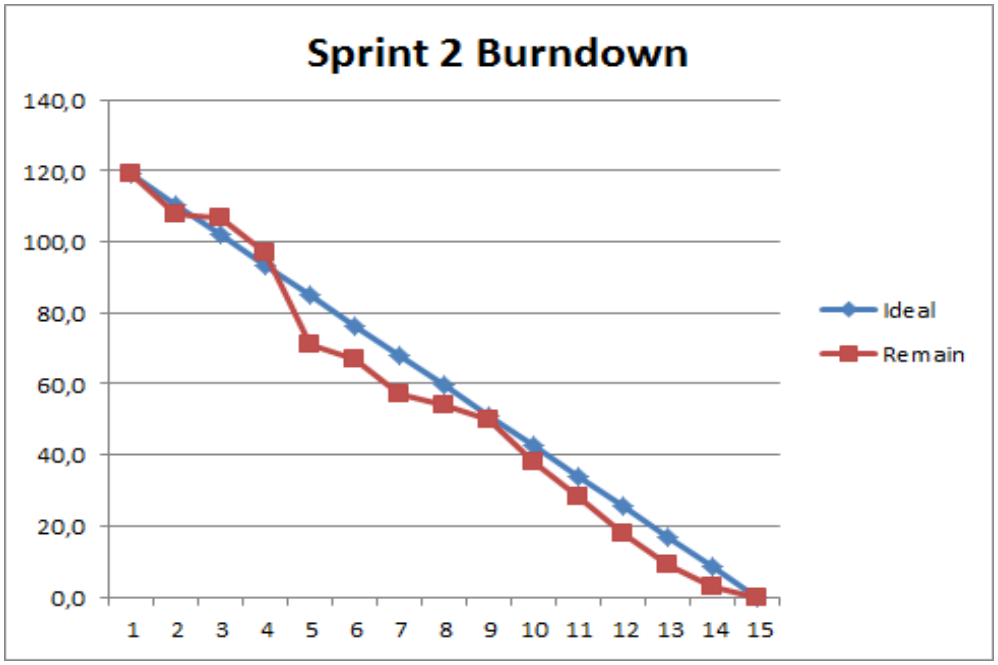


Ilustración 11-5 Sprint 2 Burndown

Gracias a los *Burndowns*, se pudo ir viendo cómo se estaban cumpliendo las tareas. Cómo se vio en la definición de *Burndowns*, si la línea roja se encuentra por debajo de la azul quiere

decir que el trabajo está adelantado. Se considera esto cómo algo positivo. Se consideran entonces que los *Sprints* son exitosos tomando en cuenta los *Burndowns*.

Velocidad

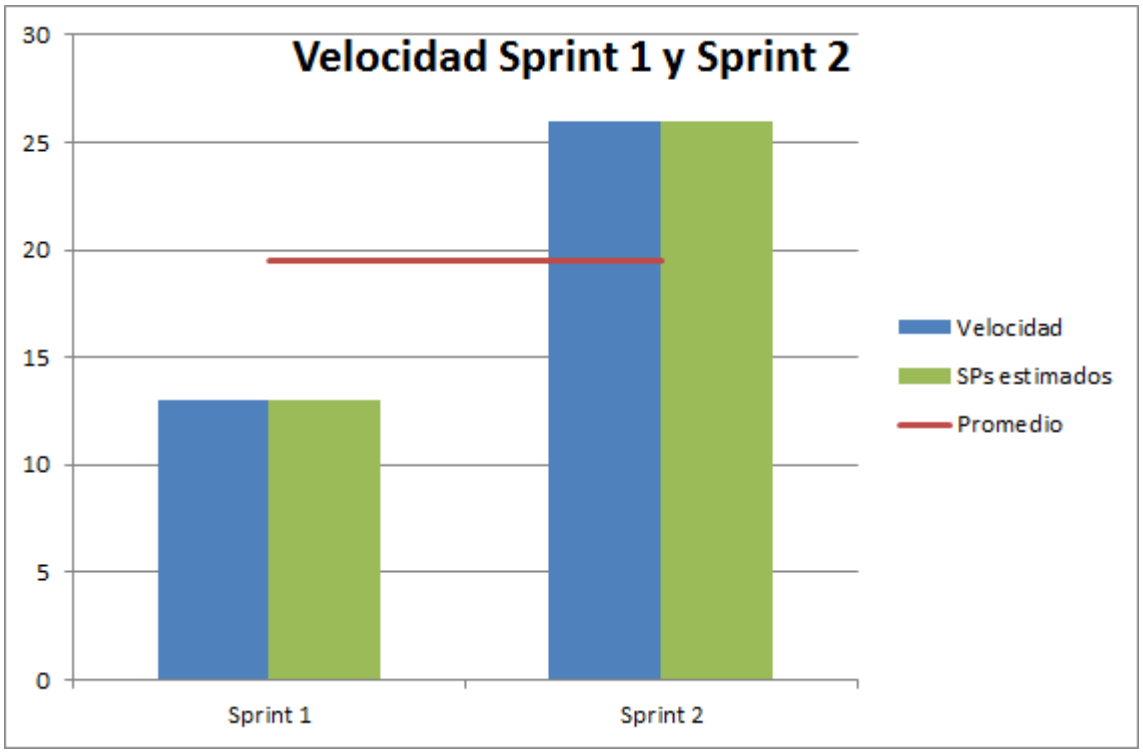


Ilustración 11-6 Velocidad Sprint 1 y Sprint 2

Cómo se puede ver, en los dos primeros *Sprints*, se cumplió al 100% con las *Story Points* estimadas. Otro punto favorable para considerar que los *Sprints* son exitosos.

Retrabajo

No hubo retrabajo en estos dos *Sprints* por lo que se concluye que los *Sprints* 1 y 2 fueron exitosos.

Sprints con Scrum puro (*Sprint* 3 y 4) (*Sprints* con el primer proceso definido)

Burndowns

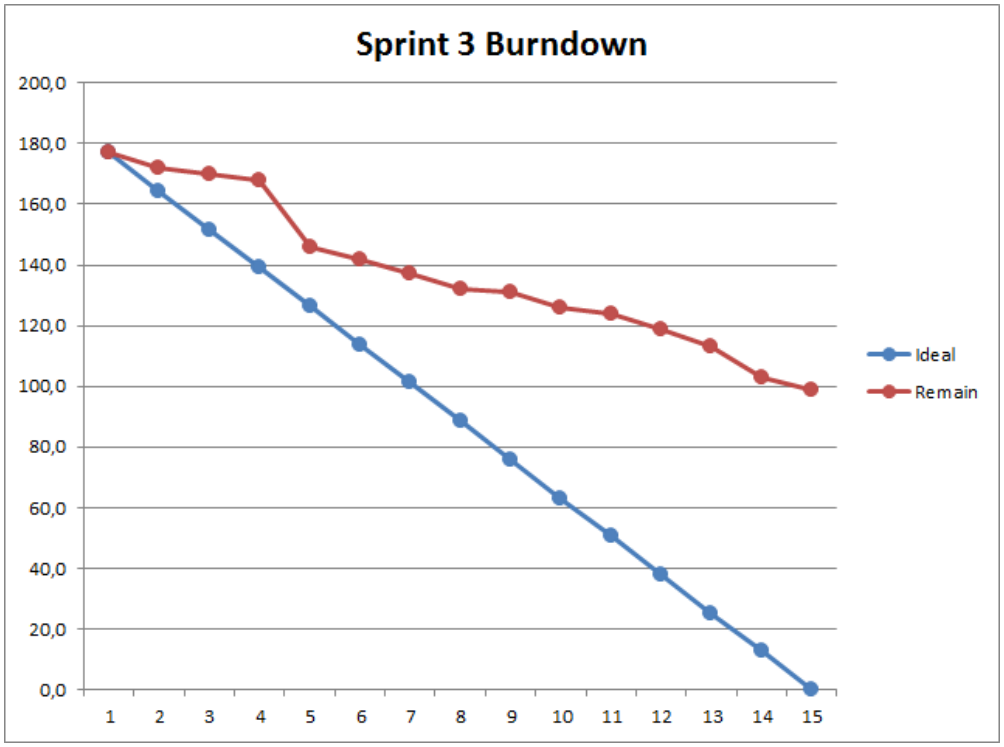


Ilustración 11-7 Sprint 4 Burdown

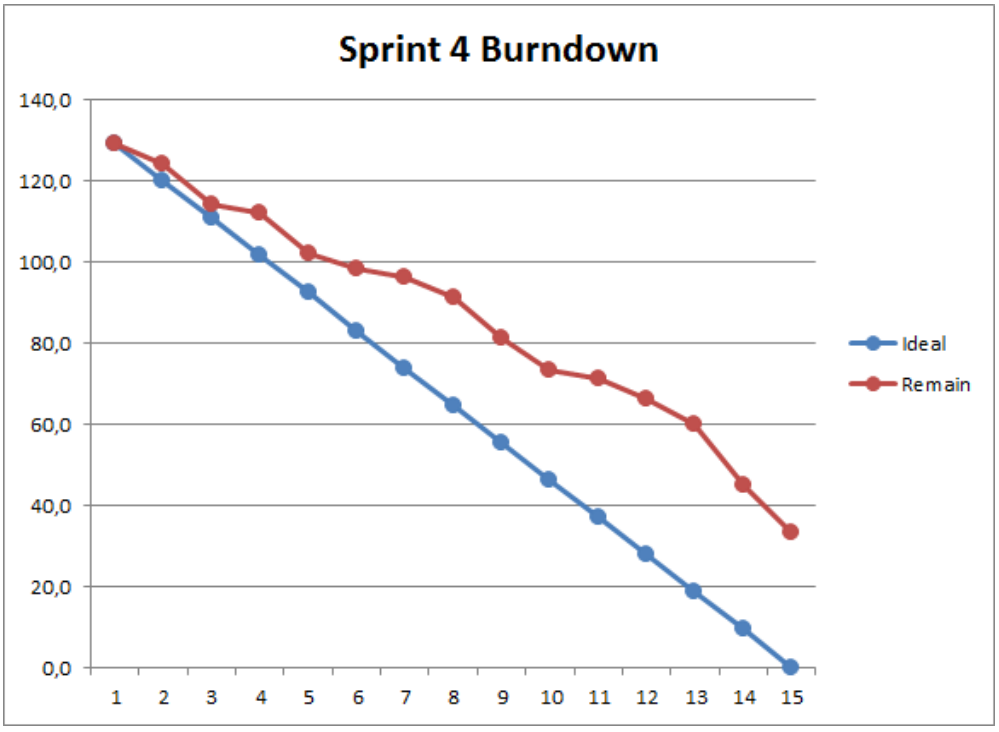


Ilustración 11-8 Sprint 4 Burndown

Cómo se puede ver a partir de los *Burndowns*, el trabajo se retrasó y no llegó a concluirse. Esto se debió a la aplicación de Scrum puro, que no se adaptaba a la dinámica del equipo. Viendo los *Burndowns*, se puede decir que los *Sprints* 3 y 4 no son exitosos. Para poder arreglar la ejecución de los *Sprint*, el SQAer revisó los *Burndowns* y se reunió con el equipo para poder evaluar maneras de mejorar los resultados. La solución tomada por todo el equipo, fue de aumentar el compromiso de cada uno para poder cumplir con sus tareas y no generar estos atrasos.

Velocidad

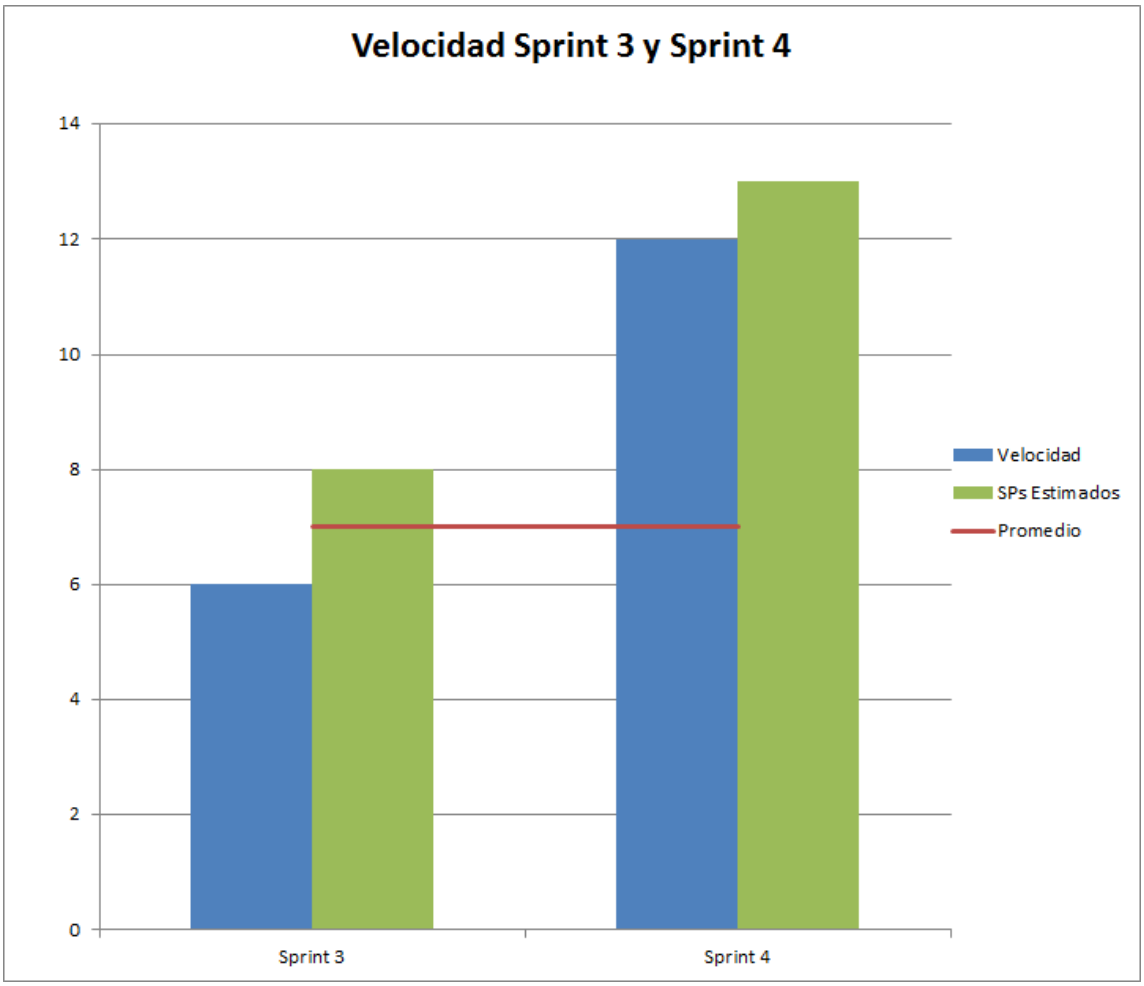


Ilustración 11-9 Velocidad Sprint 3 y Sprint 4

Cómo se mencionó anteriormente, no se pudo cumplir con los *Story Points* estimados para los *Sprints*. Estos *Sprints* siguen siendo no exitosos. Para poder mejorar este resultado, el equipo decidió, analizar nuevamente el proceso, ya que dos de sus métricas estaban dando resultados insatisfactorios.

Retrabajo

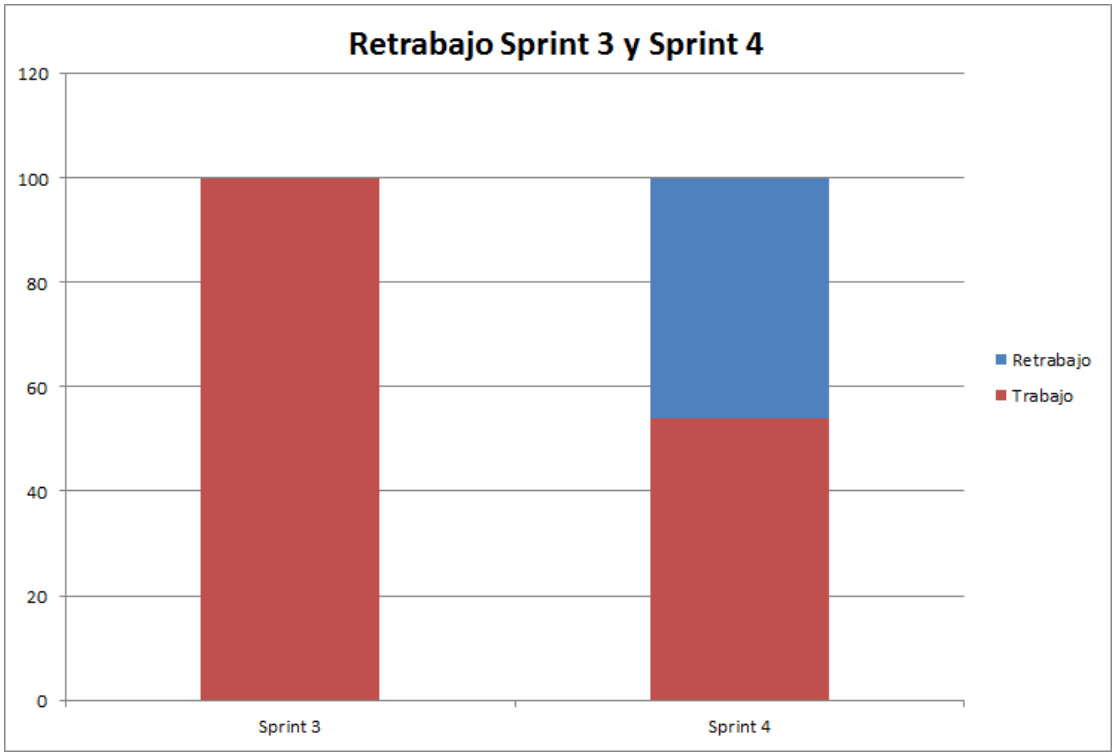


Ilustración 11-10 Retrabajo Sprint 3 y Sprint 4

El *Sprint* 3 no tuvo retrabajo gracias a la exitosa ejecución del *Sprint* 2. Sin embargo el *Sprint* 4 tuvo una gran porción de retrabajo, el cual concluye la hipótesis que los *Sprints* 3 y 4 no fueron exitosos en su implementación. Esta última métrica respaldo a la decisión del equipo en analizar el proceso actual y poder realizar mejoras sobre el mismo para que sus ejecuciones sean más exitosas.

Sprints con el segundo proceso definido (Proceso definitivo) (*Sprint* 5 a 12)

Burndown charts

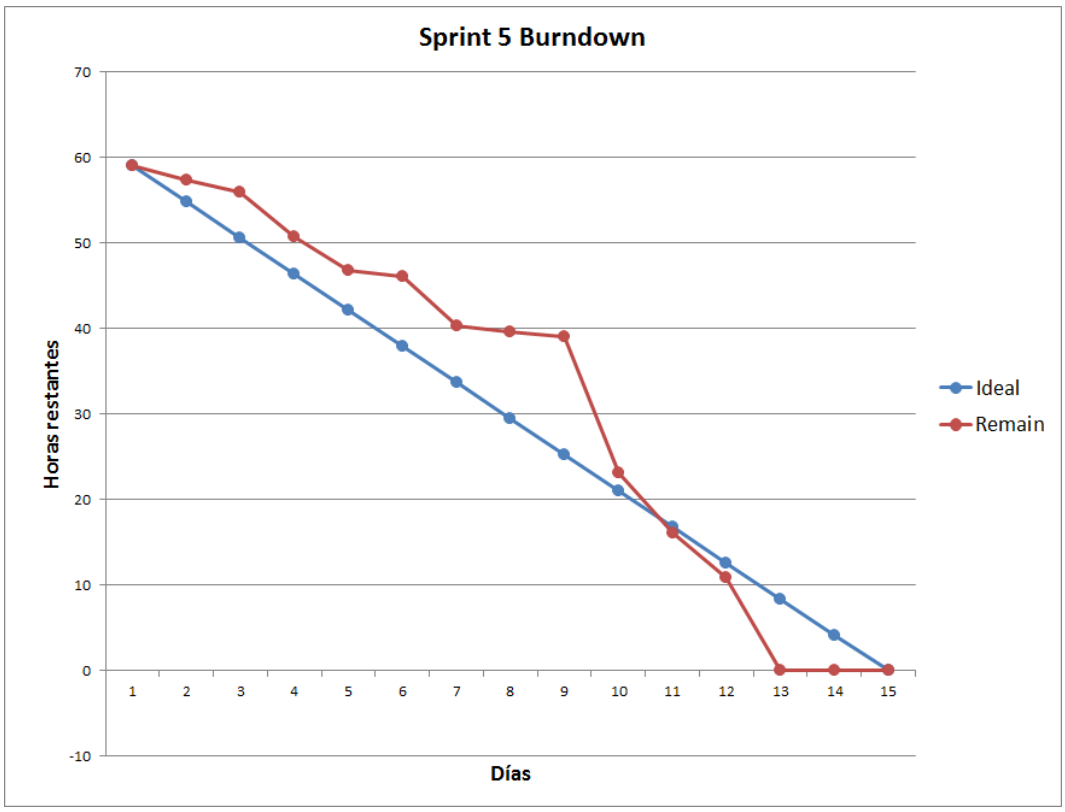


Ilustración 11-11 Sprint 5 Burndown

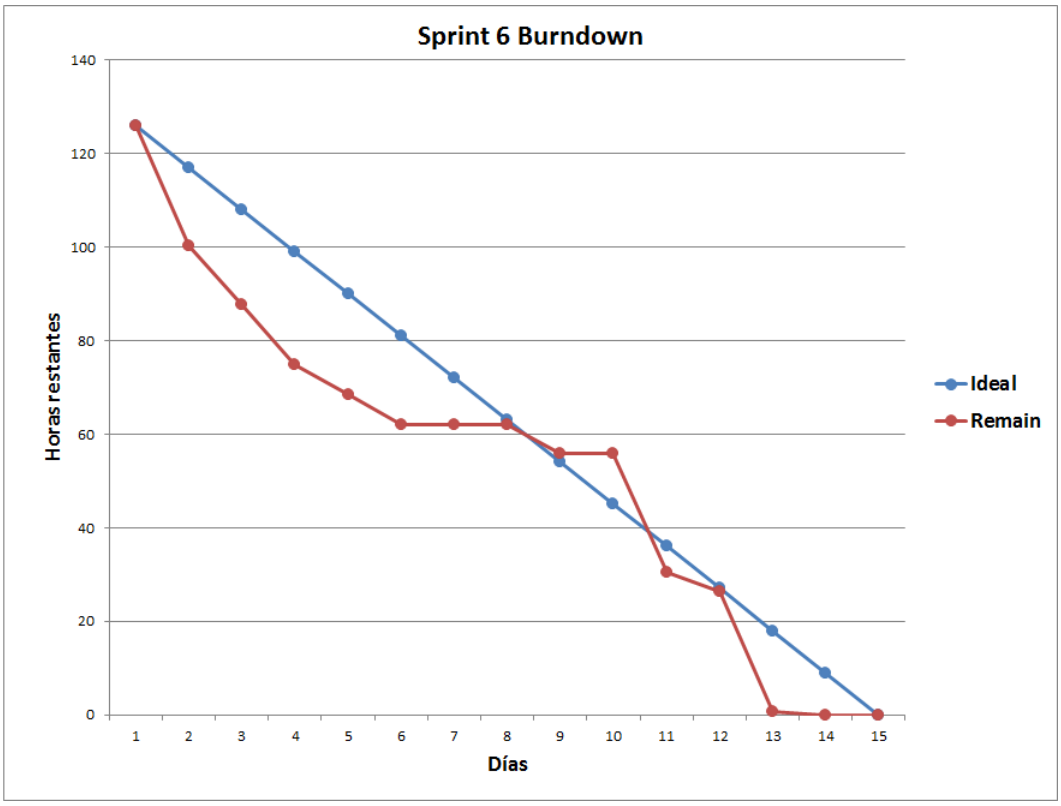


Ilustración 11-12 Sprint 6 Burndown

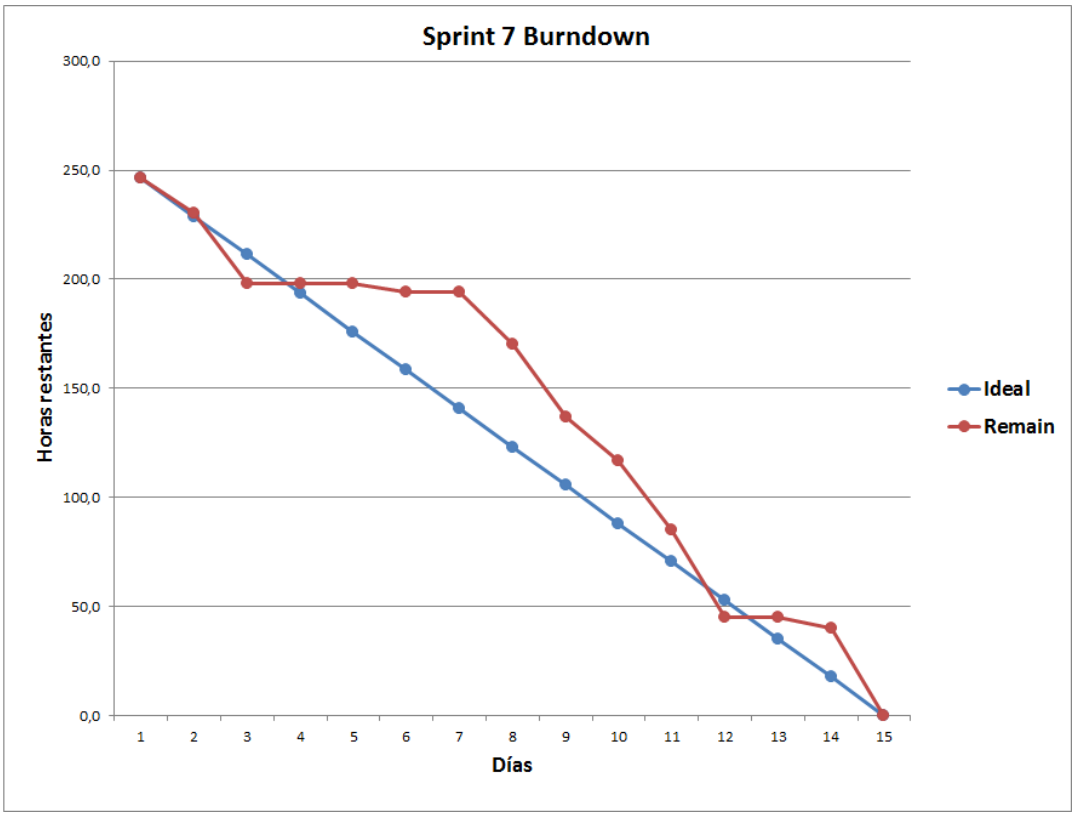


Ilustración 11-13 Sprint 7 Burndown

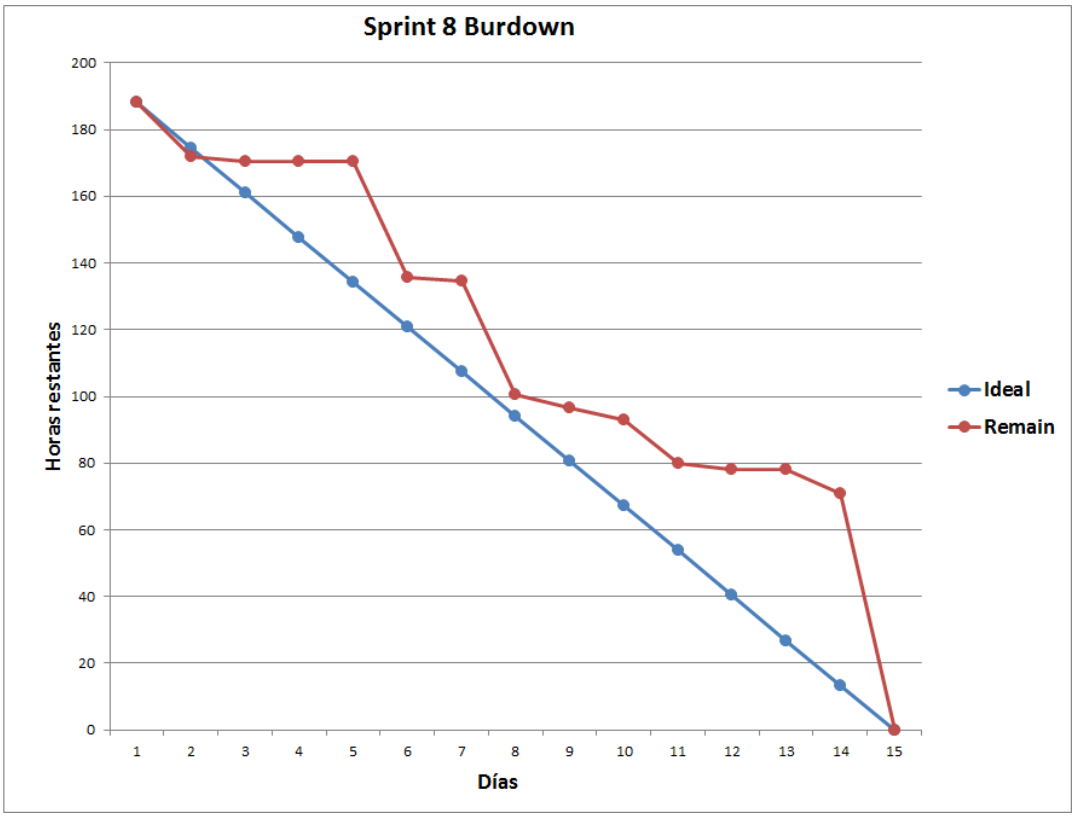


Ilustración 11-14 *Sprint* 8 *Burndown*

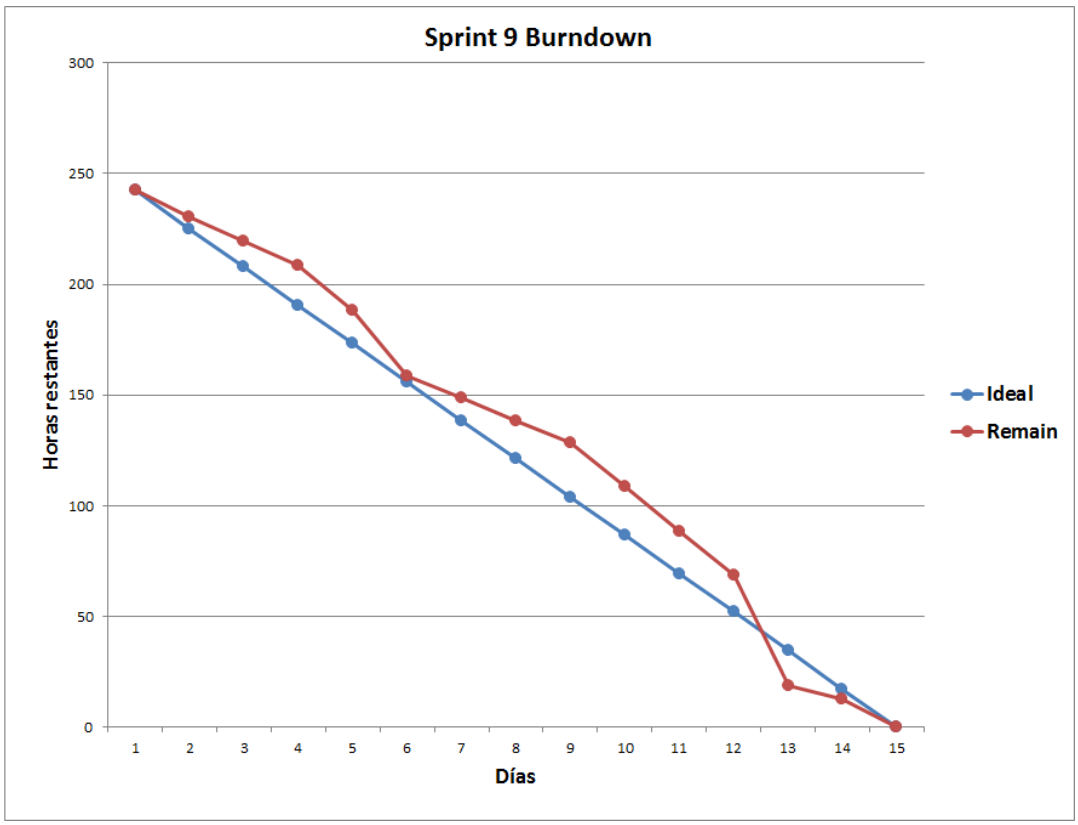


Ilustración 11-15 *Sprint* 9 *Burndown*

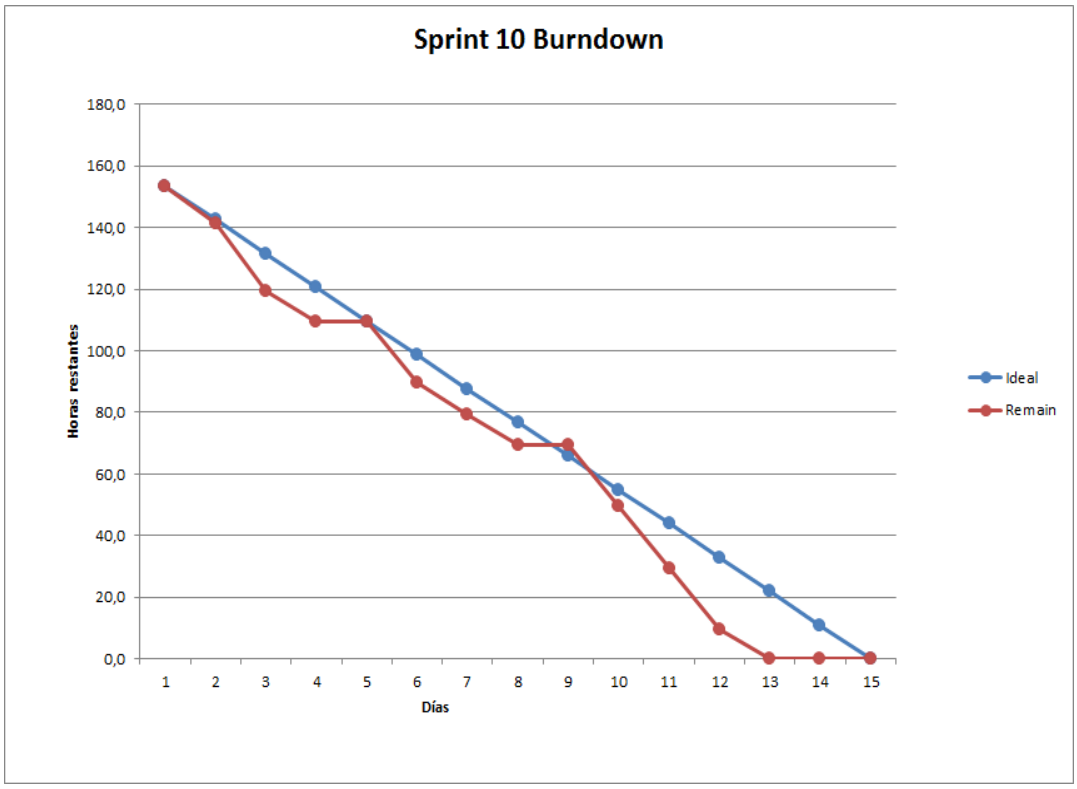


Ilustración 11-16 *Sprint* 10 *Burndown*

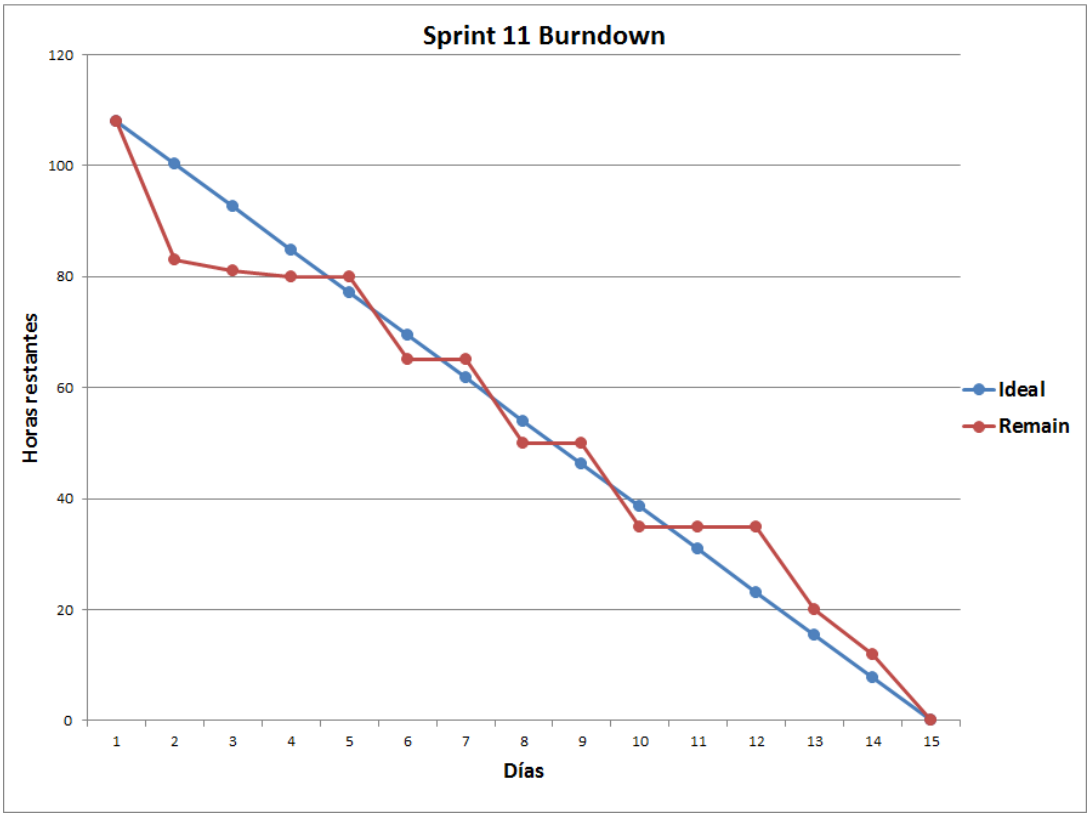


Ilustración 11-17 *Sprint* 11 *Burndown*

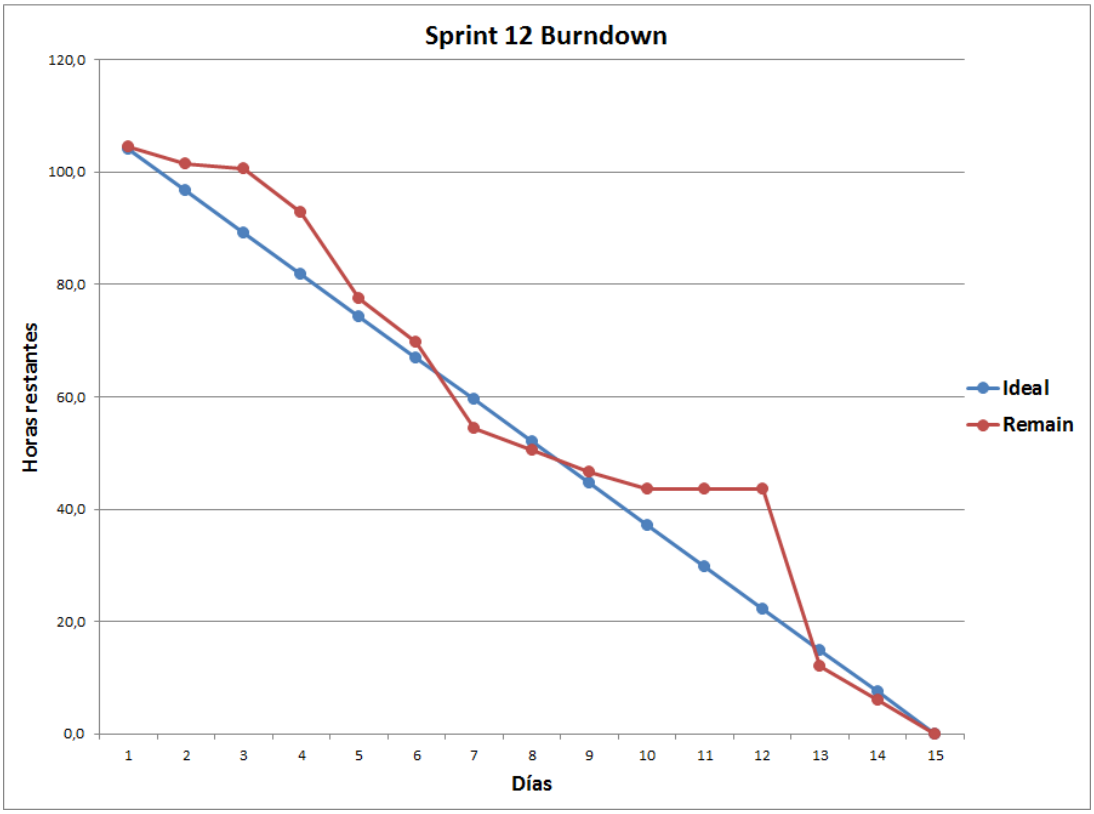


Ilustración 11-18 *Sprint* 12 *Burndown*

Al analizar los *Burndowns*, se puede ver que, si bien en algunos casos se registraron atrasos de cronograma, estos pudieron resolverse, cumpliendo con las horas de las tareas estimadas. Con respecto a los *Burndowns*, se puede afirmar, que los *Sprints* del 5 al 12 fueron exitosos. Según los *Burndowns*, el segundo proceso es más efectivo que el primero.

Velocidad

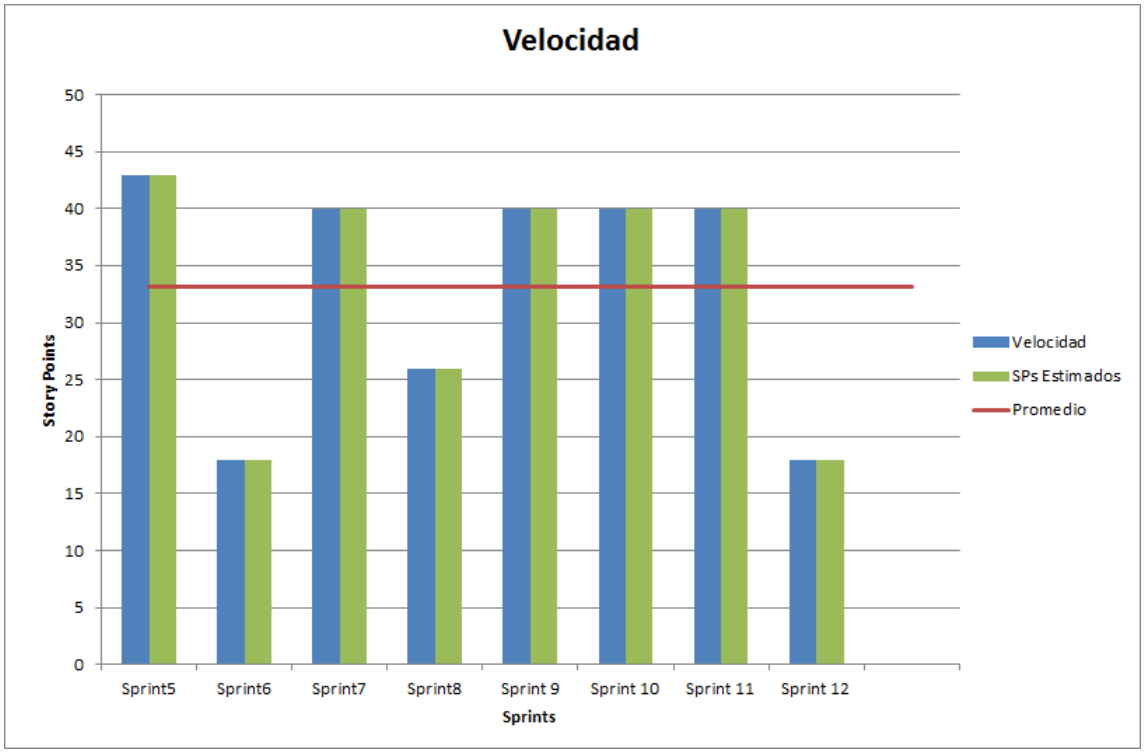


Ilustración 11-19 Velocidad *Sprint* 5 a *Sprint* 12

Todas las *Story Points* fueron cumplidas durante los 8 *Sprints*. Esto fue muy positivo para el equipo. El proceso sigue siendo efectivo.

Retrabajo

Porcentaje de retrabajo por *Sprint*

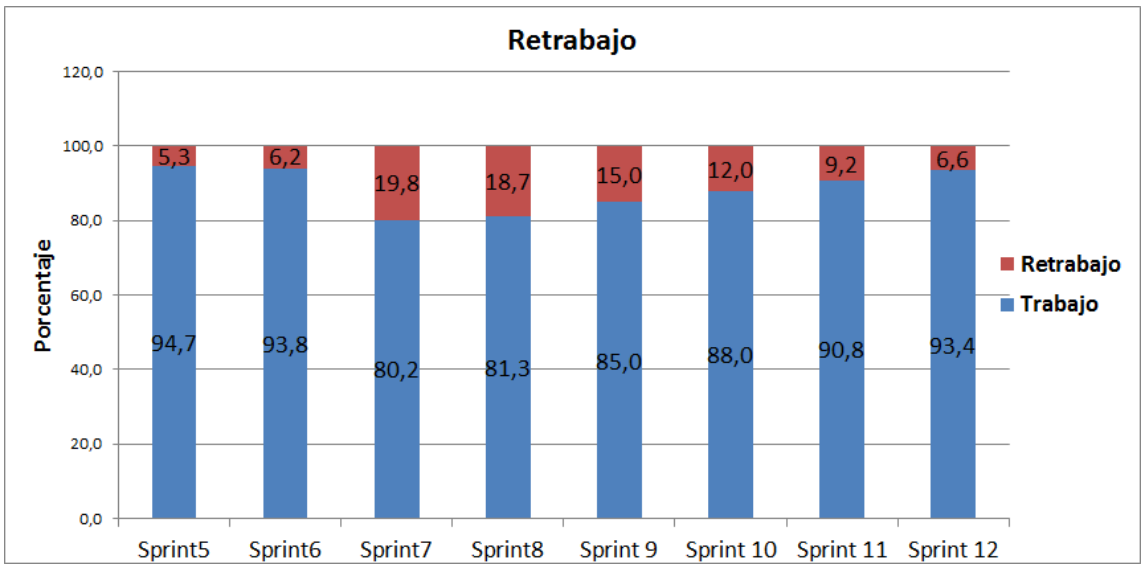


Ilustración 11-20 Retrabajo *Sprint* 5 a *Sprint* 12

Finalmente, se puede ver que el retrabajo no superó el 25% definido en la gestión de calidad y fue disminuyendo al pasar las iteraciones. Esto permitió concluir que el proceso definido se ejecutó con éxito en este proyecto.

Porcentaje de retrabajo total

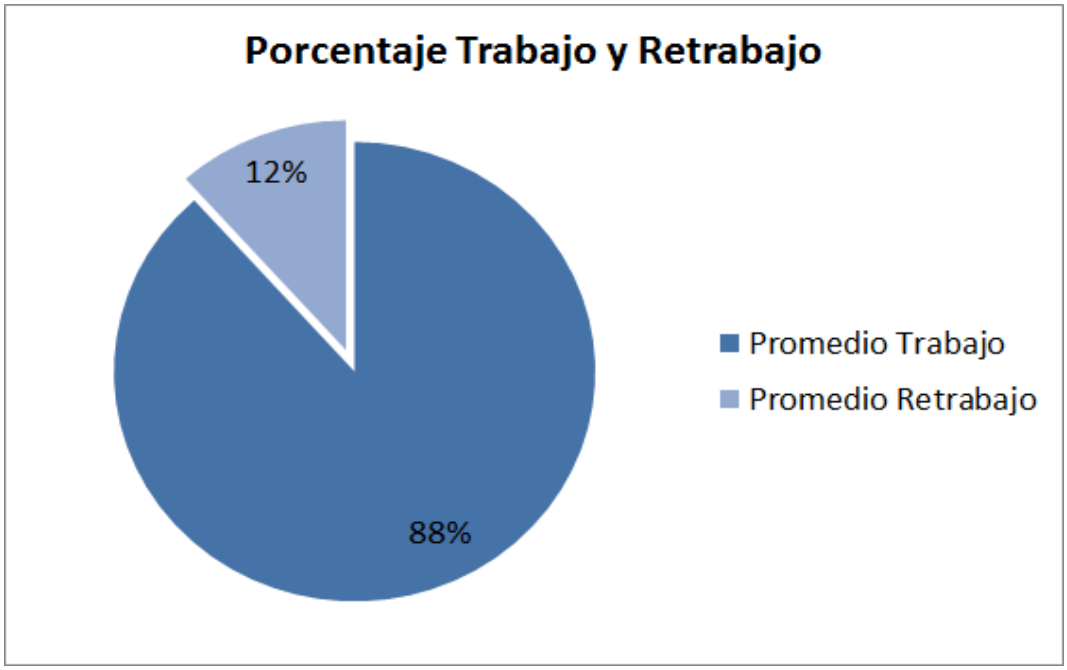


Ilustración 11-21 Porcentaje Trabajo y Retrabajo

11.9. DEFINICIÓN DEL PROCESO

Este anexo tiene el propósito de detallar las etapas del proceso y mostrar su relación con las etapas del híbrido FDS.

El proceso está formado por las siguientes etapas principales:

A. *Sprint Planning*
B. Definición de *Features*
C. Diseño y Desarrollo
D. Testing
E. *Sprint Review*
F. *Sprint Retrospective*

Estas etapas son el núcleo central del proceso. A continuación, se detallarán para entender cómo se realizan las iteraciones a más bajo nivel. A continuación se muestra el proceso detallado con sus 5 perspectivas. Cada etapa estará luego pormenorizada.

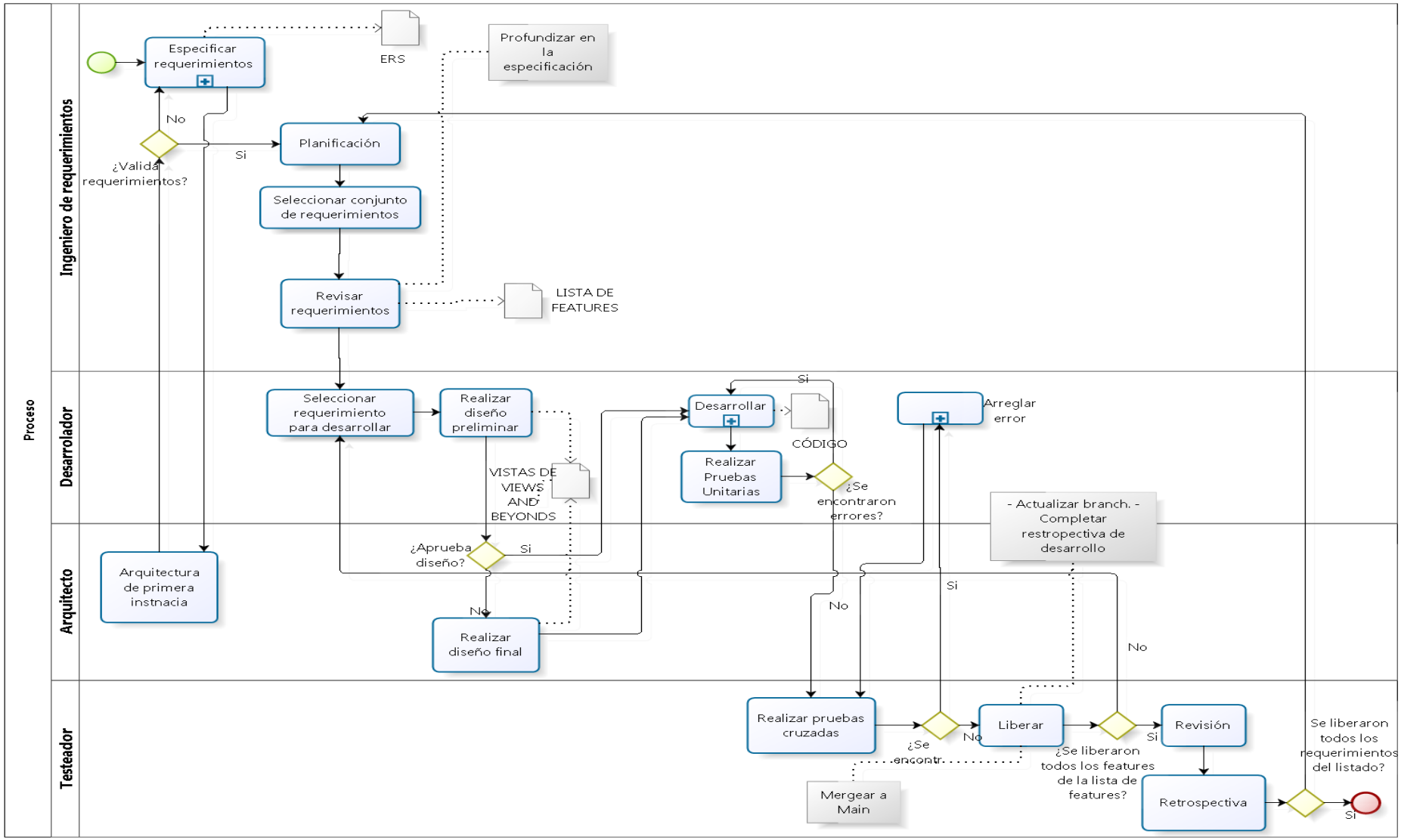


Ilustración 11-22 Proceso detallado

Detalle de las etapas del proceso

A continuación se explicará en qué consiste cada etapa del proceso a grandes rasgos.

Etapa	¿Qué se realiza?	¿Quién lo realiza?	¿Cómo se realiza?	¿Cuándo se realiza?	¿Cuáles son sus resultados?
Relevar Requerimientos	Se relevan los requerimientos	Todos	Técnicas de relevamiento	Al principio del proceso, antes de iterar	Resultados del relevamiento
Identificar Requerimientos	Se identifican los requerimientos	Ing. de requerimientos	Analizando lo relevado, se crea una lista preliminar	Después del relevamiento de requerimientos	Lista preliminar de Requerimientos
Especificar Requerimientos	Se especifican los requerimientos identificados	Ing. de requerimientos	Analizando la lista preliminar, se realiza una descripción	Después de identificar los requerimientos	ESRE - Documento de Especificación de requerimientos
Arquitectura de primera instancia	Se define una arquitectura preliminar	Arquitecto	Analizando el ESRE, se define una arquitectura adaptada	Después de relevar los requerimientos	Bosquejo de Arquitectura
Planificación	Se realiza la planificación de la iteración	Administrador de tareas	En voz alta	*Sprint Planning*	Una cantidad de horas disponibles por integrante
Seleccionar conjunto de requerimientos	Se deciden cuáles requerimientos se desarrollaran	Todos	Discutiendo	*Sprint Planning*	Lista de *Features*
Revisar *Features*	Se revisa que no haya incoherencias entre los requerimientos seleccionados	Adm. de Tareas y Arquitecto	Analizando las dependencias de los requerimientos	*Sprint Planning*	Lista de *Features* actualizada
Seleccionar *Feature* para desarrollar	Se selecciona una *Feature* que se pretende desarrollar	Desarrollador	Seleccionando una *Feature* que no esté asignada a nadie y que el desarrollador pueda realizar	Etapa de desarrollo	*Feature*
Realizar diseño preliminar	Se realiza un diseño de clases en base a la *Feature* seleccionada	Desarrollador	A partir de la *Feature* seleccionada, se realiza un diagrama de clases gracias a	Etapa de desarrollo	Diseño de clases

Tabla 11-2 Etapas del proceso			la herramienta *Star* UML		
Realizar diseño final	Se revisa el diseño preliminar realizado por el desarrollador	Arquitecto	A partir del diseño preliminar, se revisa si es coherente o si se podría mejorar	Etapa de desarrollo	Diseño final de clases
Desarrollar	Gracias al diseño final y la *Feature*, se puede empezar a desarrollarla	Desarrollador	Utilizando el IDE y lenguajes definidos	Etapa de desarrollo	Código
Realizar pruebas unitarias	Se prueba lo desarrollado	Desarrollador	Ejecutando casos de prueba	Etapa de desarrollo	Código
Realizar pruebas cruzadas	Se prueba lo desarrollado	Tester	Ejecutando casos de prueba	Etapa de *testing*	Documento de prueba
Arreglar error	Se arregla el error encontrado en la prueba cruzada	Desarrollador	Retomando el código y arreglándolo	Etapa de *testing*	Código
Liberar	Se realiza una nueva versión del producto	SCMer	Realizando un Merge con la versión estable	*Sprint Review*	Versión del Software
Revisión	Se revisan los documentos y códigos	SQAer	Abriendo los documentos, comparándolos con los estándares. Ídem para el código	*Sprint Review*	Documento de revisión
Retrospectiva	Se analizan las posibles mejores del *Sprint*	Adm. de Tareas	Realizando una lista de elementos a mejorar	*Sprint Retrospective*	Documento de retrospectiva

A continuación se brinda información adicional sobre algunas etapas.

Relevar Requerimientos

El relevamiento permite descubrir los requerimientos que serán implementados. Para lograrlo, se debe seguir una estrategia que se describe a continuación.

Se definieron las dimensiones del relevamiento de Requerimientos. Estas son, un entorno en el cual se mueve el relevamiento. Relevar es una tarea compleja, ya que muchas veces los clientes no saben exactamente lo que quieren, o muchos tienen opiniones diferentes que pueden generar conflictos. Estas dimensiones permiten mantener la consistencia de lo relevado y preguntar las preguntas correctas al momento de recolectar información.

Estas dimensiones provienen del libro [1] y son las siguientes:

Ilustración 11-23 Dimensiones del relevamiento de Requerimientos

Dominio de la aplicación

Comprende el conocimiento general sobre el área en el cual el sistema se aplicaría. Para ello se necesita realizar un trabajo de investigación para tener un entendimiento completo del área.

Problema a resolver

No solamente es importante haber encontrado el problema a resolver, pero aún más importante es entenderlo. Para poder entender el problema a resolver, es importante que la investigación que se realice en el punto anterior, comprenda todos los aspectos directos e indirectos que inciden sobre el problema.

Contexto del negocio

Generalmente, los sistemas contribuyen al desarrollo de una organización. Es importante analizar de qué forma el servicio colabora con los objetivos y metas de un equipo.

Necesidades de los interesados y restricciones

Los interesados son todas aquellas personas que se ven afectadas positivamente o negativamente por el sistema. Se deben analizar quiénes son estos interesados y analizar cuál es el impacto sobre los mismos.

Arquitectura de primera instancia

Para poder realizar esta actividad, el arquitecto debe analizar los requerimientos relevados y realizar un bosquejo de la arquitectura. Este bosquejo no es el diseño final, simplemente sirve de guía para la construcción.

Seleccionar conjunto de requerimientos

Esta etapa consiste en definir cuáles requerimientos se desarrollarán en la iteración. Los requerimientos deben estar relacionados entre sí de forma que formen *Features*. Una *Feature* es un conjunto de requerimientos dependientes entre sí.

REFERENCIAS BIBLIOGRÁFICAS

[1] G. Kotonya, *Requirements Engineering: Processes and Technique*, USA: John Wiley & Sons Ltd., 1998.

11.10. ENCUESTAS Y ENTREVISTAS

PRIMERA ENCUESTA

Resumen de la encuesta realizada a futuros usuarios del sistema.

Pregunta 1:

¿Cuál es su primera impresión acerca de la herramienta presentada?

Respuestas relevantes:

R1: Buena iniciativa. Hay que contemplar que ya existen herramientas similares y que la entrada al mercado pueda ser difícil.

R2: Puede convertirse en una herramienta muy poderosa para la toma de decisiones y la gestión de los cultivares.

R3: Es un proyecto un poco ambicioso pero que puede tener futuro

R4: Da una muy buena impresión, ya que ayudaría a llevar la gestión y el ordenamiento de las tareas a realizar chacra a chacra.

R5: Muy positivo una alternativa interesante para llevar los datos más ordenados y ser más eficiente a la hora de realizar tareas. Es muy interesante el sistema que avisa que se deben realizar tareas. En mi caso trabajo campos muy alejados unos de otros sería bueno para tener un poco más ordenado con la planificaciones de uso y cultivos.

Pregunta 2:

¿Cree que podría apoyarlo en la toma de decisiones?

Respuestas: Todas las respuestas fueron “Si”, menos una que fue “Depende del contexto”

Pregunta 3:

¿Usted realmente lo usaría?

Respuestas: Todas las respuestas fueron “Si”, menos una que fue “Si es gratis”

Pregunta 4:

¿Qué ventajas ve de usar esta herramienta?

Respuestas relevantes:

R1: Me facilita la gestión, porque me mantiene dentro de las normativas y al mismo tiempo, monitorea globalmente la gestión y me avisa de desvíos del plan.

R2: Agilidad, registro y estudio integral de la actividad. Muchas de las funcionalidades que presentan ya existen en el EROSION 6.0 pero la parte de gestión es una innovación interesante.

R3: Mejor rendimiento de los cultivos, aumento de ganancias. Puede ayudar a organizarme mejor en mis tareas así como tener un control e histórico de la evolución de mis cultivos.

R4: Como productor ayudaría a realizar una gestión diferenciada en cada una de las chacras y de esta manera finalizada la zafra sacar números, ver cuales chacras cerraron con mejores números y cuáles no y si hay alguna chacra que por varias zafras seguidas viene dando negativo analizarla con más detalle e identificar el problema para tratar de resolverlo.

R5: Ordenar los datos, ser más eficiente en los cálculos de los costos, y ser más organizado con las tareas.

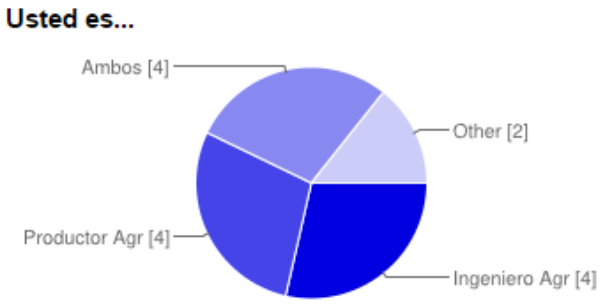
Pregunta sobre funcionalidades:

Respuestas: Todas las respuestas están entre 4 y 5 (interesantes y muy interesantes). La que obtuvo mayor puntaje es la número 10 (Como administrador de cultivos, me gustaría estar informado sobre clima, precio de *commodities*, cotizaciones de monedas, precios de insumos, noticias del agro, etc.) que obtuvo solo 2 valores 4 ("interesante").

Comentarios interesantes:

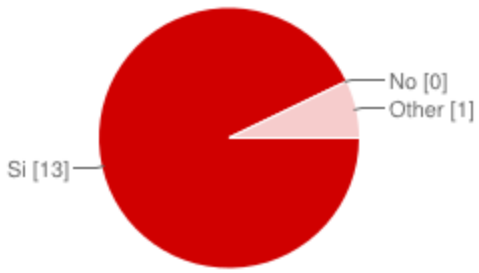
Lo que estaría bueno también, si fuera posible, con estaciones meteorológicas básicas instaladas en los establecimientos de los productores, que el sistema vaya creando un registro de precipitaciones, temperatura, humedad y viento porque en el caso de que se realice una tarea y la misma falle por ejemplo una aplicación de fungicida, se podría analizar cómo fueron las condiciones el día en que se realizó dicha tarea y ver si la falla fue por malas condiciones o por una error operativo. Por lo demás está muy bueno. Éxitos!

A continuación se muestran las gráficas de respuestas:



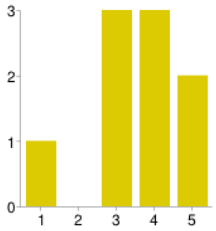
Ingeniero Agrónomo	4	25%
Productor Agrícola	4	25%
Ambos	4	25%
Other	2	13%

¿Usted realmente lo usaría?



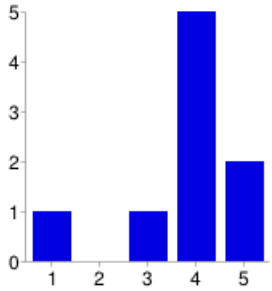
Si	**13**	81%
No	**0**	0%
Other	**1**	6%

1. Como administrador de cultivos, quiero que el sistema me muestre un detalle estándard de todos los gastos apróximados que voy a tener para cada uno de los cultivos, de acuerdo al plan elegido.



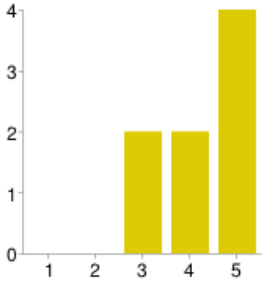
1	**1**	6%
2	**0**	0%
3	**3**	19%
4	**3**	19%
5	**2**	13%

4. Como administrador de cultivos, quiero que el sistema me muestre un detalle standard de las tareas a realizar, indicando fechas, insumos, mano de obra, etc.



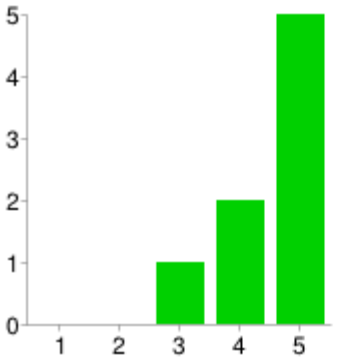
1	**1**	6%
2	**0**	0%
3	**1**	6%
4	**5**	31%
5	**2**	13%

7. Como administrador de cultivos, quiero ingresar el seguimiento de mis cultivos, saber como estoy con respecto al presupuesto y al cronograma.



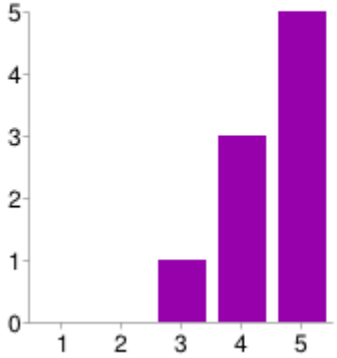
1	**0**	0%
2	**0**	0%
3	**2**	13%
4	**2**	13%
5	**4**	25%

8. Como administrador de cultivos, quiero que el sistema me alerte los desvíos del plan.



1	0	0%
2	0	0%
3	1	6%
4	2	13%
5	5	31%

9. Como administrador de cultivos, quiero que el sistema me alerte de los hitos a realizar en corto plazo.



1	0	0%
2	0	0%
3	1	6%
4	3	19%
5	5	31%

SEGUNDA ENCUESTA – USABILIDAD

Encuesta de Usabilidad

En esta encuesta se pretende evaluar el nivel de usabilidad del sistema por parte de nuestros expertos en el dominio. La encuesta está formada de 6 preguntas con puntaje del 1 al 5, siendo 1 el menor puntaje "Confuso" y 5 el más alto "Claro"

Apariencia General del Sistema

	1	2	3	4	5	
Confuso	○	○	○	○	○	Claro

Navegación entre los distintos módulos

	1	2	3	4	5	
Confuso	○	○	○	○	○	Claro

Estructura de la aplicación

	1	2	3	4	5	
Confuso	○	○	○	○	○	Claro

Facilidad de realizar acciones

	1	2	3	4	5	
Confuso	○	○	○	○	○	Claro

Respuesta de la acción de los botones/iconos/menus

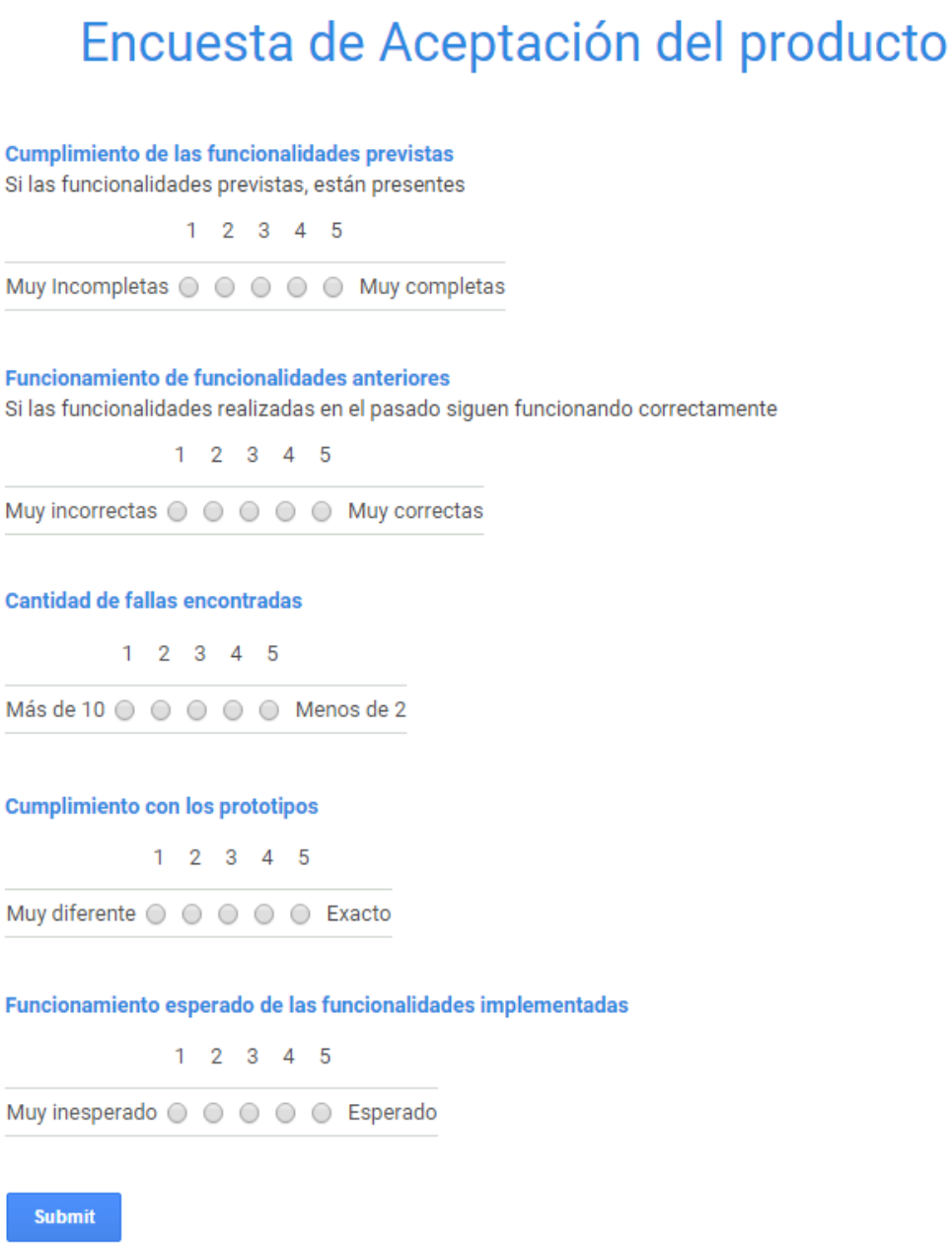
	1	2	3	4	5	
Confuso	○	○	○	○	○	Claro

Colores utilizados en la aplicación

	1	2	3	4	5	
Confuso	○	○	○	○	○	Claro

Submit

TERCERA ENCUESTA - ACEPTACIÓN DEL PRODUCTO



Encuesta de Aceptación del producto

Cumplimiento de las funcionalidades previstas
Si las funcionalidades previstas, están presentes

1 2 3 4 5

Muy Incompletas ○ ○ ○ ○ ○ Muy completas

Funcionamiento de funcionalidades anteriores
Si las funcionalidades realizadas en el pasado siguen funcionando correctamente

1 2 3 4 5

Muy incorrectas ○ ○ ○ ○ ○ Muy correctas

Cantidad de fallas encontradas

1 2 3 4 5

Más de 10 ○ ○ ○ ○ ○ Menos de 2

Cumplimiento con los prototipos

1 2 3 4 5

Muy diferente ○ ○ ○ ○ ○ Exacto

Funcionamiento esperado de las funcionalidades implementadas

1 2 3 4 5

Muy inesperado ○ ○ ○ ○ ○ Esperado

Submit

CUARTA ENCUESTA - SATISFACCIÓN DEL EQUIPO

Cómo se mencionó en la Gestión de Riesgos, uno de los Riesgos definidos se relacionaba con los conflictos del equipo. En el plan de respuesta a dicho riesgo, se realiza una encuesta de Satisfacción del Equipo para saber cuál podría ser el problema y que tan contento y proactivo es el equipo. A continuación se muestra esta encueta:

Satisfacción del equipo

¿Que tan competentes son los otros miembros del equipo?

	1	2	3	4	5	
Poco competentes	○	○	○	○	○	Muy competentes

¿Que tan honestos son los miembros del equipo?

	1	2	3	4	5	
Muy poco honestos	○	○	○	○	○	Muy honestos

¿Que tan eficientes son las reuniones en equipo?

	1	2	3	4	5	
Poco eficientes	○	○	○	○	○	Muy eficientes

¿Que tan buena es la comunicación entre los miembros del equipo?

	1	2	3	4	5	
Muy mala	○	○	○	○	○	Muy buena

¿Que tan bien los miembros aceptan críticas?

	1	2	3	4	5	
Las aceptan muy mal	○	○	○	○	○	Las aceptan muy bien

¿Que tan bien se tratan los miembros del equipo?

	1	2	3	4	5	
Muy mal	○	○	○	○	○	Muy bien

¿Que tan orgulloso estás de tu trabajo?

	1	2	3	4	5	
Muy poco orgulloso	○	○	○	○	○	Muy orgulloso

11.11. PRIMER CONGRESO URUGUAYO DE SUELOS - AGOSTO 2014

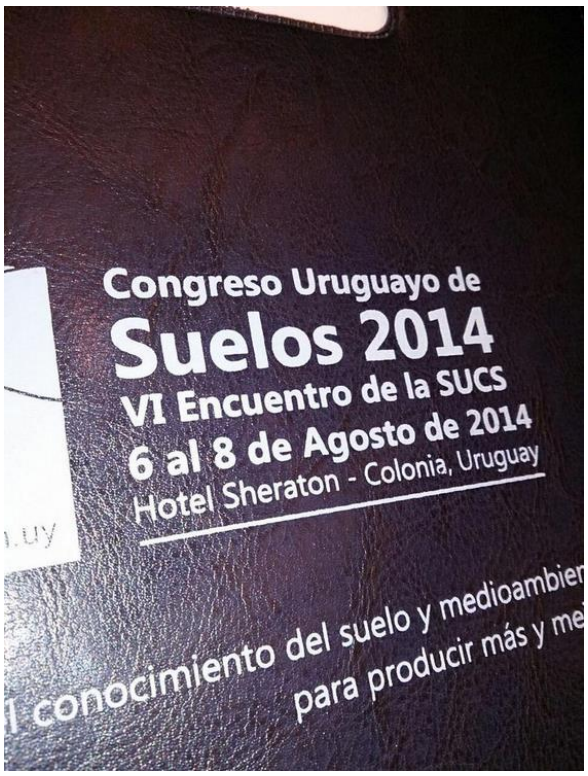


Ilustración 11-24 Folleto Congreso de Suelos

¿En qué consistió el congreso?

El congreso de suelos, son unos días en los cuales, se reúnen ingenieros agrónomos y productores agrícolas para escuchar charlas presentadas por profesores, Ingenieros Agrónomos, estudiantes en agronomía, etc. [1]

Por más información sobre el congreso: http://www.suelos.com.uy/

¿Cómo fue la participación de SIGAMAS en el congreso?

El equipo tuvo la oportunidad de formar parte del Congreso Uruguayo de Suelos, que se realizó en el Hotel Sheraton de Colonia en Agosto 2014.

Pablo Montes fue el responsable de brindar la información y de averiguar si el equipo podía registrarse en calidad de estudiante en el congreso. Esta calidad da la oportunidad de presentar un póster con información sobre el proyecto siendo desarrollado. Generalmente, estos posters son realizados por estudiantes de la facultad de Agronomía quienes los utilizan para exponer resultados de experimentos.

El equipo aprovechó esta oportunidad para crear un póster con información sobre SIGAMAS, así como imágenes con prototipos.

Además de presentar el póster, Andrés Nieves presentó SIGAMAS oralmente, delante de todos los participantes del congreso.

Finalmente, se realizaron encuestas a los participantes y se repartieron tarjetas de negocio para que puedan entrar al sitio, llenar encuestas, y contactarnos ante cualquier duda.

A continuación se muestran fotos del congreso:

SISTEMA DE GESTIÓN AGRÍCOLA Y MANEJO DE SUELOS

Ana Silva | Andrés Nieves | Clara Youdale | Mauro Varela

INTRODUCCIÓN

La erosión es el problema ambiental más importante que tiene Uruguay, el 30% del suelo del país está erosionado, entre 29 y 80 toneladas se pierden de suelo por año, siendo el 87% del área afectada la región agrícola.
En nuestro país no existe una herramienta que conjugue varias tecnologías para precisar todas las características de los suelos del Uruguay y que, además, brinde la posibilidad de planificar y ejecutar las rotaciones de los cultivos. Todo esto, considerando un correcto manejo de suelos, las capacidades de uso de cada uno de ellos y la comparación del costo-beneficio que cada uno puede ofrecer.
Por esto, SIGAMAS pretende ser un servicio que ofrezca a los productores agrícolas toda la información y las herramientas necesarias para planificar y ejecutar sus rotaciones de cultivos. SIGAMAS brindará apoyo en la toma de decisiones, ayudará a aumentar las ganancias en la producción y a aprovechar mejor las capacidades de sus suelos, evitando la degradación de los mismos.

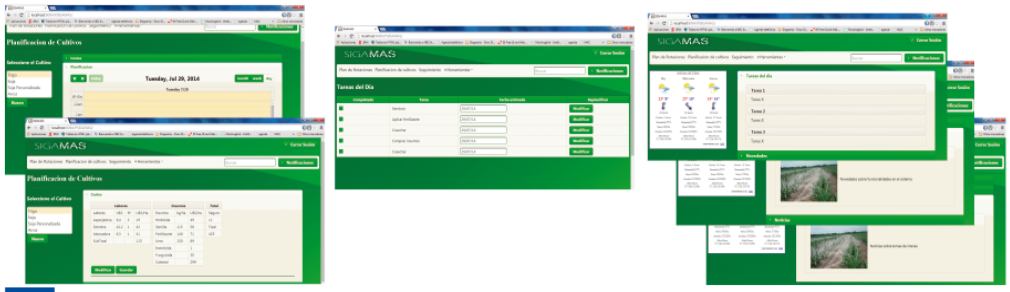
MATERIALES Y MÉTODOS

Para cumplir con la propuesta y construir un producto de gran calidad que se ajuste a las necesidades de los productores agrícolas se hicieron encuestas, cuestionarios y entrevistas con expertos del sector y potenciales clientes (se estudiaron detenidamente cada una de sus necesidades y se buscaron soluciones aplicando herramientas y métodos de ingeniería), así como estudios comparativos ("benchmarking") con software internacionales relacionados con nuestro objetivo.

RESULTADOS

Contará con un módulo de procesamiento de datos, con el objetivo de sugerir alternativas y diagnosticar posibles desvíos. Para analizar y brindar información precisa, se realizará un profundo trabajo de investigación y de recolección de datos.

PLANIFICACIÓN | EJECUCIÓN | INFORMACIÓN



CONCLUSIONES

SIGAMAS facilita la elaboración del plan de uso de suelo, y una vez tomada esta decisión, brinda las herramientas necesarias para planificar, gestionar, monitorear y mejorar todos los procesos de producción de sus cultivos teniendo en cuenta la minimización del impacto ambiental.
El productor podrá saber en cualquier lugar y momento el estado actual de sus cultivos, teniendo la información necesaria para tomar las decisiones correctas cuando sea necesario.

Ilustración 11-25 Poster

Ilustración 11-26 Ana, Clara y Mauro con el Póster

Ilustración 11-27 Andrés presentando a SIGAMAS

Ilustración 11-28 , Clara, Mauro, Andrés y el Experto del Dominio Tabaré Alcorta

REFERENCIAS BIBLIOGRÁFICAS

[1] Congreso De Suelos 2014. [En línea]. Disponible: http://www.suelos.com.uy

11.12. SITIO WEB SIGAMAS

El equipo cuenta con un sitio web www.SIGAMAS.com que le permitió publicar noticias sobre el desarrollo e información del producto. Además, en el sitio se pueden encontrar las distintas encuestas que permitieron realizar los relevamientos de requerimientos y conocer a los futuros usuarios.

Se realizó el sitio para poder presentar SIGAMAS más amigablemente y poder llegar a la mayor cantidad de personas posibles. Además se creó una página en Facebook y una cuenta en Twitter para poder notificar el avance y mantener a los interesados actualizados. La cuenta de Twitter tiene más de 100 seguidores, los cuales se encuentran en el rubro agrícola y pudieron llenar las encuestas.

Las páginas y cuentas fueron difundidas durante el congreso de suelos en Colonia, mediante tarjetas de negocio. Gracias a ellos se pudieron concretar una gran parte de las entrevistas con entes del rubro agrícola.

A continuación se muestran imágenes de captura del sitio.

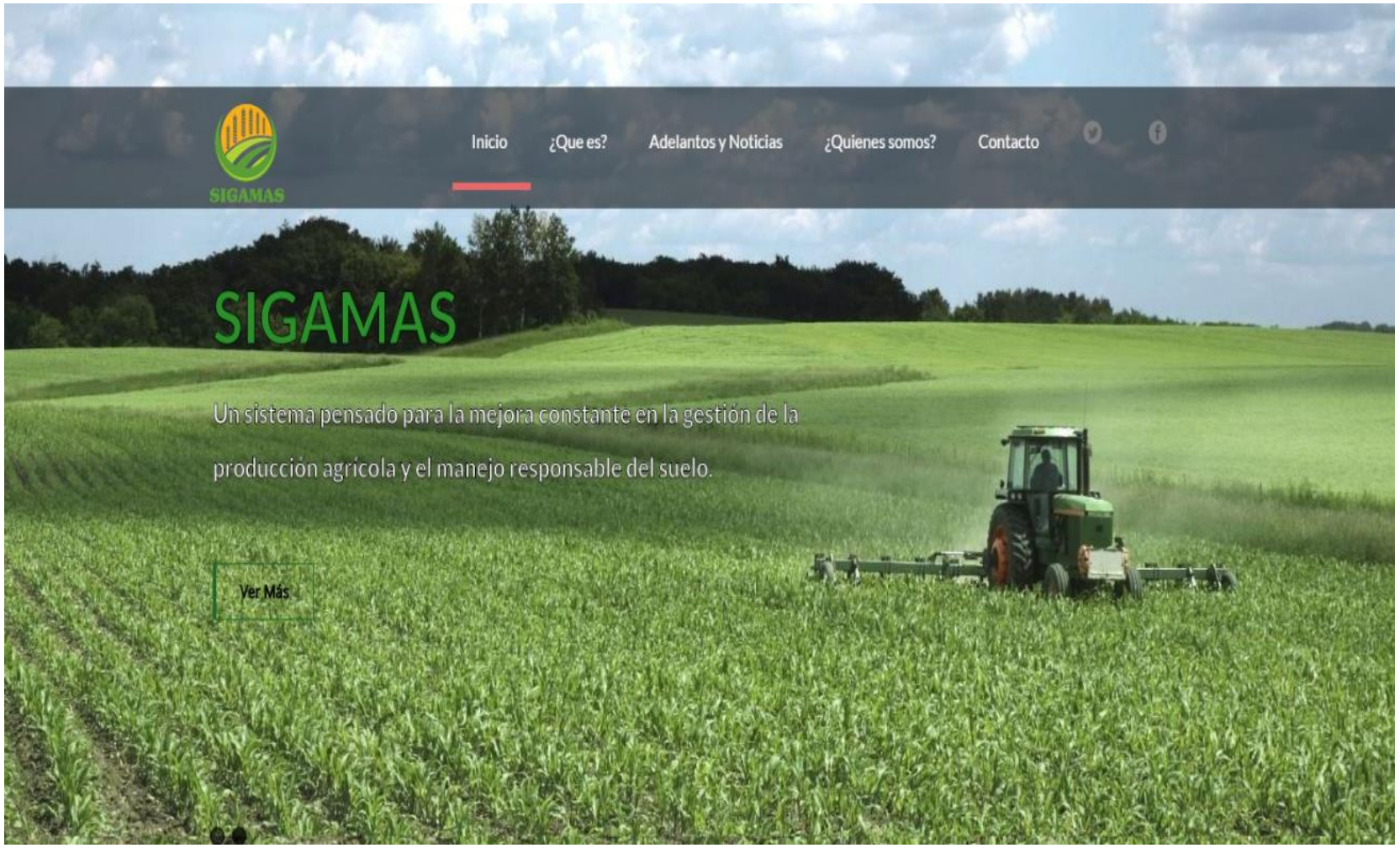


Ilustración 11-29 Página principal del sitio

Ilustración 11-30 Otro contenido del sitio

A continuación se muestran capturas de las páginas de Facebook y cuenta de Twitter.

Ilustración 11-31 Página Facebook

Ilustración 11-32 Cuenta Twitter

11.13. ESTÁNDAR DE ESPECIFICACIÓN DE REQUERIMIENTOS

A continuación se detalla el estándar que debe seguir la Especificación de Requerimientos.

Cada requerimiento debe contener los siguientes datos:

Módulo: (solo para Requerimientos Funcionales)

Cada requerimiento pertenece a un módulo específico y esto se detalla en la parte "Módulo" del estándar. Esto permite facilitar la búsqueda de un requerimiento por módulo.

Identificación:
Para diferenciar los requerimientos se utiliza una identificación propia para cada requerimiento. Esta identificación se compone de RF o RNF, dependiendo si es un requerimiento funcional o un requerimiento no funcional, seguido por un número que debe ser único para cada uno.

Nombre:
El nombre de cada requerimiento debe ser descriptivo y no debe contener más de 10 palabras.

Actor: (solo para Requerimientos Funcionales)

Los actores son los distintos tipos de usuarios que interactúan con el sistema. Estos pueden ser personas físicas o sistemas.

Descripción:
La descripción del requerimiento debe seguir las siguientes reglas:

- El lenguaje usado debe ser simple, sin usar términos engorrosos. Si de necesitarse emplear términos complejos, se debe especificar su significado.
- El lenguaje usado debe ser conciso. Las frases deben ser precisas para evitar ambigüedades o mal entendimientos.
- El lenguaje usado debe ser consistente.
- La descripción no puede sobrepasar las 5 líneas.

No es obligatoria para los requerimientos no funcionales

Especificación: (solo para Requerimientos Funcionales)

La especificación refiere a listar el nombre de los prototipos asociados a este requerimiento. Según I. Sommerville y P. Sawyer [1], usar diagramas (prototipos) puede ayudar con el entendimiento de los requerimientos. Muchas veces, los diagramas pueden ser más efectivos que el texto plano para explicar o mostrar información. Además, los diagramas en varias ocasiones dicen más, usando menos palabras. Por último, pueden ser útiles para mostrarle al cliente una visión directa del producto, evitando inconsistencias.

Se utilizarán entonces prototipos para complementar la especificación de los requerimientos.

Los prototipos es que deben ser lineales con la descripción del requerimiento. Sirven para representar un bosquejo de la aplicación y pueden tener comentarios asociados, directamente sobre el prototipo. (Nota: cuando se realice el listado de los requerimientos, se incluirán los prototipos en cada *Feature*)

Prioridad: (solo para Requerimientos Funcionales)

La prioridad sirve para especificar qué tan rápido se necesita desarrollar este requerimiento.

La prioridad puede tener uno de los siguientes valores: Alto, Medio, Bajo.

Clasificación: (solo para Requerimientos Funcionales)

La clasificación de los requerimientos refiere a especificar la criticidad de los mismos. Estos pueden ser Críticos o No Crítico.

***Story Points*: (solo para Requerimientos Funcionales)**

La cuantificación debe poder definir la complejidad del desarrollo de un requerimiento.

Implementado: (solo para Requerimientos Funcionales)

En este punto se aclara en qué estado de la implementación se encuentra el requerimiento al finalizar el proyecto de fin de carrera.

Criterios de aceptación:

Los criterios de aceptación se definen para los casos de pruebas realizados para los requerimientos clasificados como críticos en el caso de ser requerimientos funcionales. Para los requerimientos no funcionales este campo es obligatorio si se trata de un atributo de calidad y define la cuantificación del requerimiento correspondiente.

REFERENCIAS BIBLIOGRÁFICAS

[1] I. Sommerville, *Requirements Engineering: A Good Practice Guide, USA:* John Wiley & Sons Ltd., 1997.

11.14. ESRE- DOCUMENTO DE ESPECIFICACIÓN DE REQUERIMIENTOS

Introducción

Para especificar los requerimientos como lo preestablece el estándar (ver Anexo "Estándar de Especificación de Requerimientos"), la especificación de requerimiento debe seguir un formato especial el cual está formado principalmente de un nombre, una descripción, una especificación y criterios de aceptación. Si bien la especificación es realizada con prototipos, en el ESRE se incluye las pantallas de este para su mejor visualización.

Fecha	Versión	Quién	Descripción	Revisión
04/05/2014	1.0	Ana	Creación del documento ESRE	08/05/2014
07/06/2014	1.1	Ana	Actualización del ESRE	15/06/2014
16/07/2014	1.2	Ana	Actualización del ESRE	17/07/2014
12/08/2014	1.3	Ana	Actualización del ESRE	19/08/2014
28/08/2014	1.4	Ana	Actualización del ESRE	28/08/2014
25/09/2014	1.5	Ana	Actualización del ESRE	25/09/2014
01/12/2014	1.6	Ana	Actualización del ESRE	04/12/2014
10/12/2014	1.7	Ana y Mauro	Actualización del ESRE	18/12/2014

Objetivo
El objetivo de este documento es especificar los requerimientos funcionales y no funcionales que el producto satisfará. Cada Requerimiento tendrá la siguiente información: Un identificador, el módulo al cual pertenece, un nombre, una descripción, una especificación, una prioridad y una clasificación.

Criterios
El criterio para decidir si un Requerimiento es crítico o no depende de la complejidad del mismo. Con esto se refiere a la cantidad de transacciones a la base de datos que realice como a la lógica que contenga.

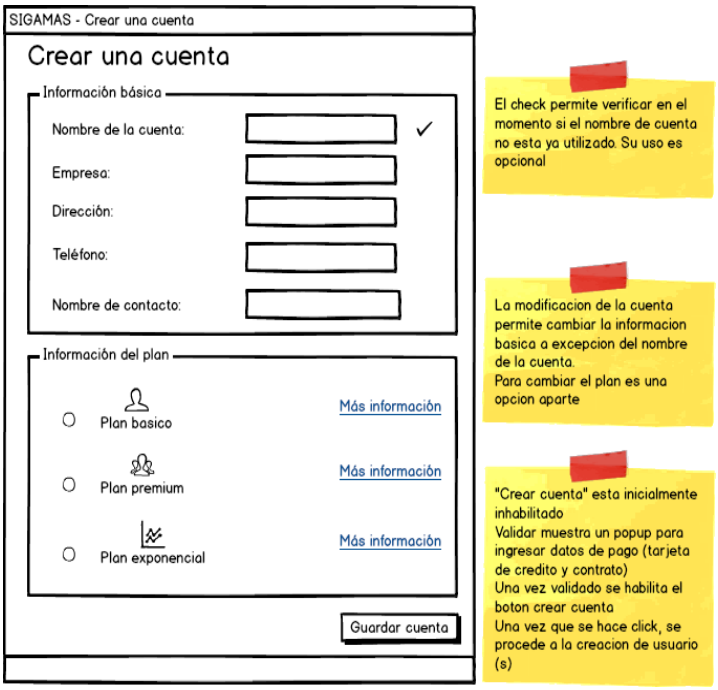
Requerimientos funcionales

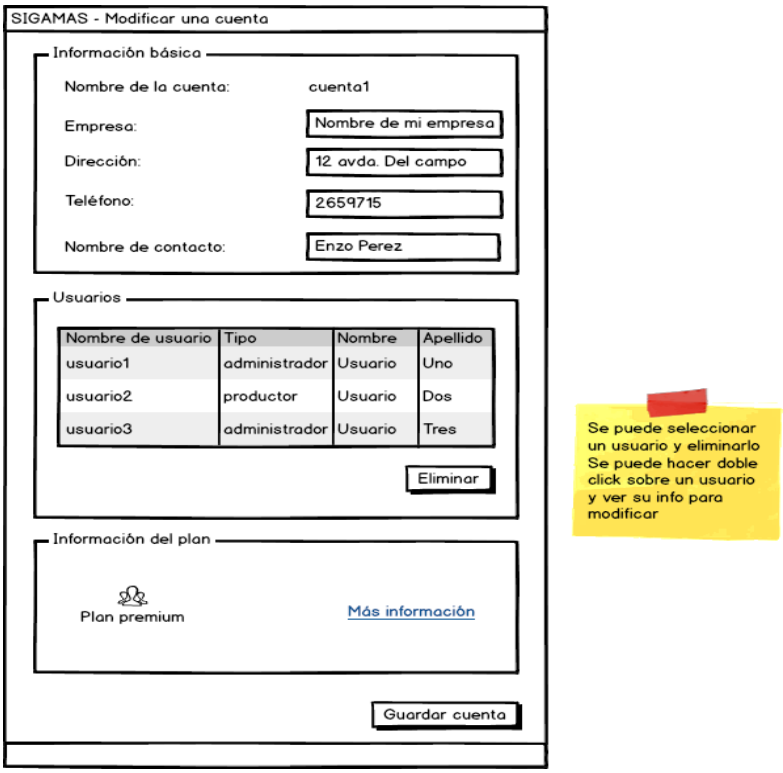
Módulo Seguridad

RF01
Nombre: ABM Cuenta de empresa
Actor: Administrador del sistema

Descripción: El sistema debe permitir la creación, modificación y eliminación de cuentas. Cada cuenta debe estar asociada con una empresa agrícola o empresa de agro-insumos. Las empresas proveedora de insumos para el agro deben tener cuentas maestras que solamente pueden crear cuentas para las empresas de producción agrícola. Las cuentas de producción agrícolas deben poderse vincular con los proyectos de cultivos y dar permisos a diferentes usuarios. Las cuentas se deben registrar con nombre de la cuenta, nombre de la empresa y RUT.
Prioridad: Alta
Clasificación: No crítico
Story Points: 13
Implementado: Se pueden crear todas cuentas por *Backoffice*, falto implementar la creación de cuentas de empresas agrícolas por parte de las empresas proveedoras de insumos.

Especificación:





RF02

Nombre: ABM Usuario

Actor: Usuario administrador

Descripción: El sistema debe permitir la creación, modificación y eliminación de diferentes tipos de usuarios. Ya sea el usuario administrador de la empresa que vende insumos para el agro (aquellos usuarios que podrán crear nuevas cuentas de empresas agrícolas) tanto como los diferentes usuarios que están asociados a la cuenta de una empresa agrícola (quienes serán los creadores, de los diferentes proyectos de cultivos según los permisos de cada uno). Para cada usuario se podrá ingresar nombre de usuario, contraseña, nombre, apellido, profesión, sexo y fecha de nacimiento.

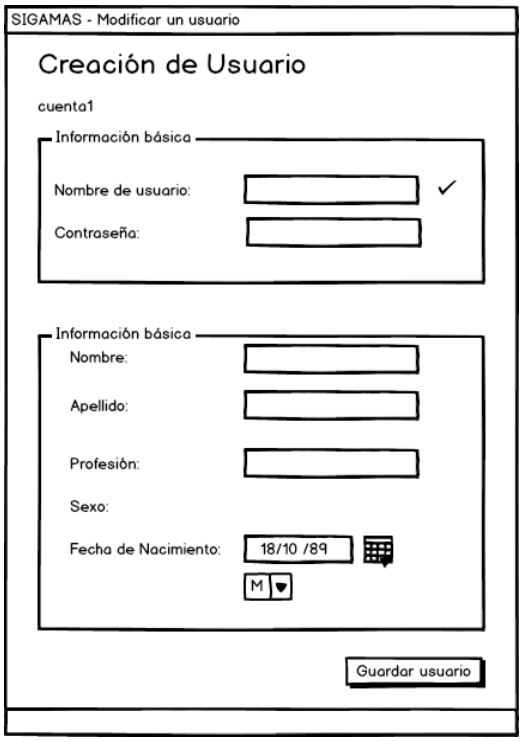
Prioridad: Alta

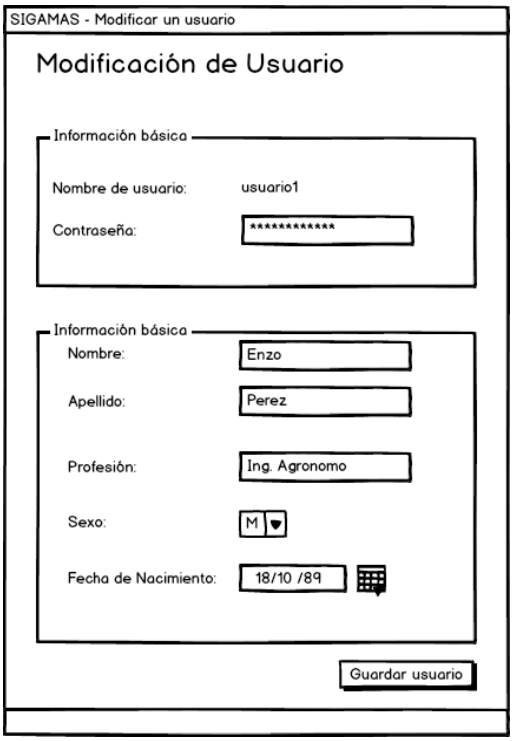
Clasificación: No crítico

Story Points: 13

Implementado: No implementado

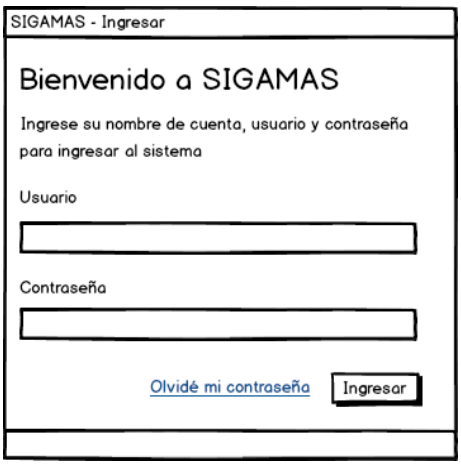
Especificación:

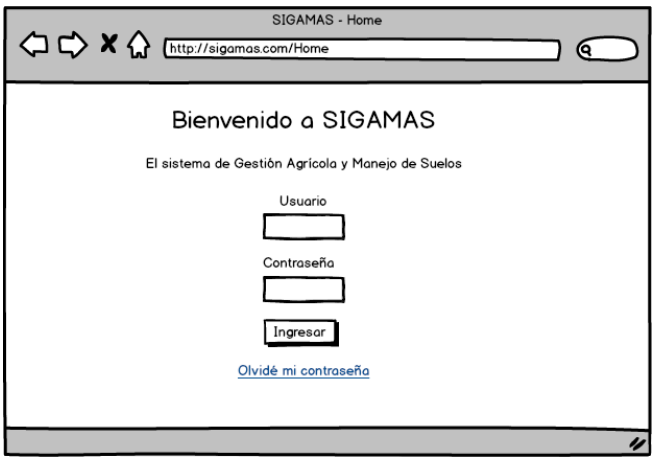




RF03

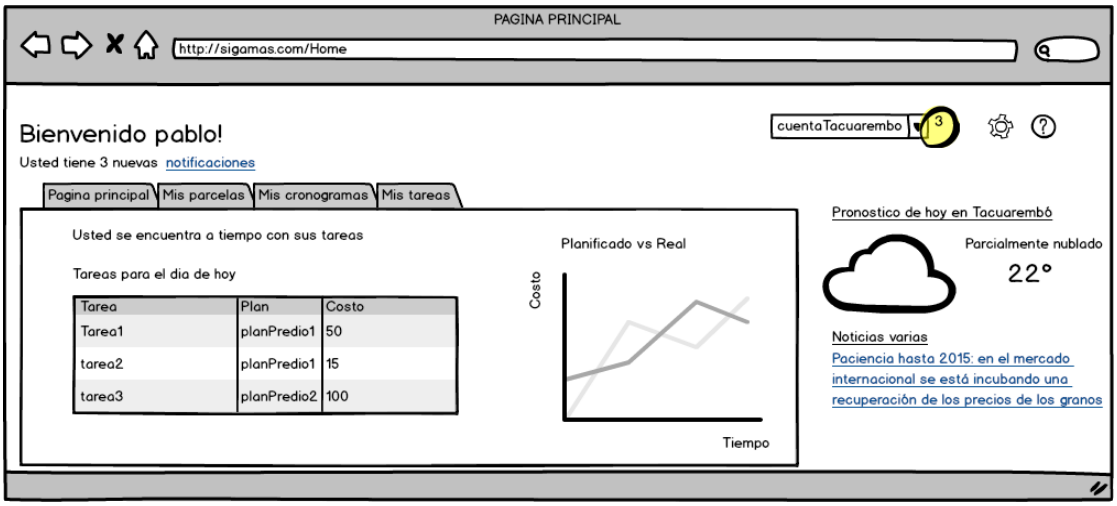
Nombre: *Login*
Actor: Usuario administrador, Usuario invitado
Descripción: El sistema permite el ingreso de los diferentes usuarios. Cuando un usuario ingresa al sistema tendrá permisos sobre los diferentes proyectos a los cuales está vinculado y podrá ver noticias y notificaciones personales. Para acceder al sistema cada usuario deberá ingresar nombre de usuario y contraseña.
Prioridad: Alta
Clasificación: No crítico
Story Points: 13
Implementado: Totalmente implementado
Especificación:

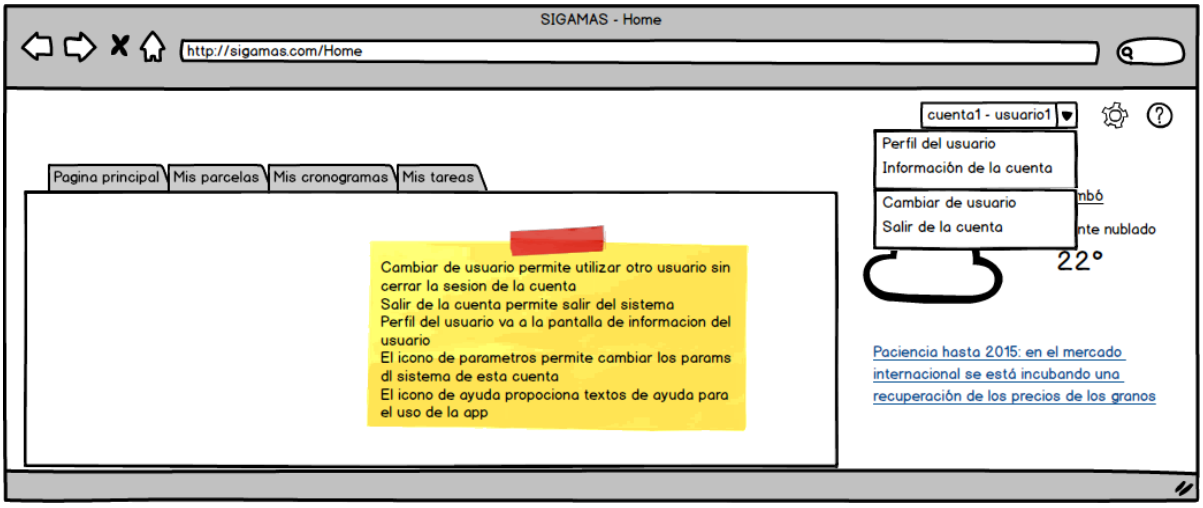




RF04

Nombre: *Logout*
Actor: Usuario administrador, Usuario invitado
Descripción: El sistema permite que los usuarios puedan cerrar su sesión. Al cerrar el navegador sin haber cerrado la sesión anteriormente el sistema cerrará la sesión automáticamente sin guardar los datos que se puedan estar ingresando en ese momento.
Prioridad: Alta
Clasificación: No crítico
Story Points: 5
Implementado: Totalmente implementado
Especificación:





RF05

Nombre: ABM Parcelas

Actor: Usuario administrador

Descripción: El sistema permite crear, modificar e ingresar parcelas asociadas a una cuenta de empresa agrícola. Dichas cuentas serán luego el predio donde se realicen los diferentes proyectos de cultivo. Para cada parcela se ingresará: nombre, padrón y departamento.

Prioridad: Media

Clasificación: No crítico

Story Points: 13

Implementado: Totalmente implementado

Especificación:

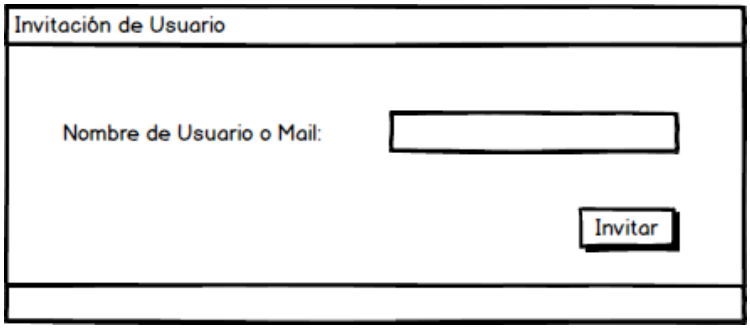
RF06

Nombre: Invitar a usuario por correo

Actor: Usuario administrador

Descripción: El sistema permite invitar por mail a otros usuarios a ver los proyectos vinculados del usuario emisor. Los usuarios “invitados” solo tendrán permiso de ver los proyectos pero no podrán realizar cambios sobre los mismos (siempre y cuando éste acepte la invitación realizada).

Prioridad: Media
Clasificación: No crítico
Story Points: 13
Implementado: Totalmente implementado
Especificación:

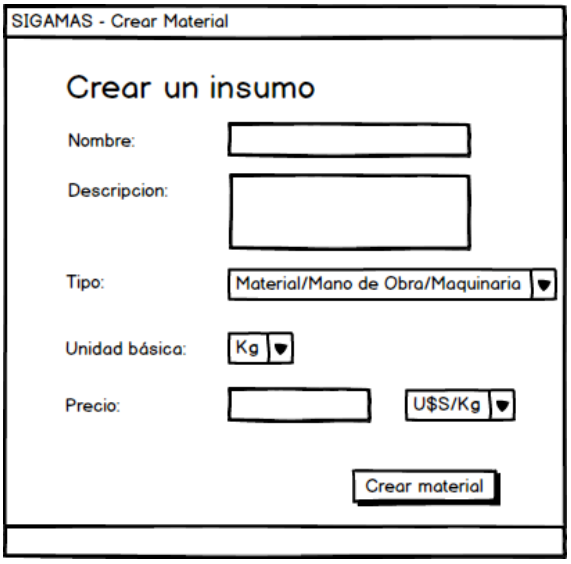


Módulo Planificación de cultivos

RF07

Nombre: ABM Insumos

Actor: Administrador del sistema
Descripción: El usuario puede crear, modificar o eliminar un insumo (dependiendo del tipo de usuario), ya sea Mano de Obra, Materiales o Maquinaria (equipos). Los insumos representan un costo en cada tarea y detallan la misma (explicitan los materiales, mano de obra y maquinaria que se necesitan para llevarla a cabo). Los datos que se podrán ingresar de cada insumo son: nombre, descripción, tipo, unidad básica y precio.
Prioridad: Alta
Clasificación: No crítico
Story Points: 5
Implementado: Planificado. Por ahora se ingresan por *Backoffice*.
Especificación:

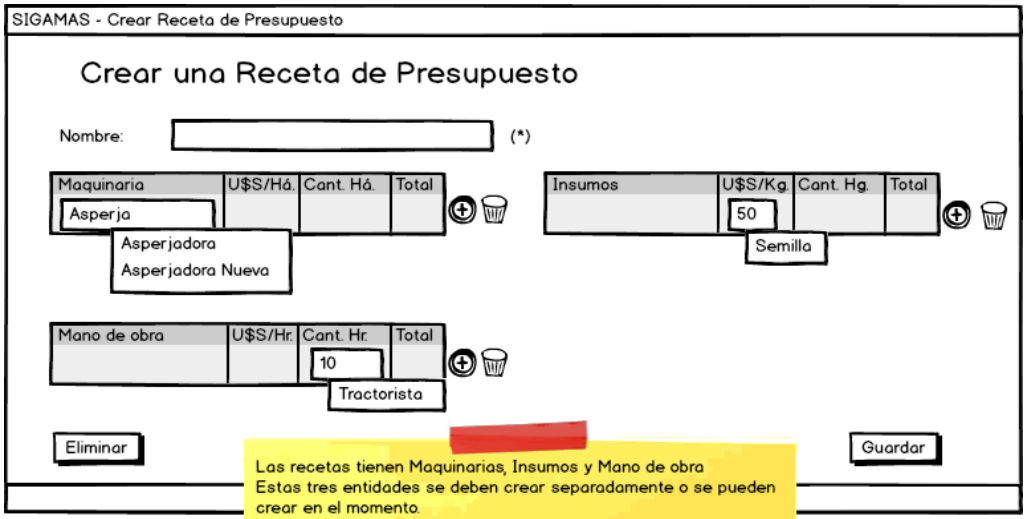


RF08

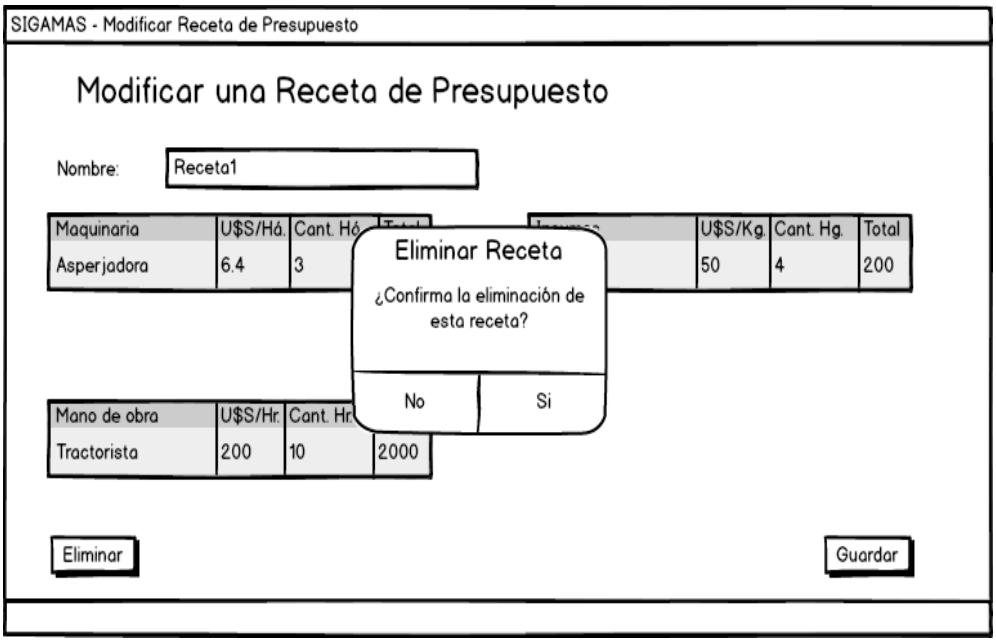
Nombre: Ver información de cultivo
Actor: Usuario administrador, Usuario invitado
Descripción: El sistema debe permitir que el usuario pueda visualizar toda la información relacionada con un cultivo. Esto incluye la fenología del mismo y las tareas relacionadas con cada etapa biológica, plagas y enfermedades.
Prioridad: Media
Clasificación: No crítico
Story Points: 5
Implementado: Totalmente implementado
Especificación:

RF09
Nombre: ABM Receta Estándar del Presupuesto
Actor: Administrador del sistema
Descripción: El usuario puede crear, modificar o eliminar una receta del presupuesto. Es decir, un administrador del sistema puede vincular insumos a cada una de las tareas. Puede seleccionar cualquier tipo de insumo, la cantidad que necesitará del mismo para realizar la tarea y el costo que su uso le significara. Esta receta estándar estará disponible para usarla en cualquier proyecto de cultivo de todas las cuentas si así se desease siendo igual para todos.
Prioridad: Media
Clasificación: No crítico
Story Points: 13
Implementado: Se implementará en un futuro

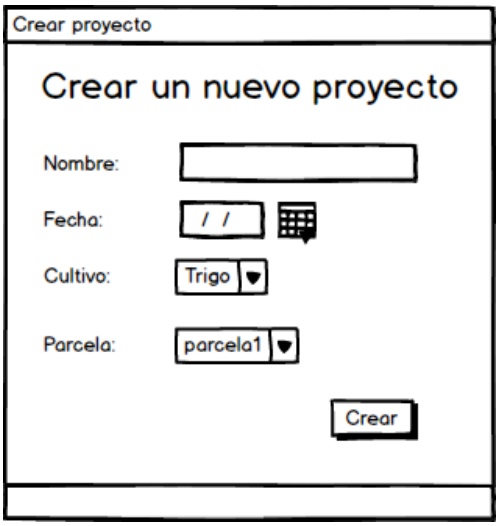
Especificación:



Las recetas tienen Maquinarias, Insumos y Mano de obra
Estas tres entidades se deben crear separadamente o se pueden crear en el momento.
Al principio las tablas estaran vacás con una fila libre. Se puede hacer click en cualquier celda y escribir texto.
Al escribir texto se buscará en la base de datos una coherencia y se mostrará una lista con los resultados.
Para la primera columna, se busca contenido parcial, mientras que para las dos siguientes contenido igual.
Se puede elegir de la lista o crear un nuevo material con los datos que se ingresan.
Al hacer click en "+" se agregará el seleccionado de la lista o se creará uno nuevo en caso de no seleccionar nada de la lista.
Al seleccionar una fila y hacer click en la papelera, se borrará ese material.
Todo esto se guardará temporalmente hasta que el usuario seleccione "Guardar" o todo se desctara si decide cerra la ventana.



RF10
Nombre: Iniciar proyecto de cultivo
Actor: Usuario administrador
Descripción: El usuario podrá seleccionar el cultivo que desea realizar e ingresar la información correspondiente a dicho cultivo: nombre del proyecto, cultivo, fecha de siembra y parcela. El sistema calcula el modelo fenológico que tendrá dicho cultivo y el seguimiento para el mismo (plan estándar de tareas y presupuesto).
Prioridad: Alta
Clasificación: Crítico
Story Points: 40
Implementado: Totalmente implementado
Especificación:

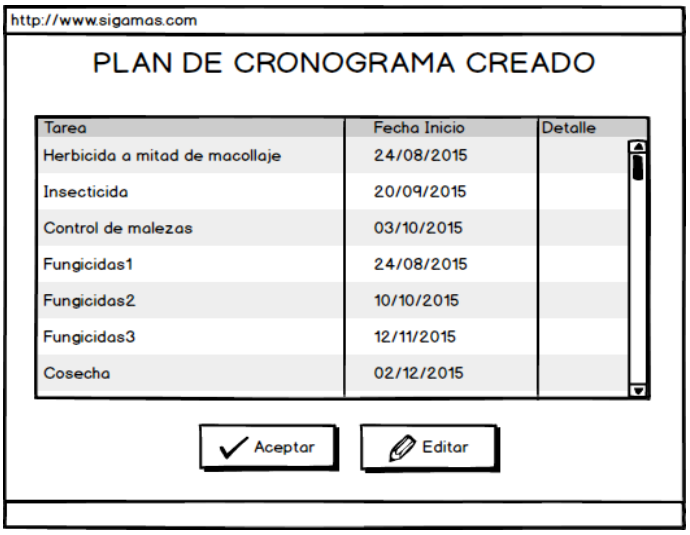


Criterios de aceptación:

Numero de escenario	Criterio de aceptación	Contexto	Evento	Resultado / Comportamiento esperado
1	Comprobar fecha de siembra	En caso que el usuario este agregando los datos correspondientes al proyecto del cultivo que desea realizar	Cuando el cliente ingresa la fecha correspondiente a la siembra	El sistema debe comprobar que la fecha ingresada sea posterior a la fecha actual. Y que este dentro de los meses aceptables para el cultivo seleccionado (mayo o junio si es cultivo de invierno, noviembre o diciembre si es cultivo de verano).
2	Comprobar localización del cultivo	En caso que el usuario este agregando los	Cuando el cliente selecciona la	El sistema debe comprobar que la ubicación de la parcela

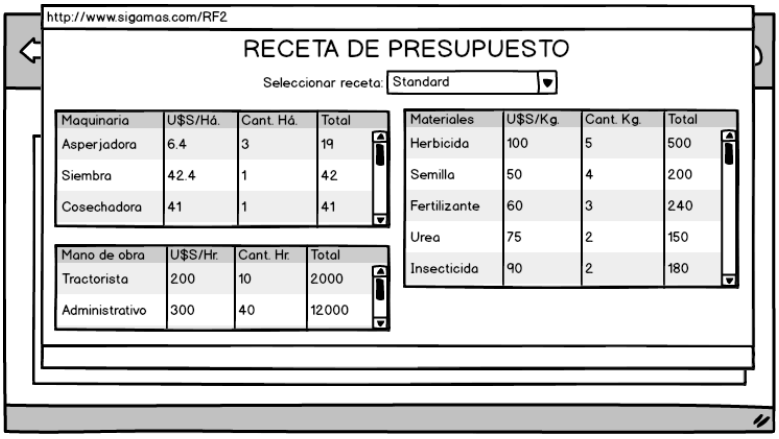
		datos correspondientes al proyecto del cultivo que desea realizar	localización de sus parcelas dónde llevará a cabo el cultivo	sea dentro de los límites de Uruguay.
3	Realizar modelo fenológico y seguimiento (plan de tareas)	En caso que el usuario haya realizado con éxito el ingreso de la información del proyecto del cultivo que desea realizar	Cuando el usuario selecciona "Guardar"	El sistema debe registrar en la base de datos todos los datos ingresados por el usuario. Además, según el cultivo especificado, el sistema debe calcular el modelo fenológico para el mismo y el seguimiento (receta estándar de tareas y de presupuesto).

RF11
Nombre: ABM Tareas estándares
Actor: Administrador del sistema
Descripción: El sistema puede crear, modificar o eliminar una tarea que forme parte del plan de la realización de un cultivo (seguimiento). Las tareas pueden estar relacionadas a un estado fenológico del cultivo en cuestión o a una fecha dentro del periodo que se realiza el cultivo (cantidad de días después de la siembra). Cada tarea es la realización de una receta, cuando se ingresa una tarea se selecciona la receta (si existe, de lo contrario se podrá crear), nombre, cuando debe realizarse, cantidad de días, descripción y observaciones.
Prioridad: Media
Clasificación: No crítico
Story Points: 5
Implementado: Si, en vez de seleccionar una receta de presupuesta solo se asumirá un costo fijo para la tarea, cuando se implemente la receta se podrá elegir una y tener los costos detallados.
Especificación:



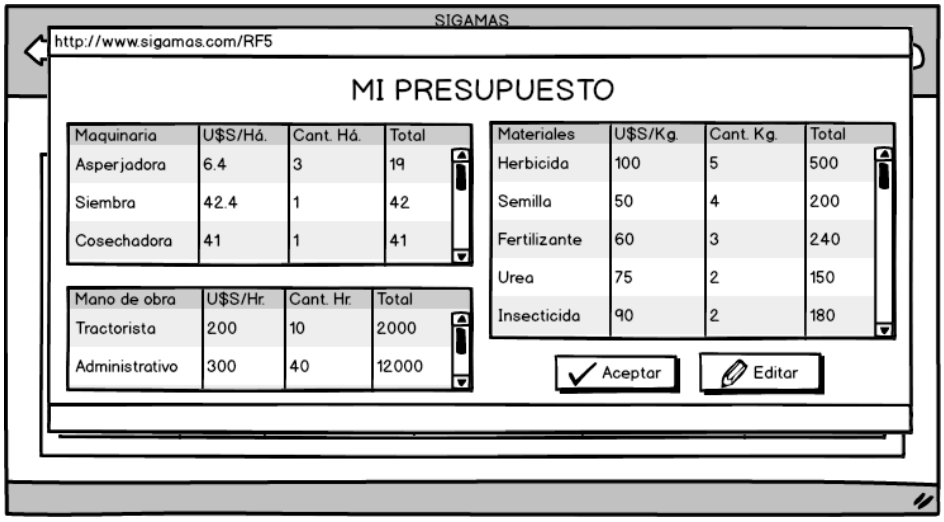
RF12

Nombre: Elegir receta de presupuesto
Actor: Usuario administrador
Descripción: El sistema permite seleccionar la receta de presupuesto estándar generada para el cultivo que se está realizando el proyecto u otra usada anteriormente (una receta personalizada por algún usuario perteneciente a la misma cuenta, que la haya creado para realizar el mismo cultivo). El sistema le despliega al usuario las recetas asociadas al tipo de cultivo seleccionado anteriormente permitiéndole seleccionar una de ellas para el nuevo proyecto de cultivo.
Prioridad: Media
Clasificación: No crítico
Story Points: 13
Implementado: Planificada para un futuro
Especificación:



RF13
Nombre: Realizar planificación de presupuesto
Actor: Usuario administrador
Descripción: El usuario puede modificar la receta de presupuesto que seleccionó para el cultivo durante la planificación del proyecto. Puede modificar todos los datos relacionados con los insumos así como la eliminación de los mismos. Si la modificación se realiza durante la planificación el sistema únicamente actualizará los datos. De lo contrario, si la modificación se realizase cuando el proyecto ya está en ejecución, el sistema debe guardar las diferentes versiones para registrar los desvíos con el plan inicial.
Prioridad: Baja
Clasificación: Crítico
Story Points: 13
Implementado: Planificada para un futuro

Especificación:



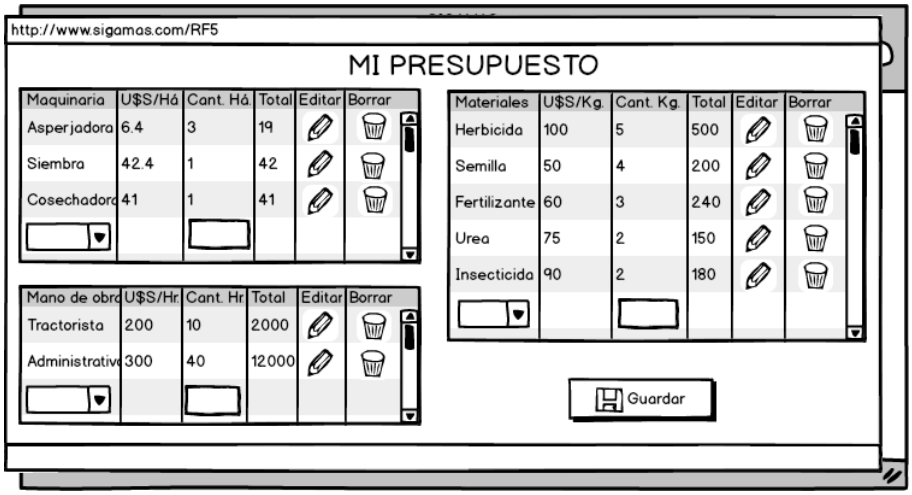
Criterios de aceptación:

Numero de escenario	Criterio de aceptación	Contexto	Evento	Resultado / Comportamiento esperado
1	El usuario modifica la receta durante la planificación	En caso que el usuario desee modificar una de las recetas ingresadas en el sistema para personalizar su planificación (antes de la fecha de siembra)	Cuando el usuario modifica la receta de presupuesto que selecciono para llevar a cabo su proyecto de cultivo antes de que inicie la ejecución del mismo.	El sistema debe permitir modificar todos los datos, esto es: agregar nuevos insumos, modificar los datos de los insumos ya existentes e incluso eliminarlo. Además debe guardar en la base de datos todos los cambios realizados.

RF14
Nombre: Realizar modificaciones de la planificación de presupuesto
Actor: Usuario administrador
Descripción: Luego de seleccionada y modificada una receta para adaptarla al plan del nuevo proyecto de cultivo, el usuario puede modificar la receta del inicial, pero mantener las diferentes versiones que ha ido construyendo. Si el usuario realiza cambios de la planificación durante la ejecución del proyecto, el sistema irá creando nuevas versiones de la

misma planificación para llevar registros del desvío de los diferentes planes a lo largo de todo el proyecto.
Prioridad: Baja

Clasificación: Crítico
Story Points: 13
Implementado: A futuro
Especificación:



Criterios de aceptación:

Numero de escenario	Criterio de aceptación	Contexto	Evento	Resultado / Comportamiento esperado
1	El usuario modifica el plan de presupuesto durante la ejecución del proyecto del cultivo	En caso que el usuario desee ir modificando durante la ejecución del proyecto de cultivo la planificación que realizó para ajustar según la realidad.	Cuando el usuario modifica el plan de presupuesto que seleccionó (y eventualmente puede haberlo modificado) durante la ejecución del proyecto del cultivo (después de la fecha de siembra)	El sistema debe permitir modificar todos los datos de un plan de presupuesto de aquellas tareas que todavía no se realizaron. Los cambios realizados los tiene que guardar en la base de datos como una nueva versión y registrar cual es el plan actual del proyecto sin eliminar los anteriores.

RF15

Nombre: Elegir cronograma (plan de tareas)

Actor: Usuario administrador

Descripción: El sistema permite seleccionar una planificación de tareas estándar generada para el cultivo u otra guardada anteriormente (un cronograma personalizado por algún usuario perteneciente a la misma cuenta, que lo haya realizado para llevar a cabo el mismo cultivo). El sistema le despliega al usuario los cronogramas asociados al tipo de cultivo correspondiente al proyecto y le permite seleccionar uno para realizar el nuevo proyecto.

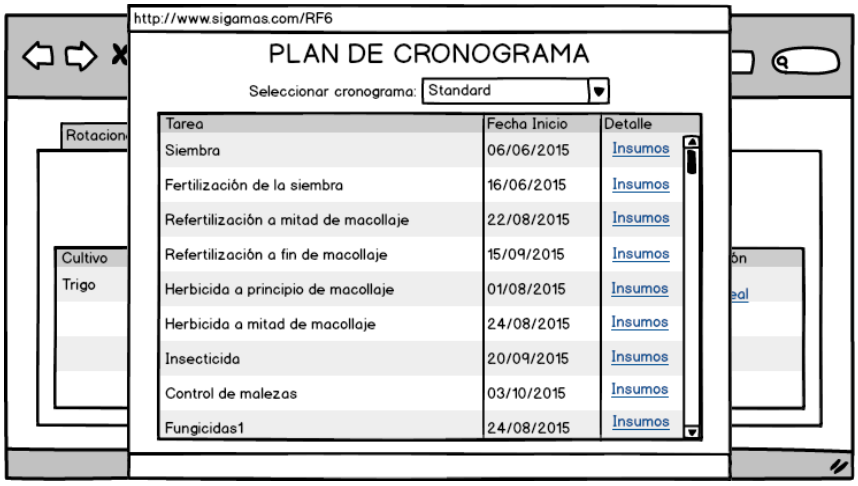
Prioridad: Alta

Clasificación: No crítico

Story Points: 13

Implementado: No en su totalidad. Actualmente se elige el cronograma por defecto. A futuro se podrá elegir un cronograma personalizado

Especificación:



RF16

Nombre: Realizar planificación de cronograma

Actor: Usuario administrador

Descripción: El usuario puede modificar el cronograma que seleccionó para personalizarlo según sus conocimientos y experiencias. Puede editar tareas. Al guardar, el sistema actualiza los datos de la planificación del cronograma para el proyecto del cultivo en cuestión. Si la modificación se realizase cuando el proyecto ya está en ejecución, el sistema debe guardar las diferentes versiones para registrar los desvíos con el plan inicial.

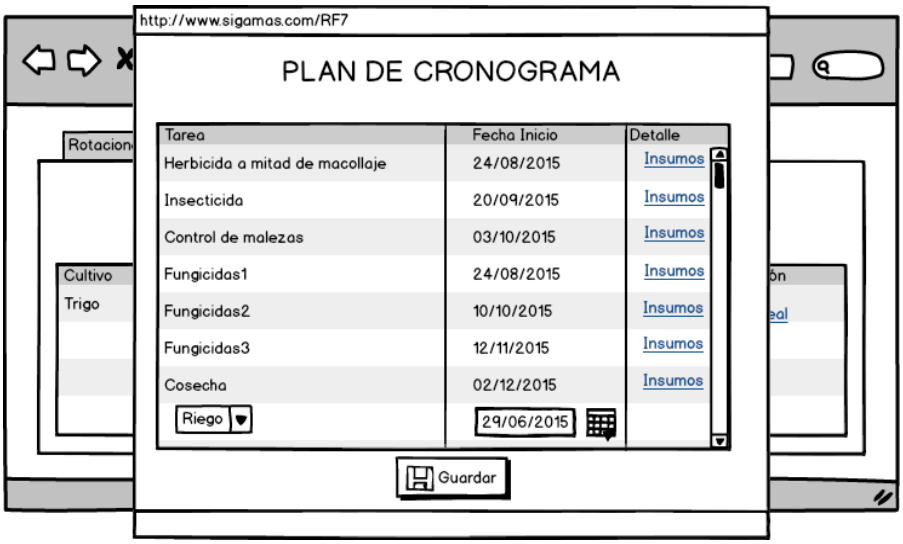
Prioridad: Media

Clasificación: Crítico

Story Points: 13

Implementado: Si

Especificación:

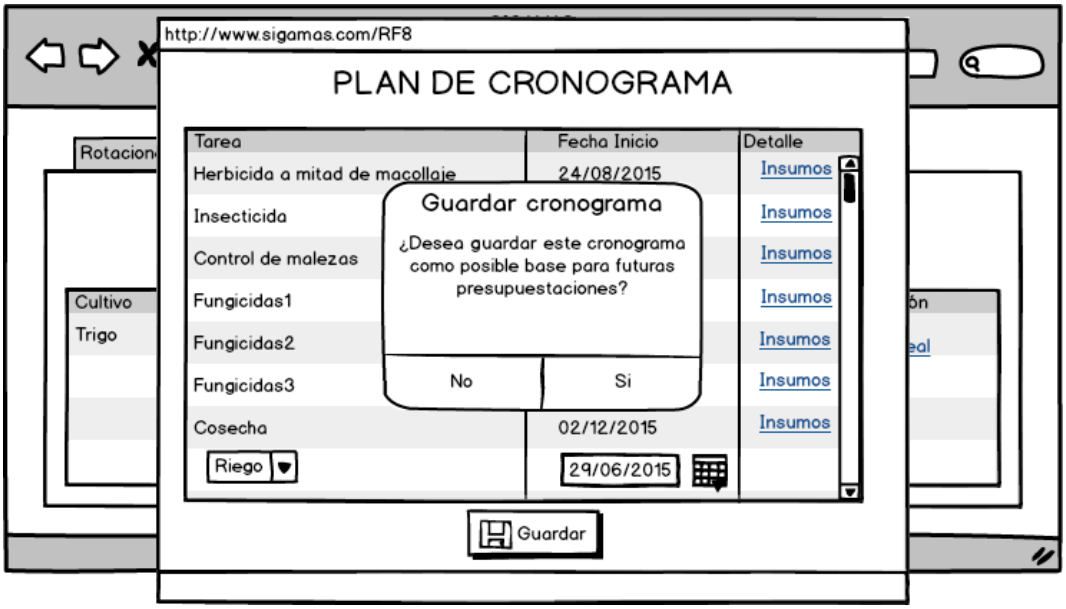


Criterios de aceptación:

Numero de escenario	Criterio de aceptación	Contexto	Evento	Resultado / Comportamiento esperado
1	El usuario modifica el cronograma durante la planificación	En caso que el usuario desee modificar el cronograma que selecciono para personalizar su planificación	Cuando el usuario modifica el cronograma que selecciono para llevar a cabo su proyecto de cultivo antes que inicie la ejecución del mismo.	El sistema deber permitir modificar todos los datos, esto es: agregar tareas nuevas, modificar información de tareas ya existes o eliminar tareas registradas en el cronograma. Además debe de actualizar en la base de datos todos los cambios realizados.

RF17
Nombre: Realizar modificaciones de la planificación del cronograma
Actor: Usuario administrador
Descripción: Luego de seleccionado y modificado el cronograma para adaptarlo al plan del nuevo proyecto de cultivo, el usuario puede modificar el cronograma inicial, pero mantener las diferentes versiones que ha ido construyendo. Si el usuario realiza cambios de la planificación durante la ejecución del proyecto irá creando nuevas versiones de la misma planificación para llevar registros del desvío de los diferentes planes a lo largo de todo el proyecto.
Prioridad: Alta
Clasificación: Crítico

Implementado: Si
Story Points: 40
Especificación:

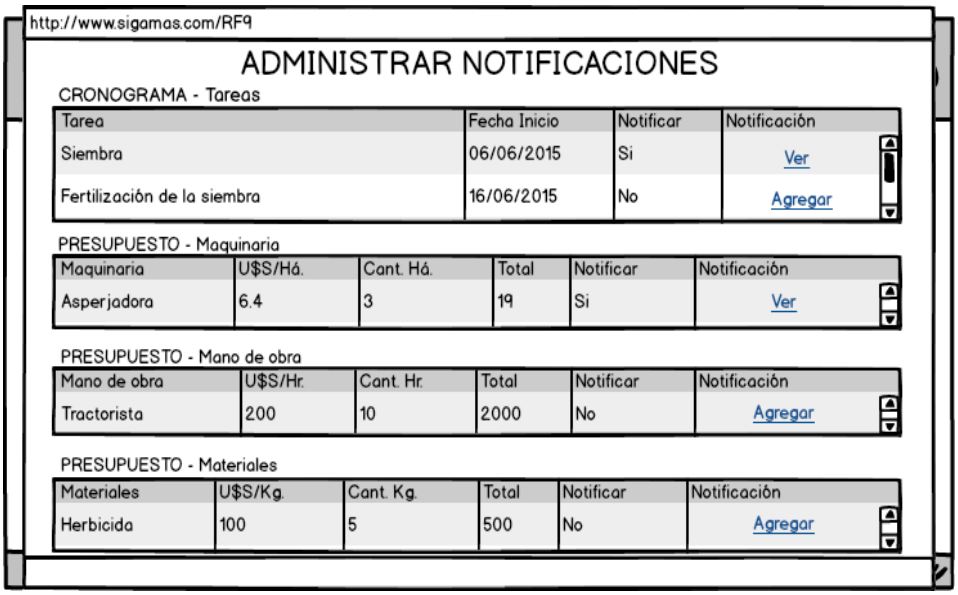


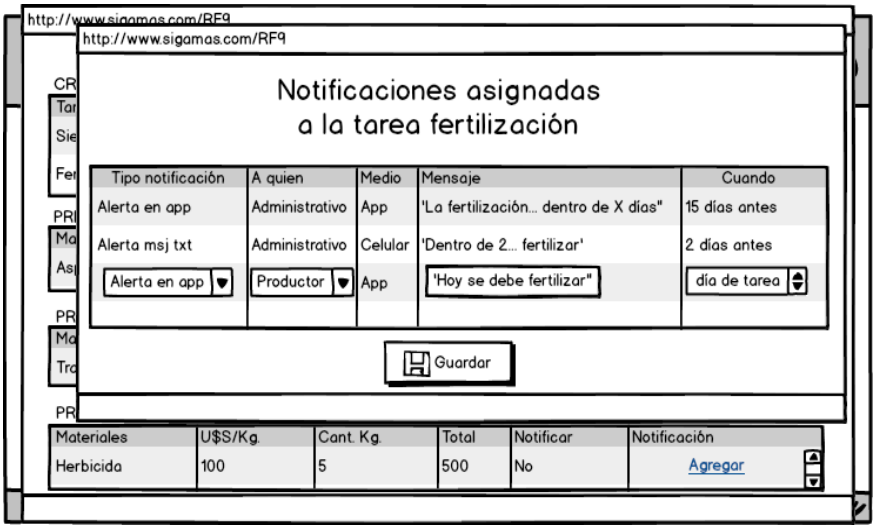
Criterios de aceptación:

Numero de escenario	Criterio de aceptación	Contexto	Evento	Resultado / Comportamiento esperado
1	El usuario modifica el plan de tareas durante la ejecución del proyecto del cultivo	En caso que el usuario desee ir modificando durante la ejecución del proyecto de cultivo la planificación que realizó para ajustar según la realidad.	Cuando el usuario modifica el plan de tareas que seleccionó anteriormente (y eventualmente modificó) durante la ejecución del proyecto del cultivo (después de la fecha de siembra)	El sistema debe permitir modificar todos los datos de un plan de tareas, tanto aquellas relacionadas con el estado fenológico del cultivo como aquellas que no lo están, siempre y cuando la fecha de las mismas sean posterior a le fecha actual. Los cambios realizados los tiene que guardar en la base de datos como una nueva versión e indicar cuál es el plan actual del proyecto sin eliminar los anteriores.

RF18

Nombre: Administrar notificaciones
Actor: Usuario administrador
Descripción: El usuario selecciona las tareas que desea ser notificado de la proximidad en el tiempo de la misma. Puede modificar la anterioridad con que desea ser informado, a través de qué medio serlo y que mensaje recibir.
Prioridad: Alta
Clasificación: No crítico
Story Points: 13
Implementado: Totalmente implementado
Especificación:





RF19

Nombre: Guardar planificaciones personales del cronograma

Actor: Usuario administrador

Descripción: Al finalizar un proyecto, el usuario puede guardar la última versión de la planificación del cronograma realizado a lo largo del proyecto de un cultivo para poder usarlo en futuras gestiones para el mismo cultivo.

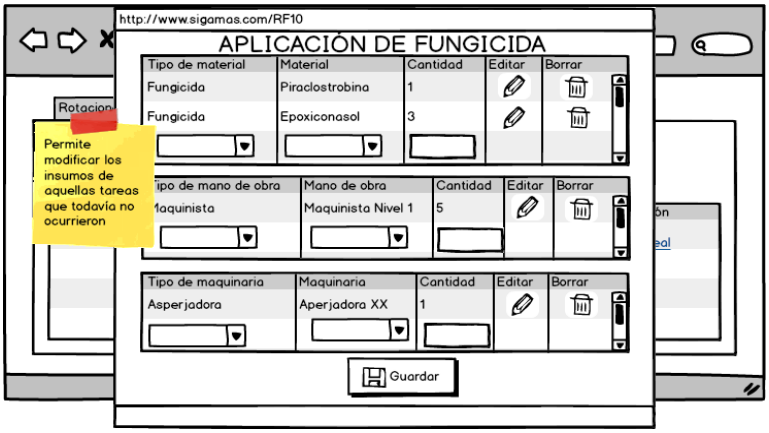
Prioridad: Baja

Clasificación: No crítico

Implementado: A futuro

Story Points: 13

Especificación:



RF20

Nombre: Guardar planificaciones personales de la receta de presupuesto

Actor: Usuario administrador

Descripción: Al finalizar un proyecto, el usuario puede guardar la última versión de la planificación de una receta realizada en un proyecto de un cultivo para poder usarlo en futuras presupuestaciones del mismo cultivo.

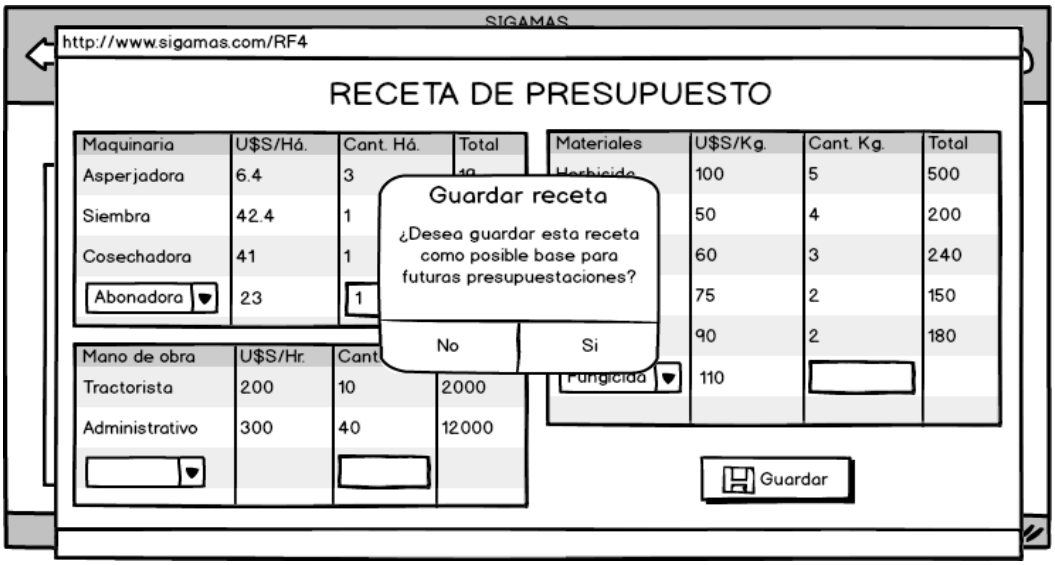
Prioridad: Baja

Clasificación: No crítico

Implementado: A futuro

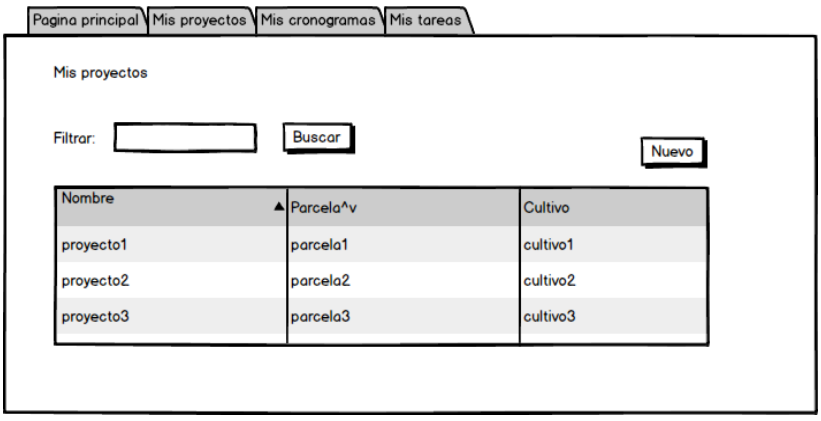
Story Points: 8

Especificación:



Modulo Ejecución de un cultivo

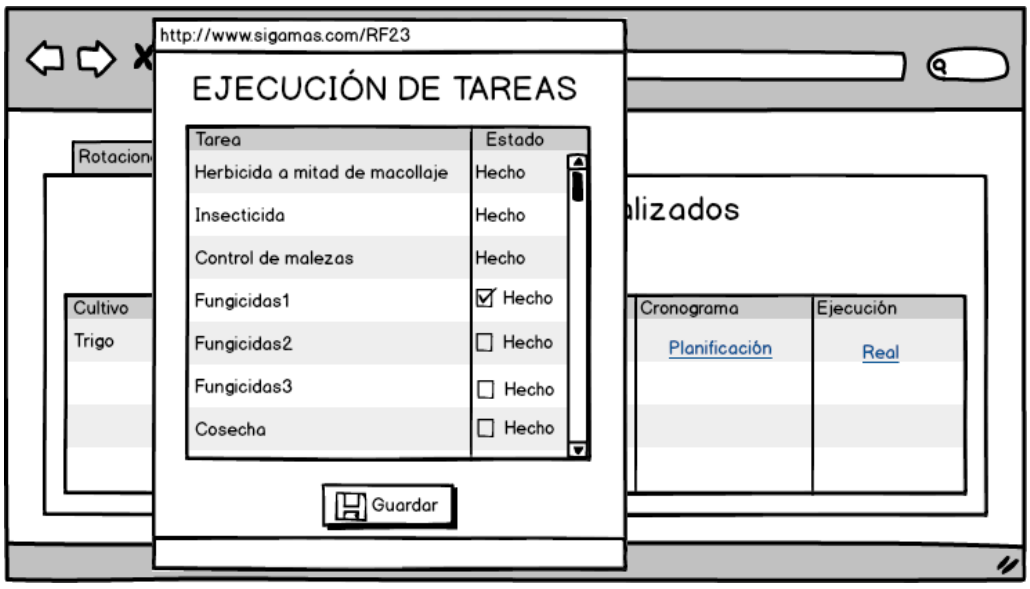
RF21
Nombre: Ver proyectos de cultivos
Actor: Usuario administrador, Usuario invitado
Descripción: El sistema debe permitir que los usuarios puedan ver los proyectos de cultivos a los cuales están asociados. Los administradores de una cuenta tendrán permiso para modificar todo lo que deseen modificar del proyecto, lo usuarios invitados (por los administradores a través del correo) solo podrán ver la información del proyecto sin interferir en el mismo.
Prioridad: Alta
Clasificación: Crítico
Implementado: Si
Story Points: 40
Especificación:



Criterios de aceptación:

Numero de escenario	Criterio de aceptación	Contexto	Evento	Resultado / Comportamiento esperado
1	El usuario debe de poder ver los proyectos de cultivos a los cuales está asociado	En caso que el usuario desee ver los proyectos de cultivo a los que está asociado	Cuando el usuario selecciona "Mis proyectos"	El sistema deberá mostrar en pantalla todos los proyectos de cultivos a los cuales está asociado dicho usuario.

RF22
Nombre: Ejecutar tareas
Actor: Usuario administrador
Descripción: A medida que se va realizando el proyecto, el usuario puede ir guardando en el sistema la ejecución de este. Puede ir ingresando todas las tareas realizadas, las fechas que se hicieron y todos los insumos que insumió su realización. Además puede ir ingresando el cronograma fenológico real, es decir las fechas en que el cultivo se encontró en cada uno de sus estados fenológicos. El sistema debe desplegar las diferentes tareas planificadas que aún no se han ejecutado permitiendo elegir una de ellas y hacerles las modificaciones convenientes para adaptarlas a la realidad. También debe permitir ingresar tareas nuevas que no estaban planificadas desde un inicio.
Prioridad: Alta
Clasificación: Crítico
Implementado: Totalmente implementado
Story Points: 40
Especificación:



Criterios de aceptación:

Numero de escenario	Criterio de aceptación	Contexto	Evento	Resultado / Comportamiento esperado
1	El usuario ingresa una tarea que estaba programada en otra fecha	Se va realizando el proyecto	Durante la ejecución del proyecto	El sistema despliega las diferentes tareas planificadas que aún no se han ejecutado permitiendo elegir una de ellas y hacerles las modificaciones convenientes para adaptarlas a la realidad. También debe permitir ingresar tareas nuevas que no estaban planificadas desde un inicio.
2	El usuario modifica o elimina el uso de un insumo en alguna tarea			
3	El usuario agrega tareas nuevas que no estaban planificadas			
4	El usuario agrega el uso de insumos que no estaban planificados			
5	El usuario ingresa un estado fenológico en una fecha que no era el previsto			

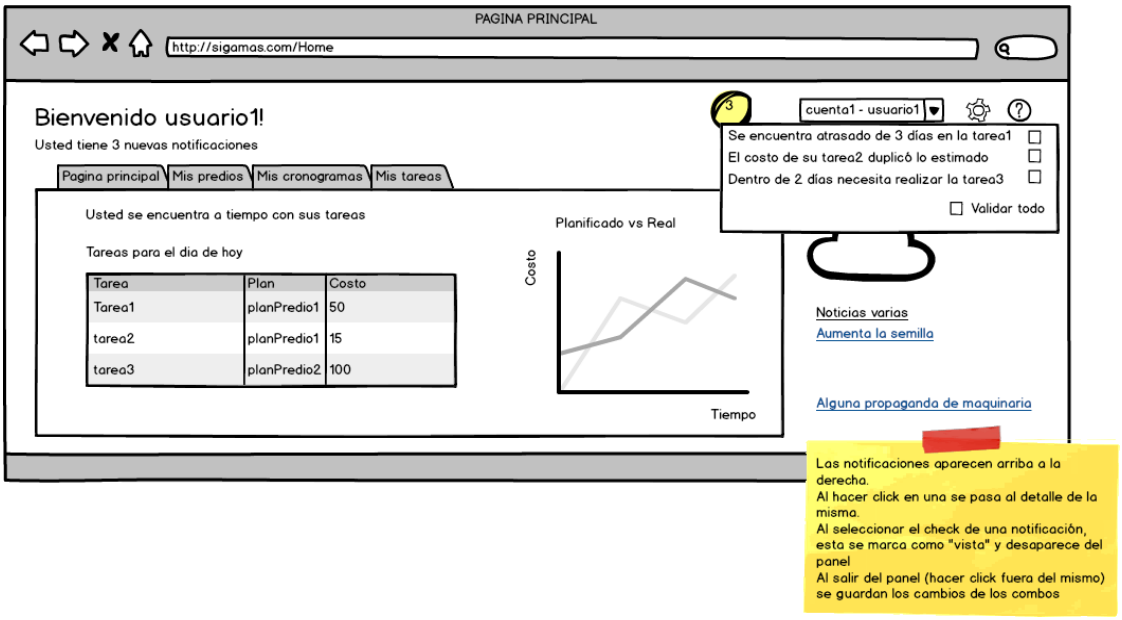
F23

Nombre: Reajustar plan
Actor: Sistema
Descripción: De acuerdo al plan establecido y los desvíos ocasionados por el clima que no se ajusta al “ideal”, el sistema debe ajustar y notificar los usuarios los cambios necesarios tanto en cronograma como en insumos para lograr un mejor rendimiento del cultivo.
Prioridad: Media
Clasificación: Crítico
Story Points: 40
Implementado: Totalmente implementado
Especificación: No corresponde
Criterios de aceptación:

Numero de escenario	Criterio de aceptación	Contexto	Evento	Resultado / Comportamiento esperado
1	Reajustar plan y notificar a usuarios	Un plan creado no se ajusta al "ideal"	Deben haber desvíos ocasionados, por ejemplo cuestiones climáticas	El sistema ajusta y notifica a los usuarios los cambios necesarios tanto en cronograma como en insumos para lograr un mejor rendimiento del cultivo.

RF24
Nombre: Notificar hitos del cronograma
Actor: Sistema
Descripción: A medida que se acercan las fechas establecidas de las notificaciones deseadas y planificados por el usuario, el sistema debe mostrar a los usuarios los mensajes correspondientes según lo ingresado en la administración de notificaciones.
Prioridad: Media
Clasificación: Crítico
Story Points: 13
Implementado: Totalmente implementado

Especificación:



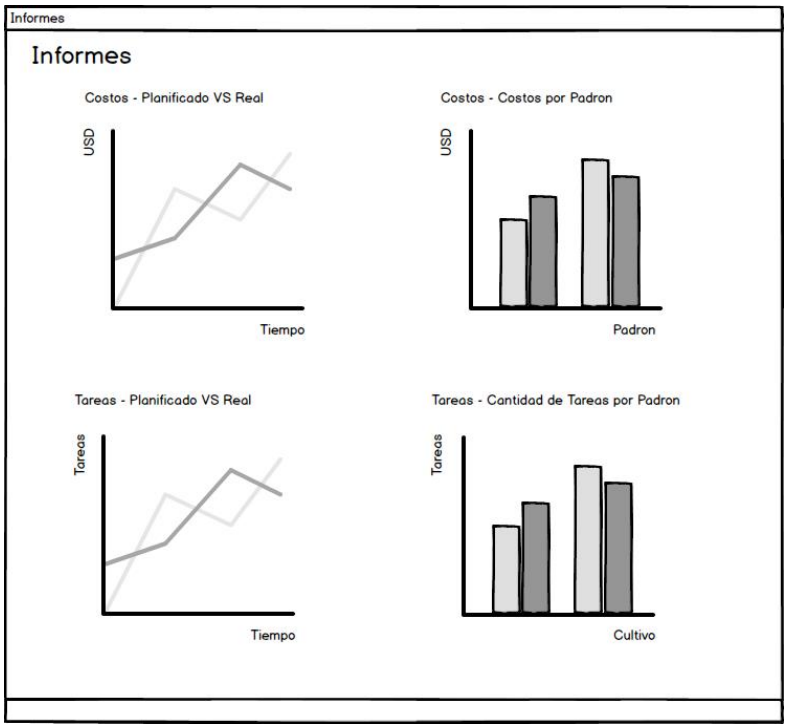
Criterios de aceptación:

Numero de escenario	Criterio de aceptación	Contexto	Evento	Resultado / Comportamiento esperado
1	Se notifican hitos del cronograma	Se ejecuta un plan creado	Se aproximan fechas establecidas para las notificaciones y planificaciones	El sistema muestra a los usuarios los mensajes correspondientes según lo ingresado en la administración de notificaciones.

Módulo Información

RF25
Nombre: Informes
Actor: Usuario administrador
Descripción: El usuario podrá visualizar distintos informes de rendimientos, desvíos, costos, etc.
Prioridad: Media
Clasificación: No crítico
Implementado: Totalmente implementado
Story Points: 13

Especificación:



RF26
Nombre: Ver noticias
Actor: Usuario administrador, Usuario invitado
Descripción: El sistema deberá permitir que los usuarios vean noticias relacionadas con la agricultura, es decir mostrará información relevante para la toma de decisiones de los usuarios de todas aquellas páginas de interés general en el sector agrícola del Uruguay.
Prioridad: Media
Clasificación: No crítico
Implementado: Totalmente implementado
Story Points: 5
Especificación: Igual al *Login* con página inicial

Requerimientos no funcionales

Los requerimientos no funcionales son características que especifican criterios que pueden usarse para juzgar la operación de un sistema en lugar de sus comportamientos específico, en contraposición a los requerimientos funcionales.
Para mantener un criterio de orden, se detallan a continuación los requerimientos no funcionales identificados agrupados según qué atributo de calidad satisfacen.

ATRIBUTOS DE CALIDAD

Arquitectura

RNF01

Nombre: Modelo cliente-servidor

Descripción: El modelo base de funcionamiento del sistema debe ser Cliente-Servidor, para n clientes.

Criterio de aceptación: La arquitectura debe seguir el patrón de arquitectura Cliente-Servidor.

RNF02

Nombre: Cliente Aplicación Web

Descripción: Los clientes deben poder acceder al sistema mediante la web.

Criterio de aceptación: Se debe implementar una aplicación web para la operativa de los clientes.

Seguridad

RNF03

Nombre: Sesiones

Descripción: Las sesiones inactivas tienen una duración determinada finita.

Criterio de aceptación: Las sesiones se deben mantener por un lapso máximo de 30 minutos en medida que la demande de u saurios no sea tal que amerite bajar este tiempo.

RNF04

Nombre: Encriptación de la información

Criterio de aceptación: Los datos sensibles serán encriptados, así como toda aquella referencia que permita relacionar la información propietaria de los usuarios con estos de forma de no comprometer los datos pertenecientes a dichos usuarios.

Auditoria

RNF05

Nombre: *Logs* del sistema

Criterio de aceptación: Se guardarán trazas de log por cada entrada y salida del sistema, así como por cada operación crítica del sistema que pueda tener impacto significativo desde el punto de vista de negocio. Las trazas de log deben contener al menos:

- Fecha
- Hora
- Módulo
- Operativa (función, método)
- Usuario
- Descripción del evento (*info, debug, error, warn*)

Performance

RNF06

Nombre: Tiempos de respuesta

Criterio de aceptación: Se deberá mantener un desarrollo fluido de las operativas del sistema de manera responsiva, para lo cual se considera:

- El sistema debe estar disponible para operar inmediatamente después de que se carga la página inicial.
- En el caso de operaciones que insumen mucho tiempo se deberá notificar debidamente al usuario mediante algún tipo de diálogo de espera.
- En el caso de operaciones que se puedan paralelizar, están deberán ser realizadas en *Background* y luego se deberá notificar al usuario.
- La carga de la página no deberá tomar más de 5 segundos. En este tiempo se tiene que dibujar la página con sus elementos y si fuera el caso de que se debieran cargar datos, estos deberían ser cargados mediante solicitudes asincrónicas (Ajax por ejemplo).

RNF07
Nombre: Tiempos de procesamiento
Criterio de aceptación:

- Ninguna operativa puede superar 5 de tiempo de procesamiento
- El tiempo promedio de respuesta debe estar en los 30 segundos

RNF08
Nombre: Tiempos de ejecución de consultas
Criterio de aceptación: El sistema debe responder en menos de 10 segundos, si es necesario con una respuesta parcial

RNF09
Nombre: Tiempos de ejecución de reportes
Criterio de aceptación: Cualquier reporte que deba tener un tiempo de procesamiento mayor a los 10 segundos deberá ser procesado en *Background* y se dispondrá al cliente en una bandeja específica y se notificará a éste debidamente. En caso de que el reporte se pueda resolver en menos de 10 segundos se dispondrá a demanda al cliente.

- ***0.1 second*** *is about the limit for having the user feel that the system is* ***reacting instantaneously****, meaning that no special feedback is necessary except to display the result.[2]*
- ***1.0 second*** *is about the limit for the* ***user's flow of thought*** *to stay uninterrupted, even though the user will notice the delay. Normally, no special feedback is necessary during delays of more than 0.1 but less than 1.0 second, but the user does lose the feeling of operating directly on the data.[2]*
- ***10 seconds*** *is about the limit for* ***keeping the user's attention*** *focused on the dialogue. For longer delays, users will want to perform other tasks while waiting for the computer to finish, so they should be given feedback indicating when the computer expects to be done. Feedback during the delay is especially important if the response time is likely to be highly variable, since users will then not know what to expect.[2]*

Capacidad

RNF10

Nombre: Capacidad de procesamiento
Criterio de aceptación:

- Transacciones posibles: El sistema deberá responder tantas solicitudes como usuarios activos existan en él al mismo tiempo
- Sesiones activas al mismo tiempo: El sistema deberá soportar tantos usuarios activos como usuarios existan en el sistema.

RNF11
Nombre: Almacenamiento
Criterio de aceptación: El sistema debe disponer del espacio suficiente para guardar todos los datos que se generan en su operativa normal.

Escalabilidad

RNF12
Nombre: Previsión de crecimiento de base de datos
Descripción: Se prevé que una vez que el usuario adopte la herramienta, su uso sea cada vez más frecuente, por lo que se tiene que considerar la cantidad creciente de datos guardados en base de datos.
Criterio de aceptación: Se prevé asignar más recursos físicos para el almacenamiento de datos.

Disponibilidad

RNF13
Nombre: Horarios de operatividad
Criterio de aceptación: El sistema deberá estar disponible 24/7
RNF14
Nombre: Requerimientos de conexión
Criterio de aceptación: Para acceder al sistema se deberá disponer de una conexión a internet con una velocidad media del mercado.

Confiabilidad

RNF15
Nombre: Tiempo máximo caído
Criterio de aceptación: El sistema tiene que estar disponible el 95% del tiempo

RNF16
Nombre: Tiempo máximo de recuperación
Criterio de aceptación: El sistema debe ser capaz de recuperarse inmediatamente después de una falla (se tendrán que manejar las excepciones y tener *Backups* para poder *Retry* o hacer *Rollback*)

RNF17
Nombre: Técnicas para confiabilidad de los datos
Criterio de aceptación: Detección de fallas mediante *Exception Handling* (Manejo de excepciones)

Integridad

RNF18
Nombre: Manejo de errores de Entradas/Salida
Criterio de aceptación: El sistema deberá manejar los errores de Entrada/Salida de modo de que no generen datos incorrectos mediante validaciones.

RNF19
Nombre: Datos con fallas
Criterio de aceptación: El sistema deberá descartar datos corruptos o con fallas para no guardar esa información.

Recuperación

RNF20
Nombre: Tiempo de recuperación
Criterio de aceptación: El tiempo de recuperación del sistema no puede exceder los 10 minutos.

RNF21
Nombre: Frecuencia de *Backups*
Criterio de aceptación: Se realizará un *Backup* diario de la base de datos.

RNF22
Nombre: Táctica de *Backups*
Criterio de aceptación: *Online* (datos y *Logs*).

Interoperabilidad

RNF23
Nombre: Intercomunicación con otras aplicaciones
Criterio de aceptación: El sistema deberá poder consumir Web *Services* que ofrezcan diferentes servicios.

Compatibilidad

RNF24
Nombre: Compatibilidad con distintos sistemas operativos
Criterio de aceptación: El sistema será compatible con todos los sistemas operativos que puedan ejecutar los browsers compatibles, esto queda determinado por los requerimientos no funcionales RNF29 y RNF02.

RNF25
Nombre: Compatibilidad con distintos *Browser*
Criterio de aceptación: El sistema deberá funcionar en todos los *browsers*. Se deberá implementar código estándar (que funcione en todos los navegadores), y no utilizar *tags* HTML específicos a un *browser*. Validar las páginas (HTML y CSS) mediante WC3 *validators*.

RNF26
Nombre: Compatibilidad con distintas plataformas de hardware
Criterio de aceptación: El sistema será compatible con las distintas plataformas de Hardware, esto queda determinado por los requerimientos no funcionales RNF29 y RNF02.

Mantenibilidad y modificabilidad

RNF27
Nombre: Estándares de arquitectura que la permiten
Criterio de aceptación: La arquitectura del sistema está definida en el documento de arquitectura.

RNF28
Nombre: Estándares de diseño
Criterio de aceptación: El diseño del sistema se realizará mediante un estándar de diseño.

RNF29
Nombre: Estándares de codificación
Criterio de aceptación: El código debe de seguir el estándar de codificación especificado.

Usabilidad

RNF30
Nombre: *Look and Feel*
Criterio de aceptación:

- Cantidad de elementos por pantalla: la cantidad de elementos por pantalla no deberá ser mayor a 15.
- *Layout* de los elementos: los elementos en la pantalla deberán seguir un mismo criterio de disposición para generar una interfaz uniforme
- Flujo de pantallas: se debe considerar un mecanismo de flujo entre pantallas uniforme, con una navegación clara, considerando:
    - En todo momento el usuario debe saber dónde está
    - En todo momento el usuario debe tener claro de dónde viene y que opciones tiene.
- Colores: se deberá respetar una paleta de colores predefinida.
- Accesos rápidos de teclado: se deben considerar accesos rápidos de teclado que permitan agilizar operaciones comunes
- Auto-aprendizaje: se debe favorecer el auto-aprendizaje de la herramienta mediante sugerencias e indicaciones claras, de forma de que el usuario tenga la sensación de dominio temprano del sistema.

Documentación

RNF31
Nombre: Requerimientos de documentación para cada audiencia
Criterio de aceptación: Se deberán considerar distintos tipos de documentación según quien sea la audiencia, de forma de que todas las decisiones de diseño sean debidamente registradas.

Testeablidad

RNF32
Nombre: Existencia de *Unit Test* para cada *Feature*
Criterio de aceptación: Cada *Feature* realizada tendrá asociado una prueba unitaria que permitirá verificar el correcto funcionamiento del mismo.

RNF33
Nombre: Existencia de Pruebas de Integración
Criterio de aceptación: Cada requerimiento realizado se verá afectado por pruebas de integración para asegurar el correcto funcionamiento del sistema/subsistema luego de impactado dicho Requerimiento.

RNF34
Nombre: *Record/Playback*
Criterio de aceptación: Deberá considerarse un sistema de prueba que permita guardar y reproducir un número de pasos determinado para emular un escenario de error.

RNF35
Nombre: *Sandbox*
Criterio de aceptación: Se deberán considerar escenarios de prueba donde los servicios de los que se consume no están disponibles, en tal caso se deberá aislar el módulo y utilizar *mocks* en su lugar.

TECNOLOGÍAS

RNF36
Nombre: Todos los sistemas se realizarán desarrollando con Java

RNF37
Nombre: Todos los sistemas se realizaran utilizando el servidor de aplicaciones J2EE

RNF38
Nombre: Se utilizará MySql cómo sistema de gestión de bases de datos.

RNF39
Nombre: Se programará con HTML 4 por compatibilidad

REFERENCIAS BIBLIOGRÁFICAS

[1] G. Kotonya, *Requirements Engineering: Processes and Technique*, USA: John Wiley & Sons Ltd., 1998.

[2] J. Nielsen, "Website Response Times", 2010. [En línea]. Disponible: http://www.nngroup.com/articles/website-response-times/

11.15. RESULTADOS DE LAS CEREMONIAS

Al finalizar cada una de las ceremonias, se recolectaron datos que se detallan a continuación. Se mostrarán los resultados de cada ceremonia y la división de tiempos para cada tarea de las ceremonias.

Sprint Planning

Antes de empezar el nuevo *Sprint*, la primera ceremonia ejecutada era la *Sprint Planning*. El resultado de esta ceremonia era la lista de tareas, *Features* a realizar, las asignaciones a las mismas y el esfuerzo estimado para cada tarea. A continuación se muestra el resumen acumulativo de todas las *Sprint Planning* según sus tareas.

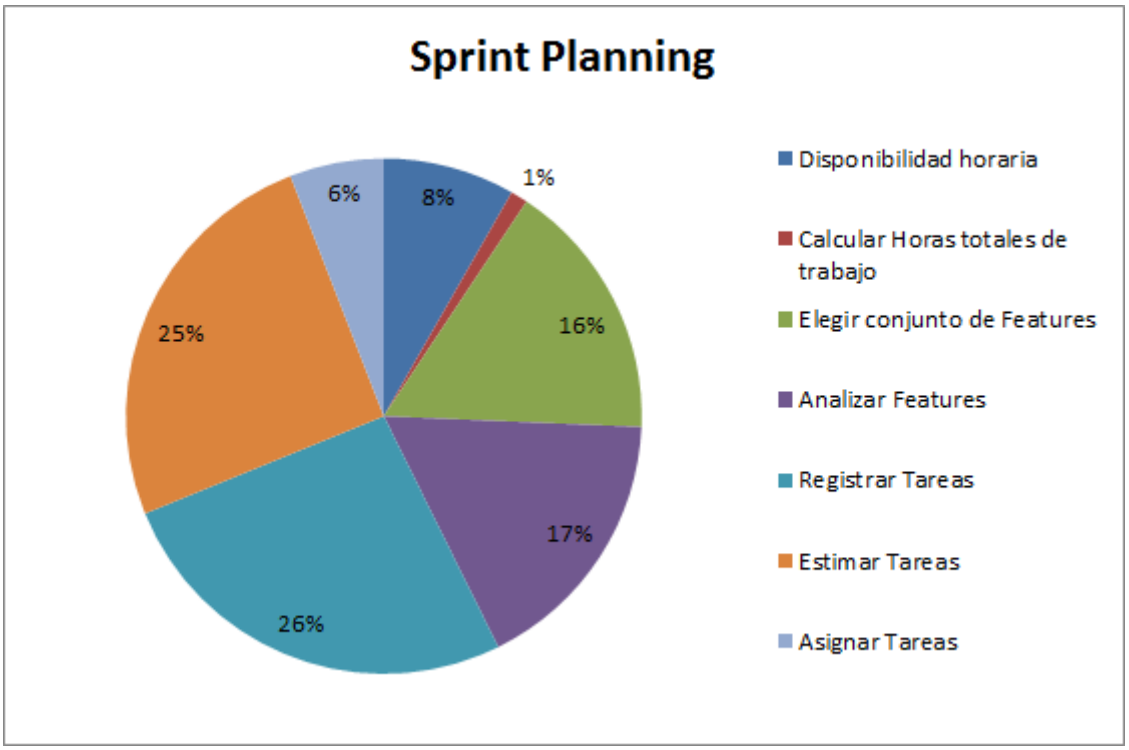


Ilustración 11-33 Acumulativo *Sprint Planning*

Sprint Review

Después de la *Sprint Review*, los resultados recolectados fueron los siguientes:

- Documento de revisión SQA
- *Sprint* cerrado y sus resultados en el AgileFant
- Demo interna del producto
- División de horas de cada tarea de la *Sprint Review*

A continuación se muestran los diferentes resultados recolectados:

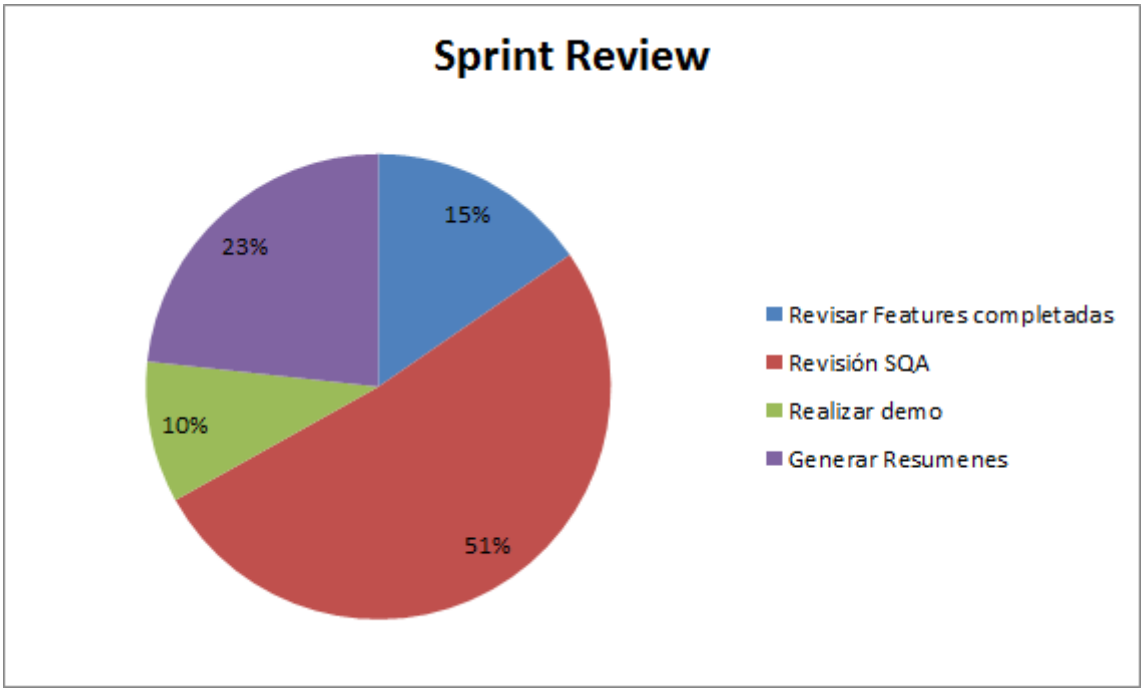


Ilustración 11-34 Acumulativo *Sprint Review*

Sprint Retrospective

Al final de cada *Sprint*, se realizaba la *Retrospective* en la cual se identificaban los elementos a cambiar.

A continuación se muestra la división de tareas de la *Sprint Retrospective*:

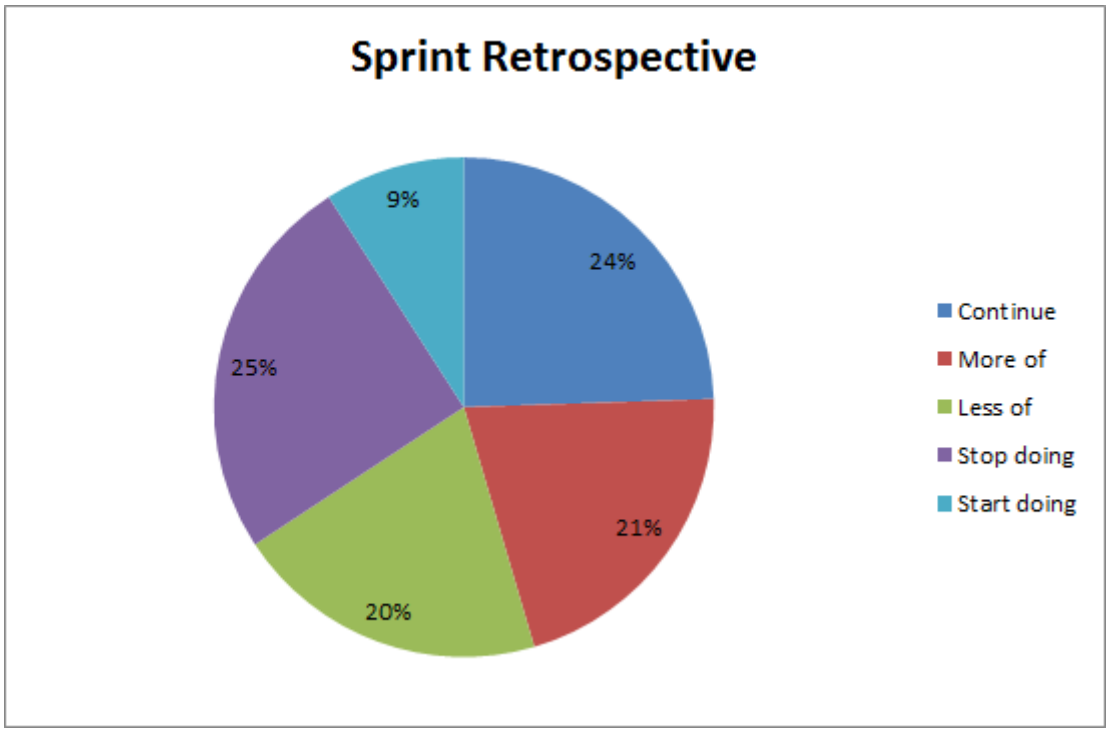


Ilustración 11-35 Acumulativo del tiempo invertido en la *Sprint Retrospective*

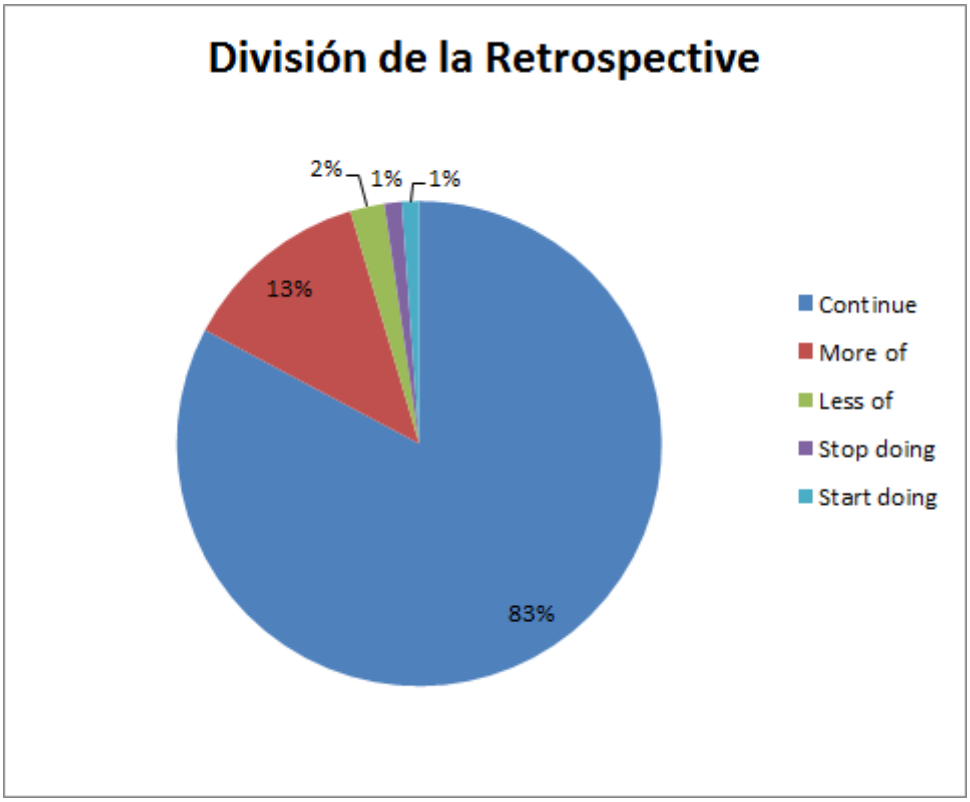


Ilustración 11-36 Acumulativo de la división de resultados de la *Sprint Retrospective*

Estos elementos se registraron en una planilla Excel con el siguiente formato:

En la columna "*Sprint*", se indica a que *Sprint* pertenece el elemento.

En la columna "*What*", se indica el nombre del elemento.

En la columna "*What to do*", cual es la actividad que hay que realizar sobre este elemento.

En la columna "*Why*?" se explica que fue lo que llevó a esta decisión.

Sprint	*What?*	*What to do?*	*Why?*
4	Mejorar organización horaria	*Start doing*	Se estimó más en base a tareas sin tener en cuenta la disponibilidad horaria de cada integrante
	Mejorar división de tareas	*More of*	Se dividió en tareas pero se podría haber ido a un nivel aún más bajo
	Buena comunicación	*Continue*	La comunicación fue importante y buena durante el *Sprint*, hay que seguir implementándola
5	Mejorar organización horaria	*More of*	Se consideró mejor la disponibilidad pero hay que mejorar
	Mejorar división de tareas	*More of*	Se dividió aún más pero faltaron tareas
	Buena Comunicación	*Continue*	2do *Continue*
	Estimación de horas	*More of*	Hay que profundizar la estimación porque hubo una gran diferencia entre lo estimado y lo real en este *Sprint*
	Daily Meetings	*Stop doing*	Las *Daily Meetings* no se están respetando pero esto no está afectando el desempeño, por lo que no son necesarias
	Horas de *Features*	*Less of*	En este *Sprint* hubieron demasiadas horas invertidas en seleccionar *Features*, debe bajar
6	Mejorar organización horaria	*Continue*	1er *Continue*
	Mejorar división de tareas	*Continue*	1er *Continue*
	Buena Comunicación	*Continue*	3er *Continue*

	Horas de *Features*	*Continue*	1er *Continue*
	Estimación de horas	More of	Sigue siendo un problema y tenemos que mejorarlo para el próximo *Sprint*
	Horas de diseño	*Less of*	En este *Sprint* hubieron demasiadas horas invertidas en diseño, debe bajar
7	Mejorar organización horaria	*Continue*	2do *Continue*
	Mejorar división de tareas	*Continue*	2do *Continue*
	Buena Comunicación	*Continue*	4to *Continue*
	Horas de *Features*	*Continue*	2do *Continue*
	Estimación de horas	*Continue*	Mejoramos la estimación de horas teniendo una diferencia muy pequeña entre lo estimado y real, lo cual hay que seguir haciendo
	División del trabajo	*Continue*	Mejoramos la división del trabajo teniendo menos diferencia con el ideal, hay que seguir así
	Registro de horas	*More of*	Falta compromiso para el registro de horas y aumenta la carga al final de *Sprint* para registrar lo que no se registró
8	Mejorar organización horaria	*Continue*	3er *Continue*
	Mejorar división de tareas	*Continue*	3er *Continue*
	Buena Comunicación	*Continue*	5to *Continue*
	Horas de *Features*	*Continue*	3er *Continue*
	Estimación de horas	*Continue*	Mejoramos la estimación de horas teniendo una diferencia muy pequeña entre lo estimado y real, lo cual hay que seguir haciendo
	División del trabajo	*Continue*	Mejoramos la división del trabajo teniendo menos diferencia con el

			ideal, hay que seguir así
	Registro de horas	*More of*	Falta compromiso para el registro de horas y aumenta la carga al final de *Sprint* para registrar lo que no se registró
	Sprint reviews	*Continue*	Se están llevando bien a cabo los *Sprint reviews* y generando resúmenes que nos ayudan a saber si vamos por el buen camino
	Sprint Plannings	*More of*	La *Sprint Planning* se está llevando a cabo pero a veces no se discuten temas importantes. Hay que mejorarlas
	Registro de pruebas	*Continue*	El registro de pruebas no tuvo ningún punto a corregir durante la revisión
	Comunicación	*More of*	Hay que mejorar la comunicación, en algunas tareas no nos estamos entendiendo
9	Mejorar organización horaria	*Continue*	4to *Continue*
	Mejorar división de tareas	*Continue*	4to *Continue*
	Buena Comunicación	*Continue*	6to *Continue*
	Horas de *Features*	*Continue*	4to *Continue*
	Estimación de horas	*Continue*	2do *Continue*
	División del trabajo	*Continue*	2do *Continue*
	Registro de horas	*Continue*	Todos registraron sus horas y no hubo que hacerlo al final del *Sprint*
	Registro del retrabajo	*Continue*	2do *Continue*
	Sprint Reviews	*Continue*	2do *Continue*
	Sprint Plannings	*Continue*	Se discutieron todos los temas pertinentes a la planificación. Surgieron discusiones interesantes.
	Registro de pruebas	*Continue*	2do *Continue*

	Comunicación	*Continue*	La comunicación vía Whatsapp ayudó muchísimo y hasta se realizaron debates que se resolvieron por ese medio
10	Mejorar organización horaria	*Continue*	5to *Continue*
	Mejorar división de tareas	*Continue*	5to *Continue*
	Buena Comunicación	*Continue*	7mo *Continue*
	Horas de *Features*	*Continue*	5to *Continue*
	Estimación de horas	*Continue*	3er *Continue*
	División del trabajo	*Continue*	3er *Continue*
	Registro de horas	*Continue*	2do *Continue*
	Registro del retrabajo	*Continue*	3er *Continue*
	Sprint Reviews	*Continue*	3er Continue
	Sprint Plannings	*Continue*	2do *Continue*
	Registro de pruebas	*Continue*	3er *Continue*
	Comunicación	*Continue*	2do *Continue*
	Registro de pruebas	*Continue*	3er *Continue*
11	Mejorar organización horaria	*Continue*	6to *Continue*
	Mejorar división de tareas	*Continue*	6to *Continue*
	Buena Comunicación	*Continue*	8vo *Continue*
	Horas de *Features*	*Continue*	6to *Continue*
	Estimación de horas	*Continue*	4to *Continue*
	División del trabajo	*Continue*	4to *Continue*
	Registro de horas	*Continue*	3er Continue
	Registro del retrabajo	*Continue*	4to *Continue*
	Sprint Reviews	*Continue*	4to *Continue*

	Sprint Plannings	*Continue*	3er *Continue*
	Registro de pruebas	*Continue*	4to *Continue*
	Comunicación	*Continue*	3er *Continue*
	Aumentar revisiones SQA	*More of*	Las revisiones de SQA se ejecutaron pero quedaron algunos documentos sin revisar por falta de tiempo. Hay que mejorar la organización de las mismas.
	Mejorar revisiones cruzadas	*More of*	Los casos de prueba no fueron todos registrados correctamente, hay que profundizar.
12	Mejorar organización horaria	*Continue*	7mo *Continue*
	Mejorar división de tareas	*Continue*	7mo *Continue*
	Buena Comunicación	*Continue*	9no *Continue*
	Horas de *Features*	*Continue*	7mo *Continue*
	Estimación de horas	*Continue*	5to *Continue*
	División del trabajo	*Continue*	5to *Continue*
	Registro de horas	*Continue*	4to *Continue*
	Registro del retrabajo	*Continue*	5to *Continue*
	Sprint Reviews	*Continue*	5to *Continue*
	Sprint Plannings	*Continue*	4to *Continue*
	Registro de pruebas	*Continue*	5to *Continue*
	Comunicación	*Continue*	4to *Continue*
	Comunicación	*Continue*	4to *Continue*
	Aumentar revisiones SQA	*Continue*	1er *Continue*
	Mejorar revisiones cruzadas	*Continue*	1er *Continue*

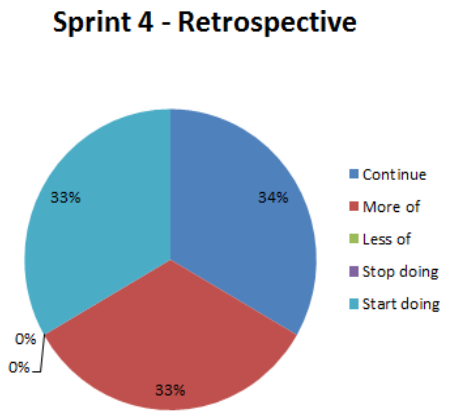
Ilustración 11-37 Resultados Retrospectives

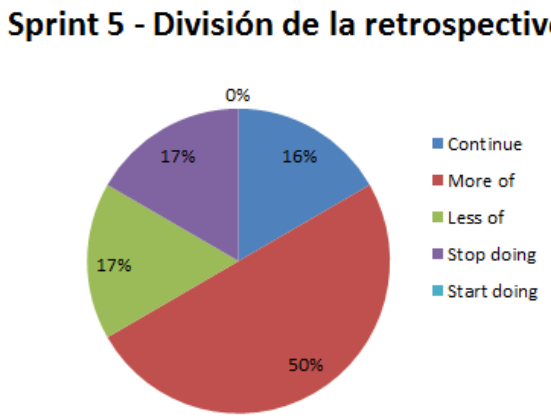
A continuación se muestra el porcentaje de estos resultados totales y por *Sprints*.

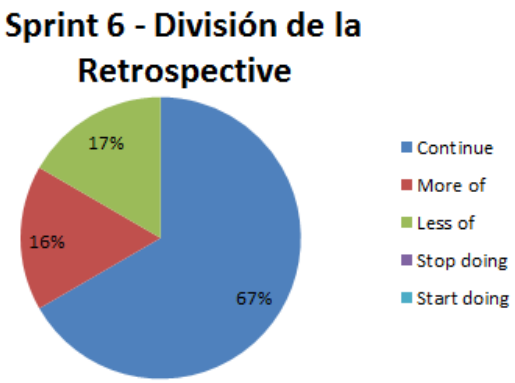
En cada *Sprint* se realizaba un gráfico circular que representan los datos registrados.

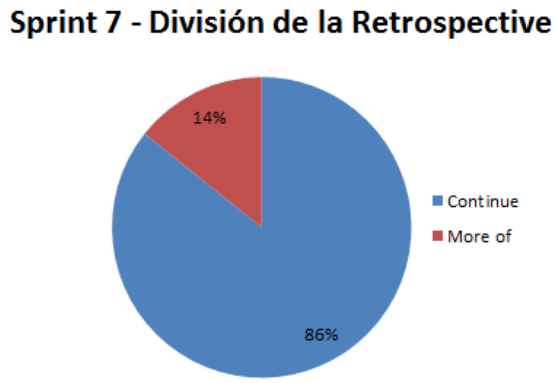
Este registro empezó en el *Sprint* 4.

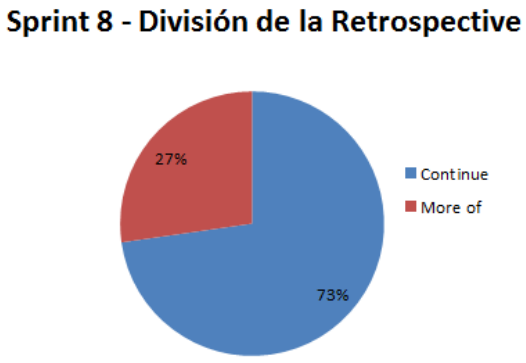
División por categoría - Por *Sprint*:

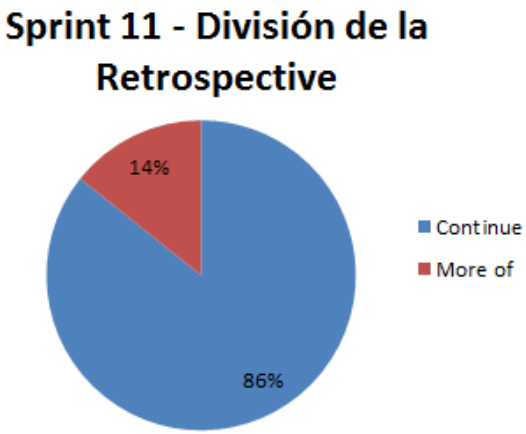


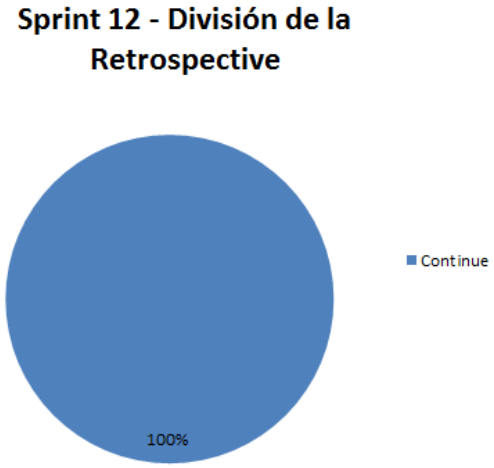








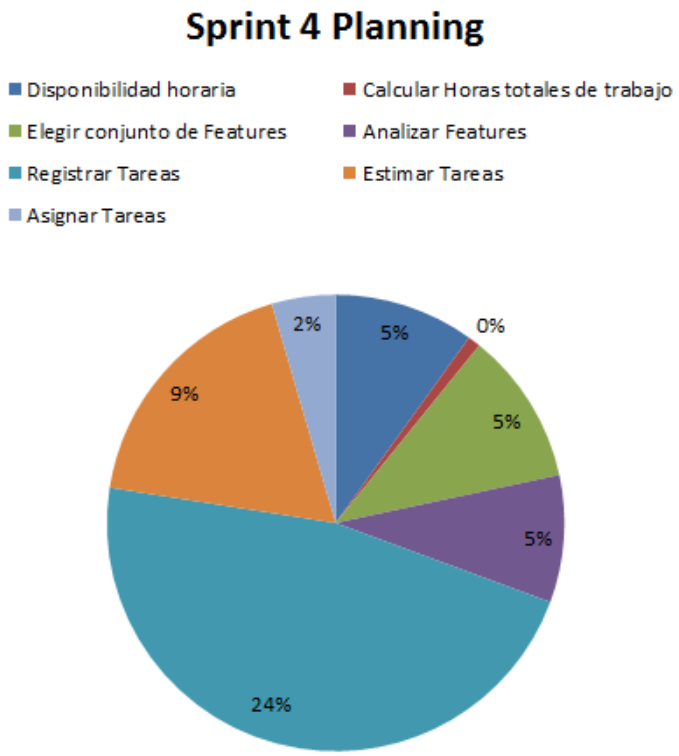


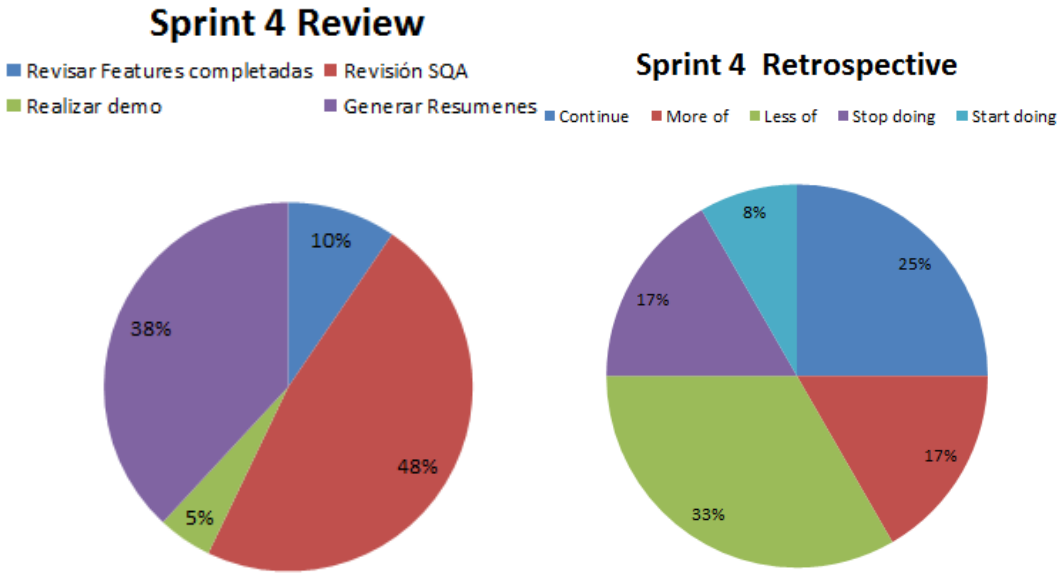


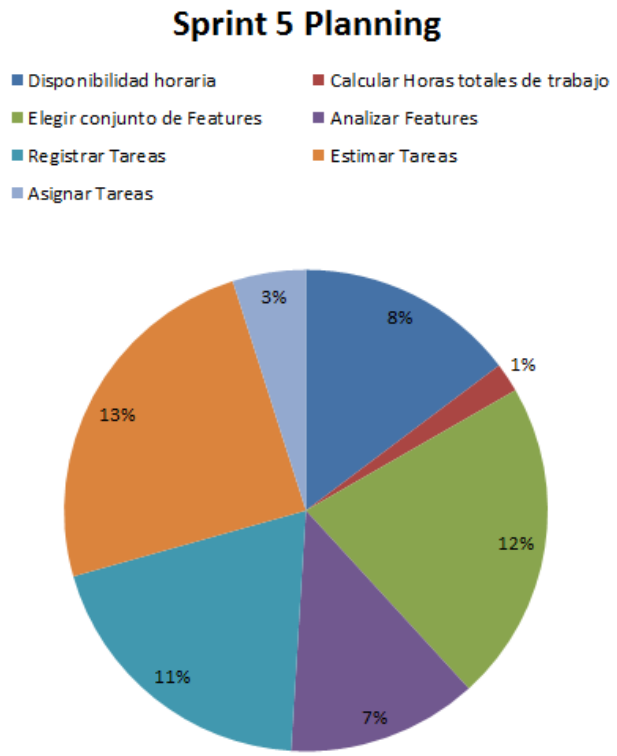
Tiempo invertido en las ceremonias por *Sprints*

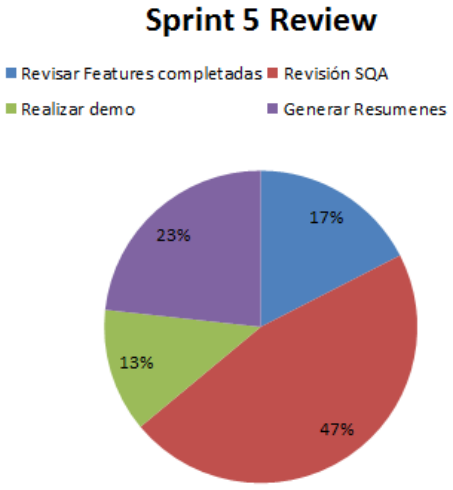
Sabiendo que cada ceremonia está compuesta por una lista de tareas, en cada una de ellas, se registró la cantidad de tiempo invertido.

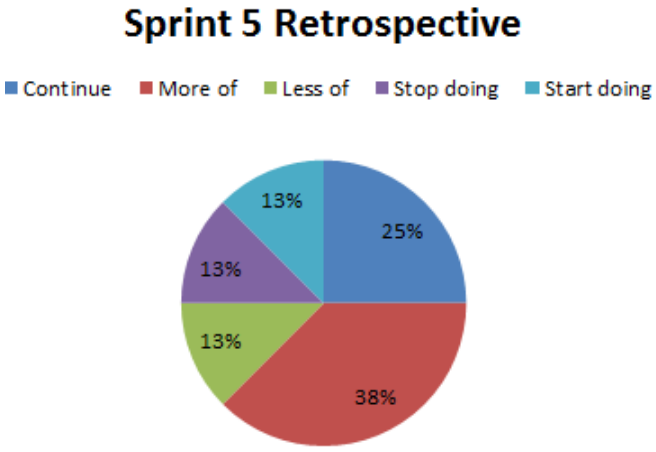
Los valores representan la división en porcentaje de tiempo invertido en cada una de las tareas.


Sprint 4 Planning
Disponibilidad horaria
Calcular Horas totales de trabajo
Elegir conjunto de Features
Analizar Features
Registrar Tareas
Estimar Tareas
Asignar Tareas
5%
0%
5%
5%
24%
9%
2%

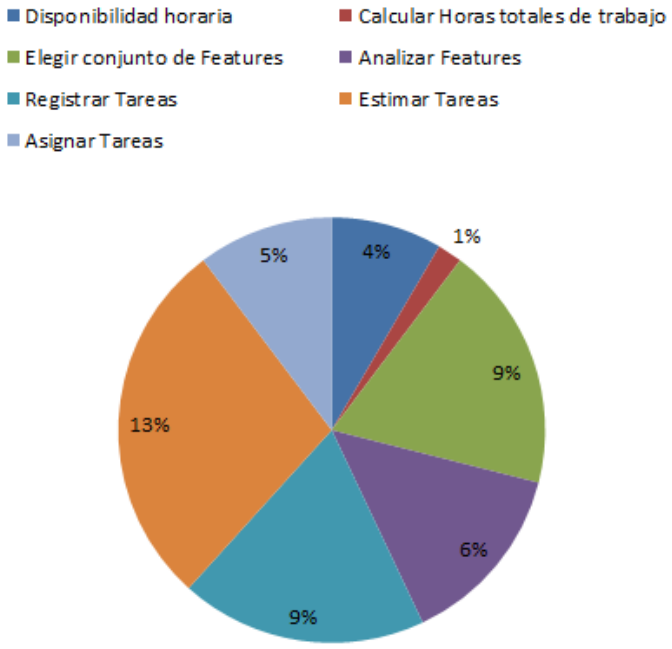

Sprint 4 Review
Revisar Features completadas
Revisión SQA
Realizar demo
Generar Resumenes
10%
48%
5%
38%
Sprint 4 Retrospective
Continue
More of
Less of
Stop doing
Start doing
25%
17%
33%
17%
8%


Sprint 5 Planning
Disponibilidad horaria
Calcular Horas totales de trabajo
Elegir conjunto de Features
Analizar Features
Registrar Tareas
Estimar Tareas
Asignar Tareas
8%
1%
12%
7%
11%
13%
3%

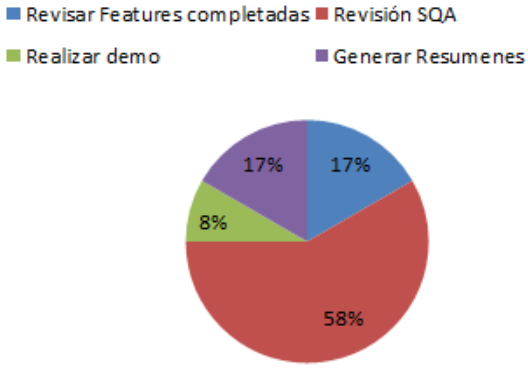

Sprint 5 Review
Revisar Features completadas
Revisión SQA
Realizar demo
Generar Resumenes
17%
47%
13%
23%


Sprint 5 Retrospective
Continue
More of
Less of
Stop doing
Start doing
25%
38%
13%
13%
13%

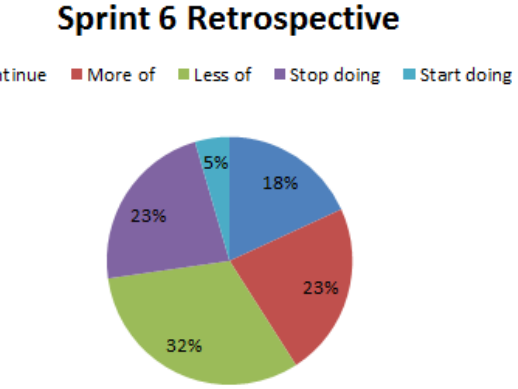
Sprint 6 Planning



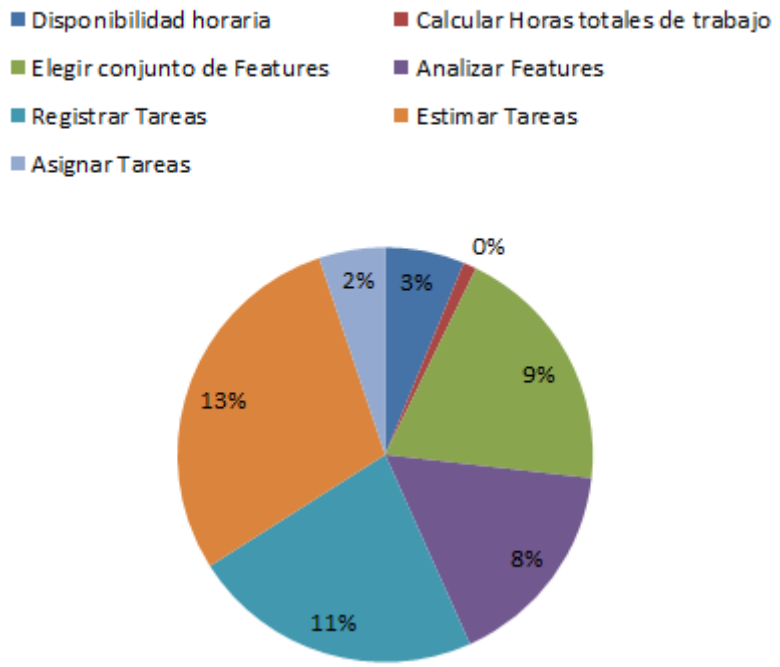
Sprint 6 Review



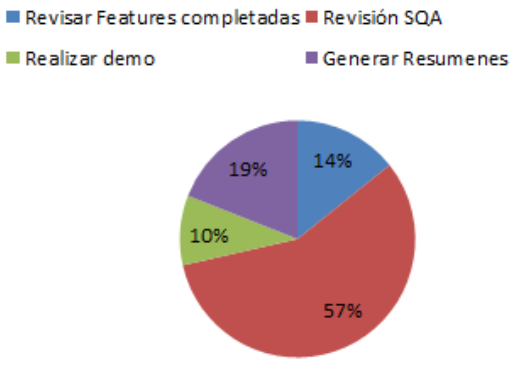
Sprint 6 Retrospective



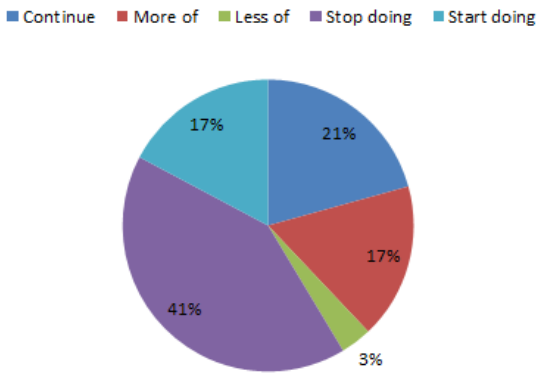
Sprint 7 Planning



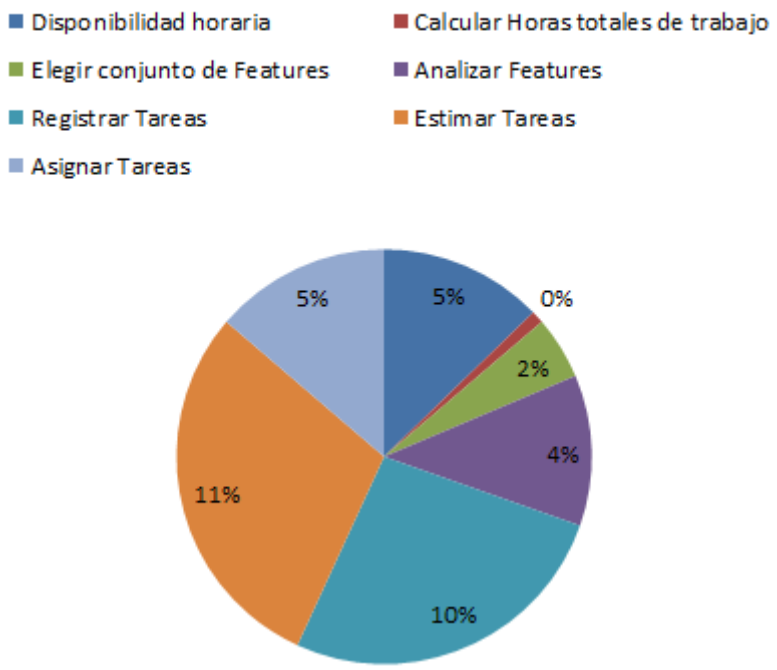
Sprint 7 Review



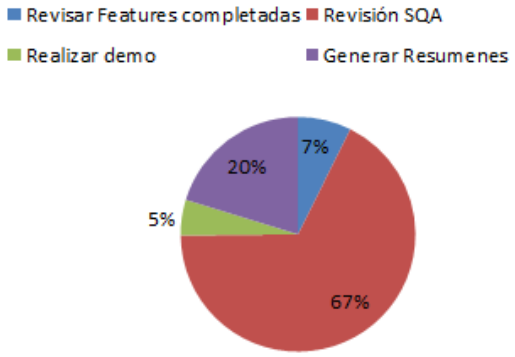
Sprint 7 Retrospective



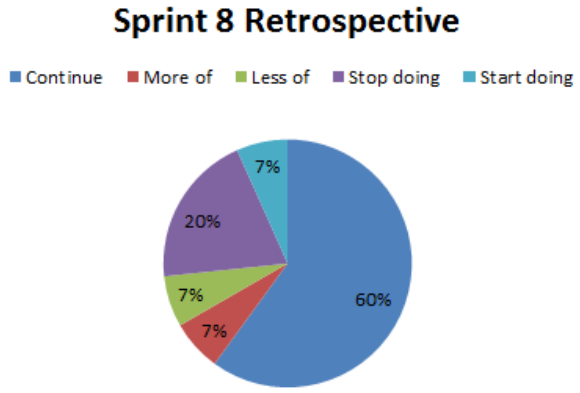
Sprint 8 Planning



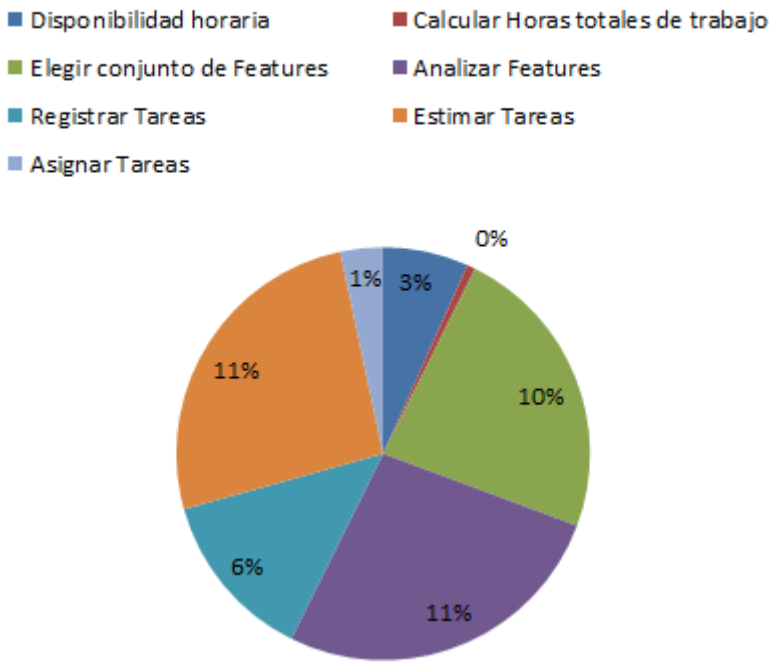
Sprint 8 Review



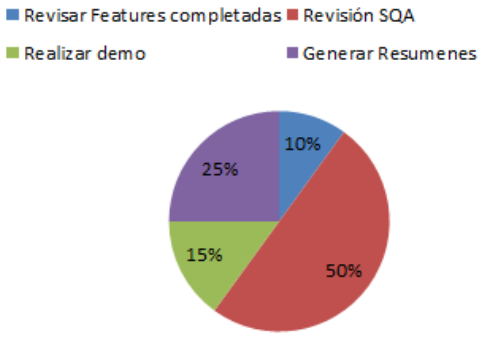
Sprint 8 Retrospective



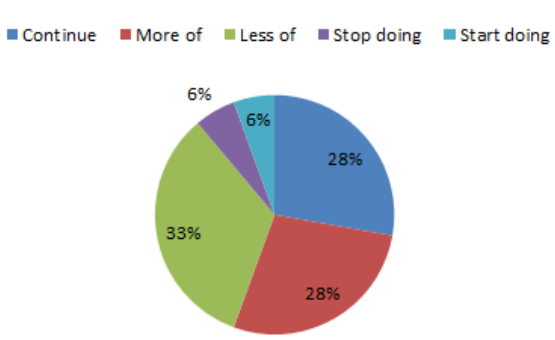
Sprint 9 Planning



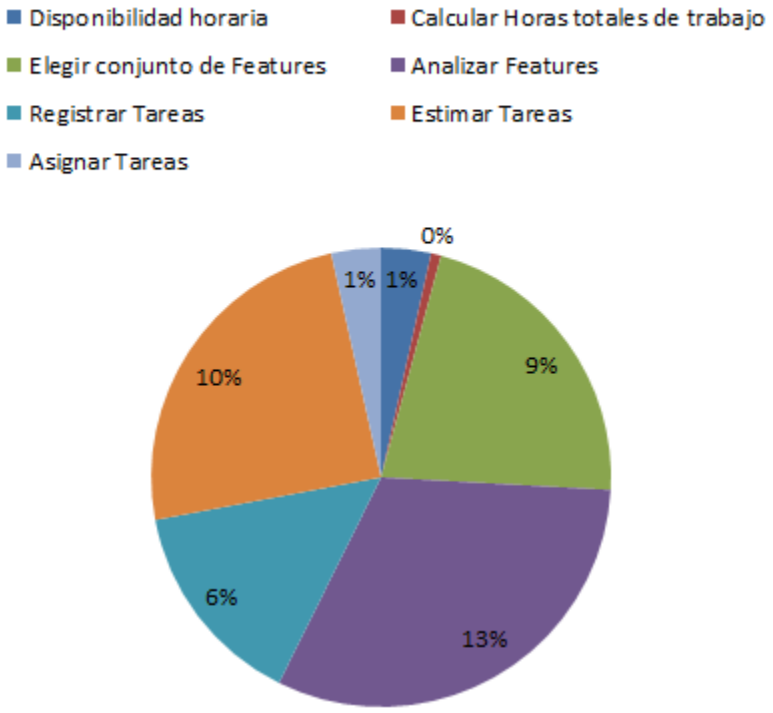
Sprint 9 Review



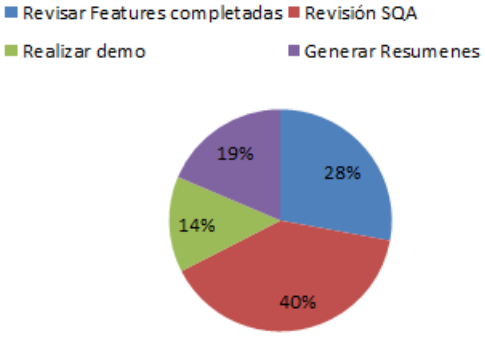
Sprint 9 Retrospective



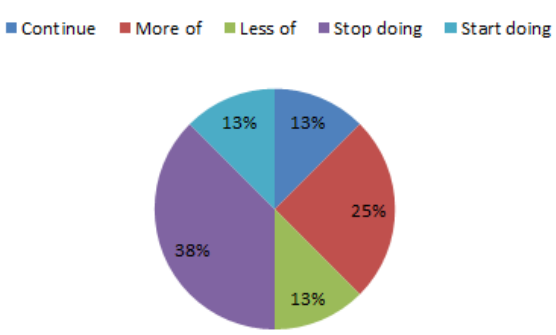
Sprint 10 Planning



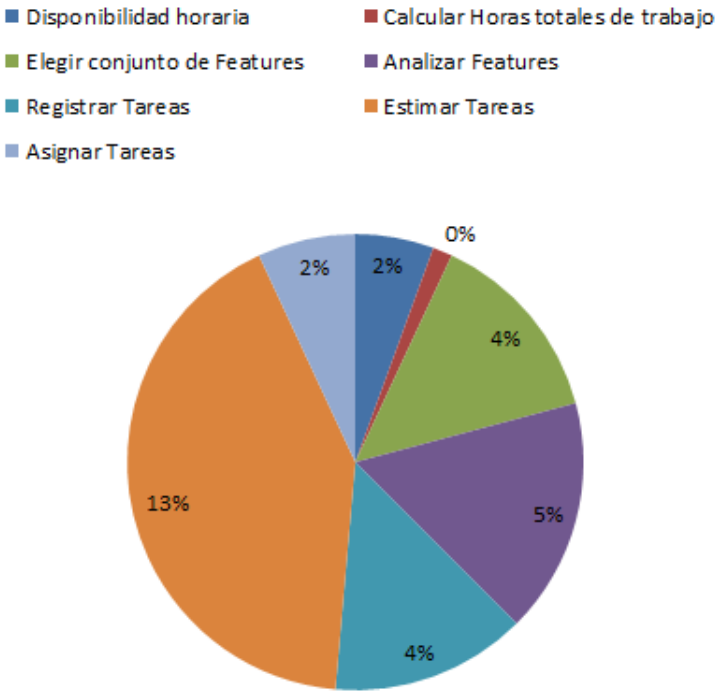
Sprint 10 Review



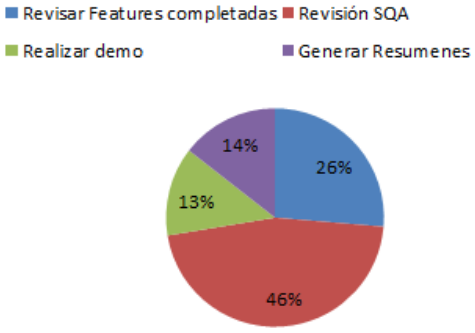
Sprint 10 Retrospective



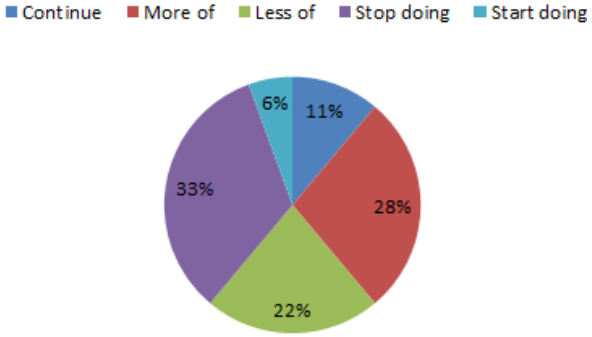
Sprint 11 Planning



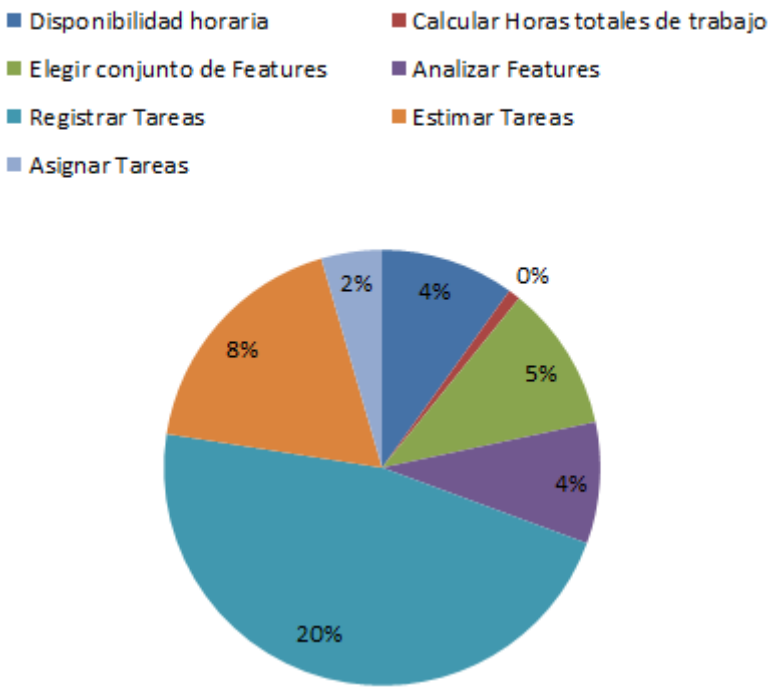
Sprint 11 Review



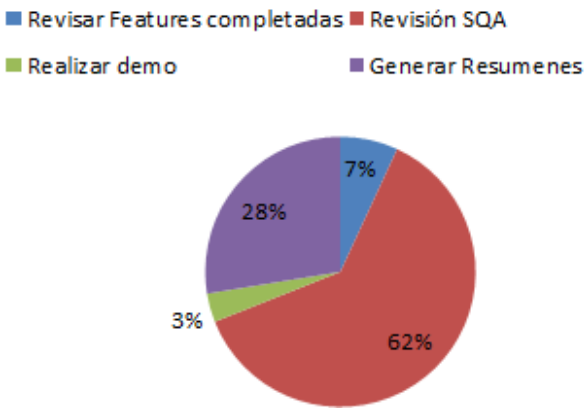
Sprint 11 Retrospective



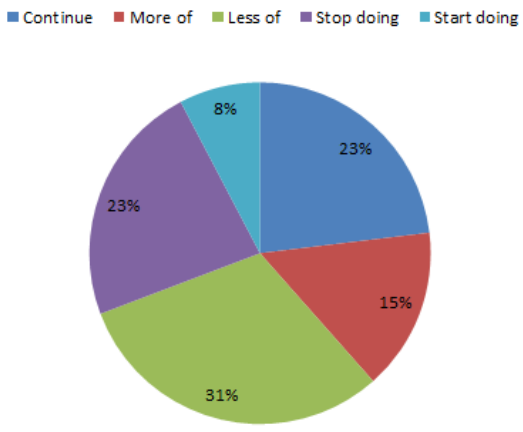
Sprint 12 Planning



Sprint 12 Review



Sprint 12 Retrospective



11.16. PLANNING POKER Y SU APLICACIÓN

¿Qué es *Planning* Poker?

Planning Poker es una técnica de estimación que se utiliza para estimar *User Stories*, Temas, Requerimientos, pero que también se puede aplicar para cualquier tipo de situación en la cual se necesita tomar una decisión.

¿Por qué usar *Planning* Poker?

Según Mike Cohn, la técnica de *Planning* Poker combina opinión experta, analogía y la desagregación en un enfoque de estimación agradable que resulta en estimaciones rápidas y seguras [1].

Gracias a esta afirmación, el equipo decidió investigar más en la técnica para confirmar que su aplicación fuera interesante y beneficiosa.

- *Planning* Poker es muy utilizado en los proyectos que siguen la metodología Scrum lo que se asemeja bien con FDS (*Feature Driven* Scrum) y el proceso.
- Prioriza la comunicación directa e informal
- Todo el equipo participa activamente
- Estimar individualmente para luego discutir y generar una estimación conjunta, lleva a mejores estimaciones.
- La necesidad de que un integrante justifique su decisión delante de todo el equipo lleva a una estimación más precisa.
- ¡Es divertido!

¿Cuál es el proceso de *Planning* Poker?

Se realiza utilizando cartas de Poker (también existen cartas especializadas para *Planning* Poker o se pueden crear cartas personalizadas).

- Deben estar presentes todos los miembros del equipo y se debe designar un moderador. Los demás miembros tienen el rol de estimador.
- Antes de empezar, cada estimador recibe un set de cartas. Cada carta tiene un número. (En el caso de SIGAMAS se realizaron dos mazos de cartas distintas, uno para estimar las *Story Points* de los requerimientos, y otro para las demás decisiones. El primer mazo contiene cartas con los números 3, 5, 8, 13 y 40. El segundo, números del 1 al 5).
- Para cada tema a estimar, el moderador lee su descripción.
- Se realiza un espacio de preguntas, las cuales se discuten y se resuelven.
- Una vez que se hayan contestado todas las preguntas, cada estimador elige una de sus cartas y la pone boca abajo sobre la mesa.
- Simultáneamente se dan vueltas todas las cartas.

¿Qué puede suceder?

Es muy probable que los resultados difieran significativamente. Según M. Cohn, esto es en realidad una buena noticia porque es un momento en el cual se realizarán discusiones.

Las discusiones son muy importantes porque permiten resolver malentendidos o dar a luz factores que no se habían tomado en cuenta antes.

Por ejemplo, para uno de los integrantes, un requerimiento puede haber sido estimado con un 3 *Story Points*, porque no se dio cuenta que ese requerimiento necesita pruebas exhaustivas adicionales (lo que sube su nivel de dificultad), mientras que otro integrante la estimó con 13 por este motivo.

Se toman nota de todas las posibles discusiones.

A partir de ese momento, se necesita volver a estimar hasta que la diferencia sea aceptable y que se llegue a un acuerdo.

Cartas utilizadas

Ilustración 11-38 Estimaciones de requerimientos

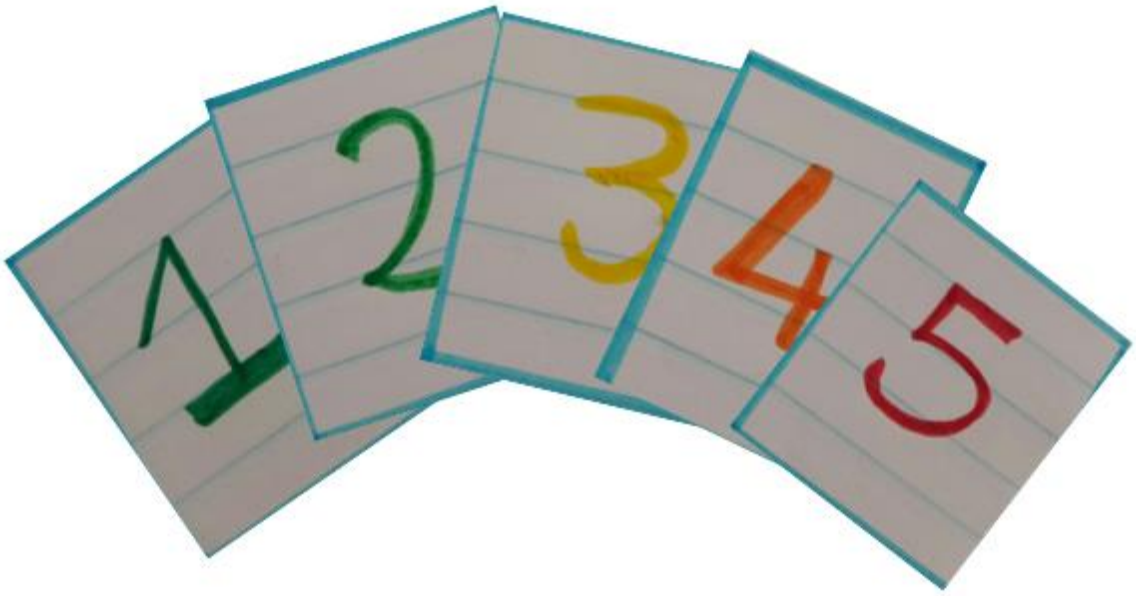


Ilustración 11-39 Otras estimaciones

REFERENCIAS BIBLIOGRÁFICAS

[1] Mike Cohn, *Agile Estimating and Planning, USA*: Pearson Education Inc., 2008.

11.17. RESULTADOS DE LAS REVISIONES SQA

Planilla de Documentos

Sprint	Priorización	Fecha	Nombre	Descripción	Estado
1	Planificar	15/06/14	ESRE	Necesita estar en la carpeta Etapa 1 - Planificación	*Done*
2	Planificar	17/07/2014	ESRE	Necesita la tabla de revisión	*Done*
	Planificar	17/07/2014	Plan SCM	Necesita un párrafo con los Objetivos del documento	*Done*
	Planificar	17/07/2014	ESRE	Texto debe estar justificado	*Done*
3	Planificar	31/07/2014	Documento de arquitectura	No se actualizó la tabla de revisión	*Done*
	Planificar	31/07/2014	Plan SCM	No se actualizó la tabla de revisión	*Done*
	Planificar	31/07/2014	Como documentar el desarrollo	Necesita la tabla de revisión	*Done*
4	Planificar	19/08/2014	Plan SCM	Errores gramáticos a corregir	*Done*
	Planificar	19/08/2014	ESRE viejo	Eliminar este documento	*Done*
	Planificar	19/08/2014	Configuración de desarrollo	Necesita la tabla de revisión	*Done*
5	Planificar	28/08/2014	Plan SCM	Corregir errores señalados	*Done*
	Planificar	28/08/2014	ESRE	El requerimiento RNF01 no sigue el formato	*Done*
	Planificar	28/08/2014	Plan de *Testing*	No se actualizó la tabla de revisión	*Done*

	Planificar	28/08/2014	Preguntas Tabaré	Darle más formato al documento, para que sea más entendible. Separar por preguntas	*Done*
6	Planificar	11/9/14	Plan de *Testing*	La fecha de actualización es incorrecta, revisar	*Done*
	Planificar	11/9/14	Casos de prueba (Google Docs)	Eliminar este documento ya que tenemos la Google *Sheets* con estos datos	*Done*
7	Planificar	25/09/2014	Plan de SQA	Necesita revisión	*Done*
	Planificar	25/09/2014	ESRE	El requerimiento RF09 no sigue el formato	*Done*
	Planificar	25/09/2014	Metodologías de desarrollo	Corregir errores señalados	*Done*
8	Planificar	9/10/14	Reunión FUCREA	Actualmente llamado "*Untitled*", necesita llamarse Reunión FUCREA	*Done*
9	Planificar	23/10/2014	Lecciones aprendidas	Necesita la tabla de revisión	*Done*
	Planificar	23/10/2014	Lecciones aprendidas	Corregir errores señalados	*Done*
10	Planificar	6/11/14	Retrabajo	Eliminar este documento, ya que se pasó su contenido al doc. RESUMEN	*Done*
	Planificar	20/11/2014	Lecciones aprendidas	La última versión no tiene el nombre de la persona que la realizó en la tabla de revisión	*Done*
11	Planificar	4/12/14	ESRE	El requerimiento RF05 no sigue el formato	*Done*
12	Planificar	18/12/2014	ESRE	El requerimiento RF12 no sigue el formato	*Done*

Ilustración 11-40 Otras estimaciones

Planilla de Código

Sprint	Priorización	Fecha	Nombre	Descripción	Estado
5	Delegar	28/08/2014	CuentaBeanLocal	Falta comentar los métodos	*Done*
	Delegar	28/08/2014	UtilCuenta	Falta comentar los métodos	*Done*
6	Planificar	11/9/14	BeanLogin	El nombre del método "*validateUser*" está en inglés, cambiar a español	*Done*
	Delegar	11/9/14	BeanLogin	Falta comentar el método ingresar	*Done*
	Delegar	11/9/14	BeanLogin	La variable "*userPassword*" está en inglés, traducir al español (claveUsuario)	*Done*
7	Planificar	25/09/2014	BeanParcela	Falta comentar el método crearParcela	*Done*
	Planificar	25/09/2014	BeanParcela	El nombre del método centrarmapa no cumple con el estándar, la segunda palabra debe empezar con una mayúscula	*Done*
	Planificar	25/09/2014	BeanParcela	El comentario del método crearMapa no sigue el estándar , revisar	*Done*
	Delegar	25/09/2014	BeanUsuario	Falta comentar los métodos	*Done*
	Delegar	25/09/2014	BeanUsuario	El método modificar debe renombrarse a modificarUsuario para que	*Done*

				sea más entendible	
8	Planificar	9/10/14	UtilUsuario	La variable "managedRol" se crea al principio del método pero no se usa hasta más tarde, declararla al momento de usarla	*Done*
	Planificar	9/10/14	UtilCuenta	La variable "cuenta" se crea al principio del método y no se usa hasta más adelante, declararla al momento de usarla y no antes	*Done*
	Planificar	9/10/14	UtilCuenta	Una variable está declarada con el nombre "lala" y es poco descriptivo, renombrarla para que su nombre sea descriptivo	*Done*
	Delegar	9/10/14	BeanParcela	Documentar el método *init* para que sea más entendible	*Done*
	Delegar	9/10/14	BeanParcela	La variable parcelas se inicia al principio del método y es usada más adelante, declararla al momento de usarla	*Done*
	Delegar	9/10/14	BeanUsuario	Documentar el método progresoCreación	*Done*
	Delegar	9/10/14	BeanUsuario	Corregir error ortográfico en el catch de crearUsuario	*Done*
9	Planificar	23/10/2014	BeanProyecto	Eliminar comentarios innecesarios en el método crearProyecto	*Done*

	Planificar	23/10/2014	beanSeguimiento	Renombrar clase a BeanSeguimiento	*Done*
	Delegar	23/10/2014	UtilsMensajes	Falta comentar los métodos	*Done*
	Delegar	23/10/2014	SeguimientoBean	Falta documentar método recalcularTareasSucesoras	*Done*
	Delegar	23/10/2014	SeguimientoBean	Falta documentar método validarTarea	*Done*
	Delegar	23/10/2014	SeguimientoBean	declarar la variable sePuedeValidar al momento de usarla	*Done*
10	Planificar	6/11/14	NotificacionBean	Falta comentar los métodos	*Done*
	Planificar	6/11/14	SeguimientoBean	Arreglar comentario del método recalcularTareasSucesoras, faltan los parámetros	*Done*
	Planificar	6/11/14	SeguimientoBean	Renombrar diferenciafechas a diferenciaFechas	*Done*
	Delegar	6/11/14	SeguimientoBean	Eliminar espacio adicional en el método recalcularTareasSucesoras	*Done*
11	Planificar	4/12/14	SeguimientoBean	Comentar adentro del método recalcularTareasSucesoras para que este sea más entendible	*Done*
	Planificar	4/12/14	SeguimientoBean	Ídem para método validarTarea	*Done*
	Delegar	4/12/14	SeguimientoBeanL	Falta comentar los últimos	*Done*

			ocal	4 métodos	
12	Delegar	18/12/2014	SeguimientoBeanLocal	Falta comentar el último método	*Done*

Ilustración 11-41 Revisión de Código

Resultados de la evaluación de las métricas generadas

Sprint	Fecha	Nombre	Lectura de la métrica	Decisión tomada
1	15/06/14	*Sprint* 1 *Burndown*	Nada que comentar	--
	15/06/14	Velocidad *Sprint* 1	Nada que comentar	--
	15/06/14	Retrabajo	Nada que comentar	--
2	17/07/14	*Sprint* 2 *Burndown*	Nada que comentar	--
	17/07/14	Velocidad *Sprint* 2	Nada que comentar	--
	17/07/14	Retrabajo	Nada que comentar	--
3	31/07/14	*Sprint* 3 *Burndown*	No se concluyó el B*urndown*. Ver como no generar atrasos.	Se debe mejorar la *Sprint Planning* y ver si estos atrasos son generados por una mala gestión de las mismas
	31/07/14	Velocidad *Sprint* 3	No se pudieron completar todas las *Story Points* planificadas. La política del equipo es lograr terminar todas las *Story Points* definidas en un *Sprint*	El equipo debe comprometerse y realizar horas adicionales para terminas las *Story Points* especificadas. Además, debe ir mirando el *Burndown* para saber si el desarrollo se encuentra o no atrasado.
	31/07/14	Retrabajo	Nada que comentar	--

4	19/08/14	*Sprint* 4 *Burndown*	Siguen habiendo atrasos, aunque estos se disminuyeron	El equipo debe ir mirando los *Burndowns* mientras se ejecuta el ciclo y no solamente al final.
	19/08/14	Velocidad *Sprint* 4	No se pudieron completar todas las *Story Points*.	Aumentar compromiso del equipo aún más. Evaluar otras metodologías ágiles ya que Scrum no está produciendo los resultados esperados.
	19/08/14	Retrabajo	Se generó un gran porcentaje de retrabajo debido a la mala ejecución del *Sprint* 3	No se realizarán acciones adicionales. Se considera que aumentando el compromiso del equipo y analizando otras metodologías esto se podría solucionar.
5	28/08/14	*Sprint* 5 *Burndown*	Se notaron atrasos en la mitad del *Sprint* pero estos se cancelaron hacia el final	--
	28/08/14	Velocidad *Sprint* 5	Nada que comentar	--
	28/08/14	Retrabajo	Nada que comentar	--
6	11/09/14	*Sprint* 6 *Burndown*	Nada que comentar	*Done*
	11/09/14	Velocidad *Sprint* 6	Nada que comentar	--
	11/09/14	Retrabajo	Nada que comentar	--
7	25/09/14	*Sprint* 7 *Burndown*	Se notaron atrasos en la mitad del *Sprint* pero estos se cancelaron hacia el final	--

	29/09/14	Velocidad *Sprint* 7	Nada que comentar	--
	29/09/14	Retrabajo	Nada que comentar	Done
8	09/10/14	*Sprint* 8 *Burndown*	Todo el *Burndown* presentó atrasos. Pero pudo completarse	El equipo no debe olvidarse de ir mirando el *Burndown* a medida que se completan tareas, para no aumentar la carga horaria al final del *Sprint* para poder realizar todas las tareas.
	09/10/14	Velocidad *Sprint* 8	Nada que comentar	--
	09/10/14	Retrabajo	Nada que comentar	--
9	23/10/14	*Sprint* 9 *Burndown*	Nada que comentar	--
	23/10/14	Velocidad *Sprint* 9	Nada que comentar	--
	23/10/14	Retrabajo	Nada que comentar	--
10	20/11/14	*Sprint* 10 *Burndown*	Nada que comentar	--
	20/11/14	Velocidad *Sprint* 10	Nada que comentar	--
	20/11/14	Retrabajo	Nada que comentar	--
11	04/12/14	*Sprint* 11 *Burndown*	Nada que comentar	--
	04/12/14	Velocidad Sprint 11	Nada que comentar	--
	04/12/14	Retrabajo	Nada que comentar	--
12	18/12/14	Sprint 12	Nada que comentar	--

		Burndown		
	18/12/14	Velocidad *Sprint* 12	Nada que comentar	--
	18/12/14	Retrabajo	Nada que comentar	--

Ilustración 11-42 Evaluación de las métricas generadas

Resultados de las pruebas

El objetivo de las pruebas era de encontrar fallas que pudieran ser Leves, Medias o Graves. El sistema no cuenta con fallas graves. A continuación se muestra la gráfica de % de ejecución de casos de prueba con resultados no deseados.

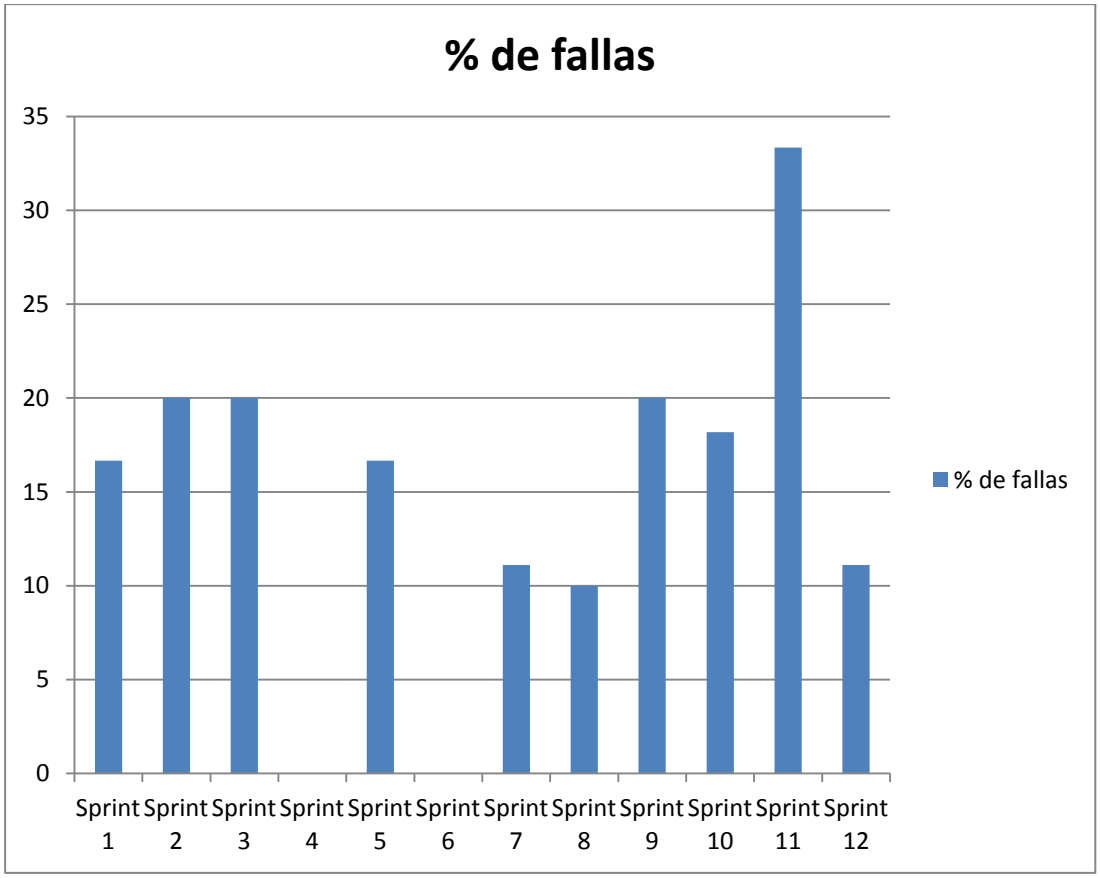


Ilustración 11-43 Resultados de las pruebas

Resultados de la encuesta de usabilidad

A continuación se muestra el resumen obtenido de las encuestas de usabilidad que se realizaron al Experto del dominio Tabaré Alcorta y a Pablo Montes.

La usabilidad se midió con una nota del 0 al 5, siendo 0 una muy mala usabilidad y 5 una muy buena usabilidad. Además de estos valores, todos los *Sprints*, el equipo evaluaba cuál

era el valor mínimo aceptable de los resultados de las encuestas. En rojo se expresa el nivel obtenido y la línea violeta indica el valor mínimo aceptable por el equipo.

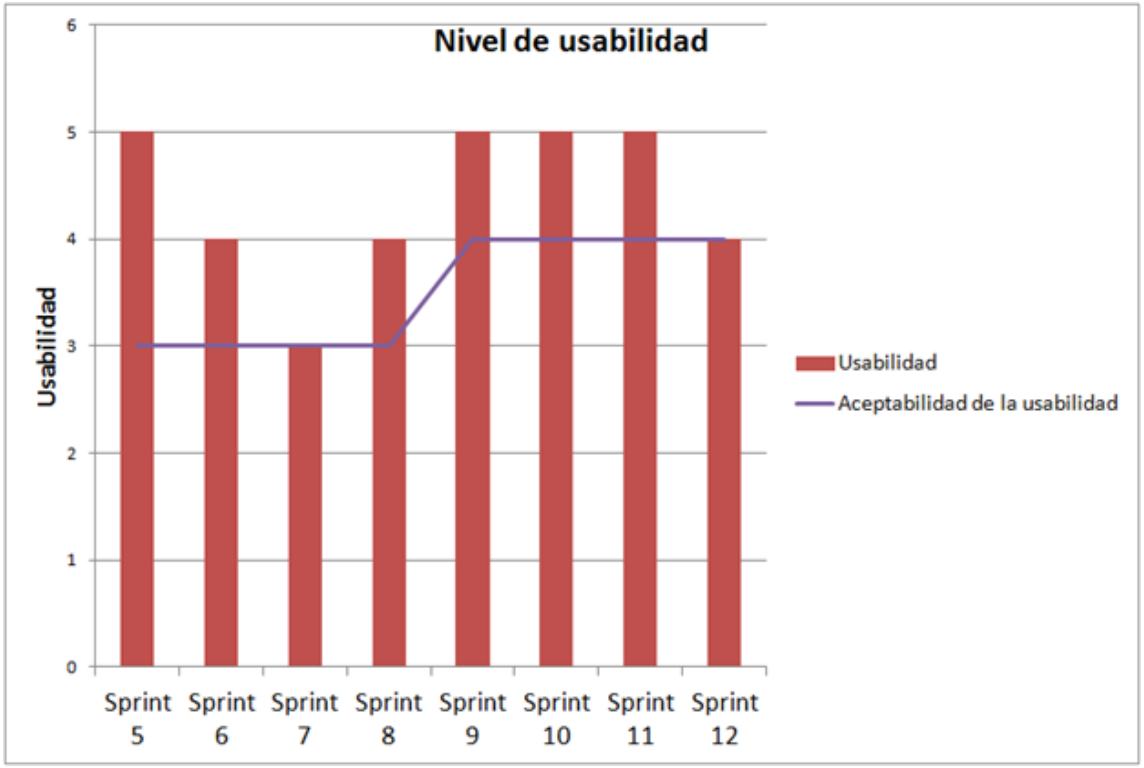
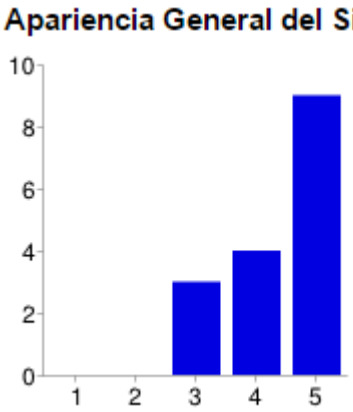
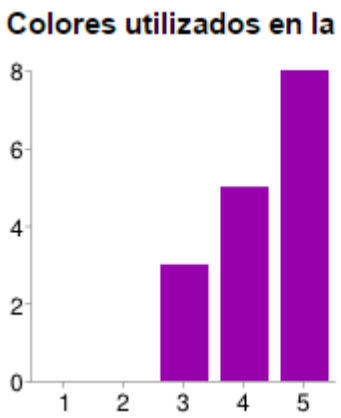


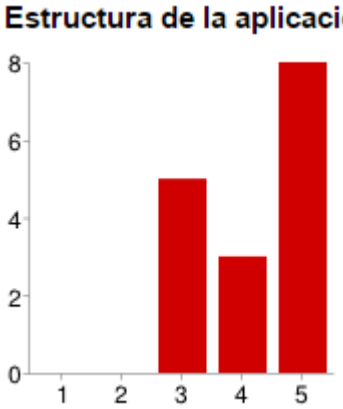
Ilustración 11-44 Resultados encuesta de usabilidad

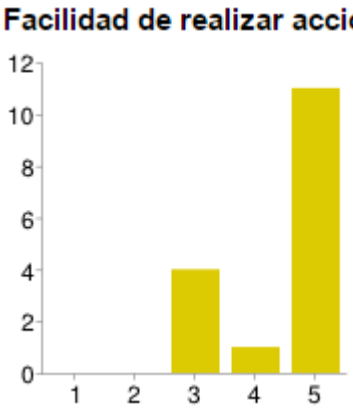
Cómo se puede ver, los niveles de usabilidad nunca estuvieron por debajo del mínimo aceptable, lo cual se considera cómo un éxito para el proyecto. Es importante aclarar que del *Sprint* 5 a 9, el nivel de usabilidad era 3 porqué el equipo se encontraba en etapas tempranas del desarrollo lo cual podía implicar un menor conocimiento de las tecnologías, no habiéndose cumplido la curva de aprendizaje al 100%. A partir del *Sprint* 9, se elevó este mínimo de usabilidad, considerando que el equipo ya estaba familiarizado con las tecnologías. Otra razón por la cual se decidió elevar este mínimo fue, que se acercaba el final del proyecto y se necesitaba entregar un producto con los valores máximos de usabilidad posibles.

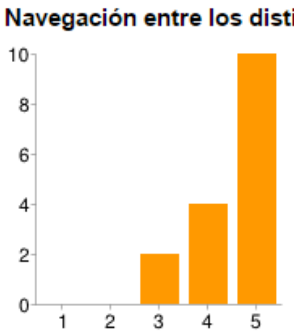
A continuación se muestra el resumen de las encuestas totales obtenidas según las preguntas.











Resultados de la encuesta de aceptación del producto

A continuación se muestra el resumen obtenido de las encuestas de aceptación del producto que se realizaron al Experto del dominio Tabaré Alcorta y, según su disponibilidad, a Pablo Montes.

En azul se expresa el nivel obtenido y la línea roja indica el valor mínimo aceptable por el equipo.

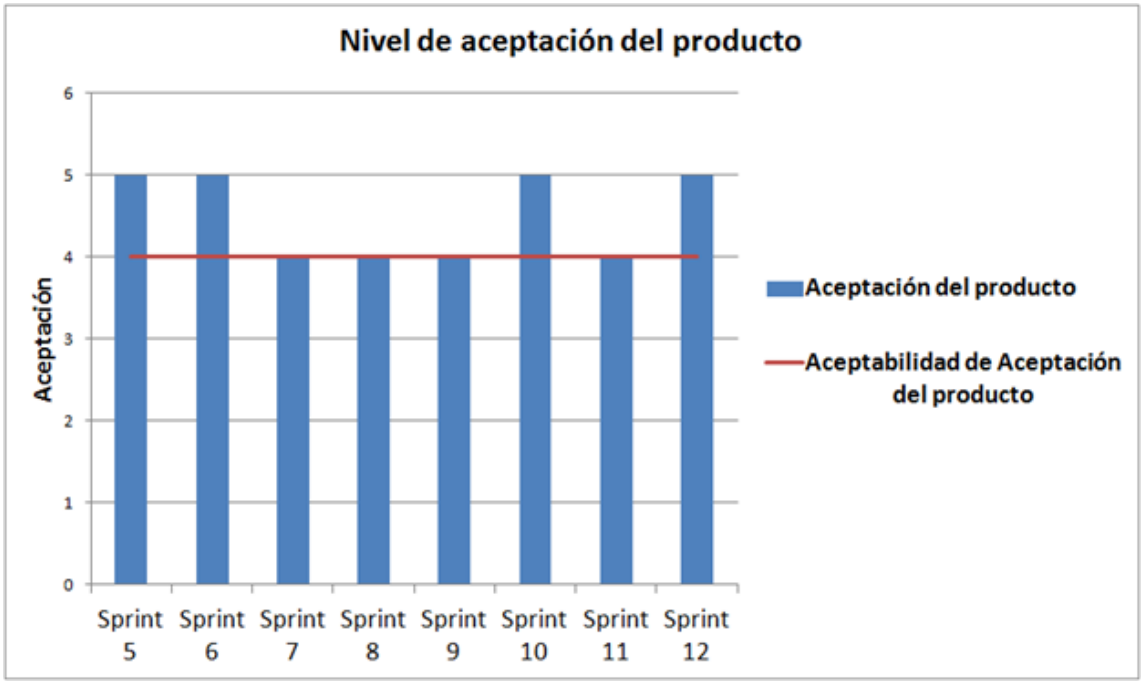
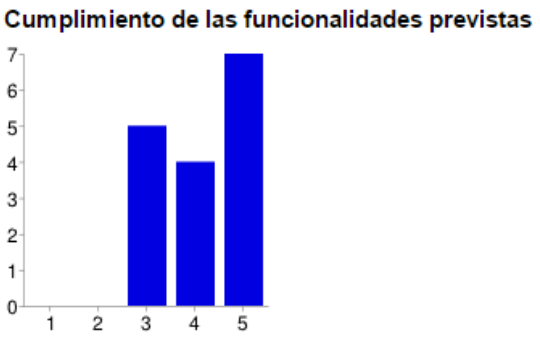
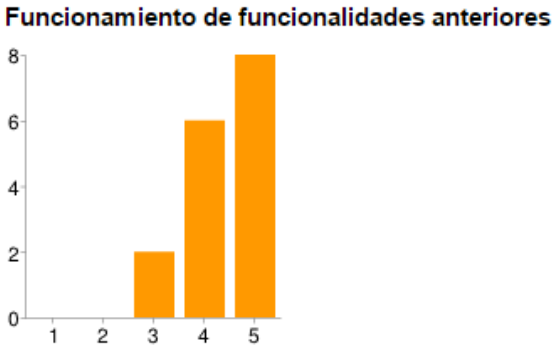


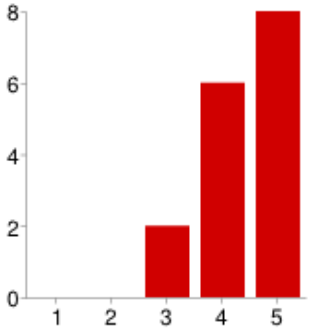
Ilustración 11-45 Resultados de la encuesta de aceptación del producto

A continuación se muestra el resumen de las encuestas totales obtenidas según las preguntas.

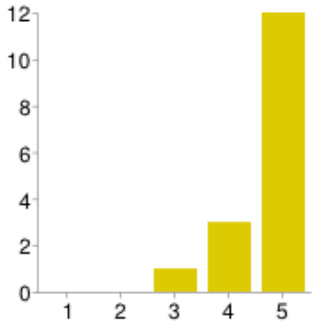




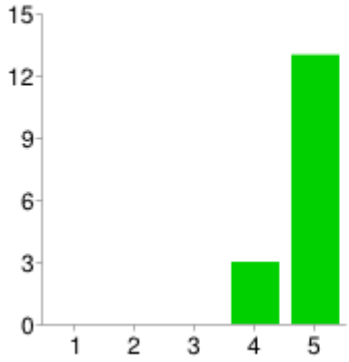
Cantidad de fallas encontradas



Cumplimiento con los prototipos



Funcionamiento esperado de las funcionalidades implementadas



11.18. PLAN DE GESTIÓN DE RIESGOS

El propósito de este anexo es explicar cuáles fueron los planes de respuesta que se realizaron con los riesgos definidos.

La gestión de riesgos consiste en desarrollar opciones que permitan reducir las amenazas a los objetivos del proyecto.

Para que un plan de respuesta sea óptimo, el mismo debe:

- Aplicarse en el buen momento
- Ser aceptado por todos los integrantes
- Adecuado a la gravedad del riesgo

Planes de respuesta a los riegos con MITIGACIÓN

Plan de Respuesta del riesgo R1 - No tener toda la información necesaria para generar los planes de tareas

Contexto

Para poder realizar los proyectos adaptados a un cultivo, se necesitan tareas específicas, propias de dicho cultivo. Estas tareas necesitan realizarse en determinadas fechas o fases y en determinado orden para que el cultivo, nazca, crezca y muera en fechas óptimas.

Objetivos

Investigar para obtener toda la información necesaria para poder realizar un plan de tareas real y justificado

Recursos involucrados

Todo el equipo, durante 14 días

Cronograma

Día 1 a 3 - Realizar encuestas web para tener información sobre las tareas

Día 4 a 7 - Reuniones con los expertos del dominio con preguntas sobre las posibles tareas

Día 8 a 10 - Revisión de las preguntas y listado de tareas

Día 11 a 14 - Registro de tareas en el sistema

Disparador:

El plan de respuesta se disparaba al no encontrarse información sobre tareas de planes de uso de suelo.

Plan de Respuesta del riesgo R2 - Definir un alcance demasiado grande para la entrega del proyecto

Objetivos

Realizar planificaciones exitosas para que el alcance se ajuste o lograr cumplirlo

Recursos involucrados

El arquitecto y el Ing. de Proceso, durante 7 días

Cronograma

Día 1 Analizar alcance actual

Día 2 a 3 - Estimar a grandes rasgos el esfuerzo necesario para cumplir lo restante

Día 4 a 6 - Analizar y realizar ajustes pertinentes

Día 7 - Actualizar ESRE si necesario

Disparador:

Se disparaba este plan de respuesta si un *Sprint* no se consideraba exitoso, con respecto a su *Burndown*, velocidad y retrabajo.

Plan de Respuesta del riesgo R3 - No aceptación de los usuarios

Objetivos

Obtener la máxima aceptación posible por parte de los usuarios

Recursos involucrados

SQAer y Expertos del dominio, 3 días

Cronograma

Día 1 - Realizar una reunión con expertos del dominio para analizar el producto y su usabilidad

Día 2 a 3 - Analizar resultados de la reunión y realizar mejoras pertinentes

Disparador:

Si el resultado de la encuesta de usabilidad es menor al valor mínimo definido.

Plan de Respuesta del riesgo R4 - Conflicto en el equipo

Objetivos

Lograr mantener una relación armoniosa sin discusiones que lleven a disputas

Recursos involucrados

Administrador de Tareas, durante 2 días

Cronograma

Día 1 - Evaluar satisfacción del equipo

Día 2 - Reunir al equipo, discutir y encontrar soluciones al conflicto

Disparador:

Si existe una disputa entre dos o más compañeros.

Plan de Respuesta del riesgo R5 - No poder enfrentarnos a las tecnologías y acceso a datos necesarios

Objetivos

Investigar las tecnologías necesarias para el desarrollo para poder enfrentarnos a las mismas

Recursos involucrados

Arquitecto, durante 8 días

Cronograma

Día 1 a 3 - Buscar información sobre los datos necesarios

Día 4 a 5 - Analizar tecnologías encontradas, pros y contras, beneficios y perjuicios

Día 5 a 7 - Elaborar documentación acerca de estas tecnologías

Día 8 - Revisar documentación con todo el equipo

Disparador:

Si se debe enfrentar a una tecnología nueva totalmente desconocida.

Plan de Respuesta del riesgo R7 - Cambios en el proceso

Objetivos

Poder enfrentarnos y adaptarnos a los cambios en el proceso

Recursos involucrados

Todo el equipo, durante todo el proyecto

Cronograma

Aplicar el proceso e ir evaluándolo

Plan de Respuesta del riesgo R10 - No tener las métricas adecuadas

Objetivos

Poder tener métricas que representen el trabajo realizado, y posibilite tomar medidas si no se está ejecutando cómo deseado.

Recursos involucrados

SQAer, durante 1 día

Cronograma

Día 1 - Evaluar las métricas generadas

Disparador:

Si un *Sprint* no es “exitoso” según sus métricas

Planes de respuesta a los riesgos con EVITAR

Plan de Respuesta del riesgo R9 - No tener un proceso definido

Objetivos

Tener un proceso definido y que se pueda aplicar y ejecutar

Recursos involucrados

Ingeniero de proceso, durante 3 días

Cronograma

Día 1 a 3 - Definir el proceso

Disparador:

Si no hay proceso definido

11.19. ESTÁNDAR DE CODIFICACIÓN

Para el estándar de codificación se basó en *Google Java Style* [1] y se realizaron las modificaciones pertinentes para generar nuestro propio estándar para Java.

A continuación se listan las buenas prácticas que nuestro código debe cumplir:

- La estructura de la clase debe ser igual que las clase especificada en el diseño.
- Todas las clases especificadas en el diseño deben estar presentes en el código.
- Evitar líneas de más de 100 caracteres.
- Si una expresión no puede estar en una sola línea, quebrarla en una coma, operador y deberá estar alineada con el comienzo de la expresión de la línea anterior.
- Se recomendará una declaración por línea.
- Se debe intentar inicializar las variables locales donde se declaran.
- Los paquetes se escribirán siempre en letras minúsculas.
- Las clases debe tener la primera letra en mayúscula.
- Los métodos son verbos escritos en minúscula, cuando el método tiene varias palabras, cada palabra comenzará en mayúscula.

1. Denominación de archivos

Los nombres de las clases son *case-sensitive*.

- Cada nombre debe empezar con una mayúscula y cada palabra dentro del nombre empieza con una mayúscula también.
- No podrán existir espacios dentro del nombre ni caracteres especiales.
- Los nombres serán en español.
- Los paquetes Java deberán tener la siguiente denominación
    - En los módulos EJB
      uy.edu.ort.SIGAMAS.[modulo].[entidad]

      uy.edu.ort.SIGAMAS.[modulo].[entidad].utils

      (opcional) uy.edu.ort.SIGAMAS.[modulo].[entidad].excepciones
    - En los Proyectos WEB
      uy.edu.ort.SIGAMAS.SIGAMASweb.[entidad]

2. Formato

<u>Idioma</u>

La codificación se realizará en español (nombre de los métodos, nombre de las clases, nombre de las variables, etc.)

<u>Llaves</u>

Las llaves se usan siempre donde pueden ser opcionales. En los *if, else, for, do y while* mismo si el contenido está vacío o solo contiene una línea.

Las llaves siguen el estilo de Kernighan y Ritchie [2] (*Egyptian brackets*) para bloques no vacíos y bloques con la siguiente estructura:

- No introducir salto de línea antes de abrir la llave
- Introducir salto de línea después de abrir la llave
- Introducir salto de línea antes de cerrar la llave
- Introducir salto de línea después de cerrar la llave al menos que esté seguido por un *else* o una coma.

Ejemplos:

Formato correcto:

```
if (condicion) {
        codigoIf();
} else {
        codigoElse();
}
```

Formatos incorrectos:

```
if (condición) codigoIf();
```

```
if (condicion)
{
        codigoAEjecutar();
}
```

```
if (condicion) {
        codigoIf();
}
else {
        codigoElse();
}
```

Gets y Sets

Los *gets y sets* se encapsularán con un *editor fold* de la siguiente forma

```
// <editor-fold defaultstate="collapsed" desc="Gets y Sets">
        // lista de gets y sets
// </editor-fold>
```

Este encapsulamiento se situará al final de la clase, antes de la última llave de cierre.

3. **Bloques vacíos concisos**

Un bloque vacío debe estar cerrado inmediatamente después de abierto, sin ningún carácter o línea entre las llaves.

Ejemplos:

Formato correcto:

```
void bloqueVacio() { }
```

Formatos Incorrectos:

```
void bloqueVacio() {
}
```

```
void bloqueVacio()
{
}
```

4. **Indentación de 4 espacios (un tabulador)**

Cada vez que se codifica un nuevo bloque, la indentación aumenta de 4 espacios. Cuando el bloque termina, la indentación vuelve a la indentación del nivel anterior. El nivel de indentación se aplica para código y también comentarios a lo largo del bloque.

Ejemplos:

Formato Correcto:

```
void bloque() {
    //Comentario con 4 espacios
    4Espacios();
}
```

Formatos Incorrectos:

```
void bloque() {
  //Comentario con 2 espacios
    4Espacios();
}
```

```
void bloque() {
    //Comentario con 4 espacios
  2Espacios();
}
```

5. Una declaración por línea

Cada declaración está seguida por un salto de línea

Ejemplos:

Formato Correcto:

```
void metodo() {
  metodo2();
  metodo3();
}
```

Formato Incorrecto:

```
void metodo() {
  metodo2(); metodo3();
}
```

6. Ajuste de línea (*line wrapping*)

El ajuste de línea ocurre cuando el código ocupa más de una sola línea y se divide en múltiples líneas para evitar pasarse del límite definido para cada columna. En este caso se definirá un límite de 100 caracteres. Existen muchas formas de ajustar una línea, por lo que se entenderá que cualquier forma será aceptable mientras no genere incoherencias en el código.

7. Constructores específicos

Clases *Enum*

Después de cada coma que separa una constante del *enum*, un salto de línea es opcional.

Una clase *enum* que no contiene métodos ni documentación puede ser formateada opcionalmente cómo si fuera una declaración de un *Array*.

Ejemplo:

```
private enum Enum { VALOR1, VALOR2 }
```

Declaración de variables

Nombre

El nombre de las variables debe ser conciso y descriptivo acorde a su funcionalidad.

Una variable por declaración

Cada declaración de variable sólo declara una variable. Declaraciones del estilo int a, b no se usan.

Declaración al necesitarla

Las variables locales no se declaran al principio de cada bloque. Se declaran al momento de ser usadas para minimizar su alcance (*scope*).

Arrays

No declarar al estilo C

Los corchetes son parte del tipo no de la variable: *String[] args* y no *String args[]*

Switchs

Indentación

La indentación del *switch* sigue el mismo comportamiento de la indentación general.

Seguimiento con comentario

Si un case no termina con *un break, continue, return* o con una excepción, debe terminar con un comentario (en general // *fall through*) en nuestro caso será // sigue.

Esto indica que la ejecución (capaz) seguirá al próximo case.

Case por defecto

Cada bloque de *switch* tendrá una declaración default mismo si no contiene código.

Anotaciones

Las anotaciones que se aplican a una clase, método o constructor aparecen inmediatamente después del bloque de documentación y cada anotación está listada con una línea para cada una de ellas.

Ejemplo:

@Override

@Nullable

public String metodo(){ .. }

8. Comentarios

- Los comentarios se harán antes del inicio de cada método o inicio de cada clase.
- Los comentarios pueden tener la siguiente modalidad: /* ... */, o //.
- Los comentarios serán en español.
- En el caso de los comentarios de múltiples líneas /* ... */ cada nueva línea deberá empezar con *

Ejemplos:

Comentario de descripción de método

```
/**
  * nombreDelMetodo descripción corta
```

```
   *
   * @param lista de parámetros (opcional)
   * @return TipoDeRetorno (opcional)
*/
```

Comentarios varios

```
// Este es un comentario
// valido

/* Este también es un comentario

* Valido*/
```

REFERENCIAS BIBLIOGRÁFICAS

[1] Google Java Style. [En línea]. Disponible: https://google-styleguide.googlecode.com/svn/trunk/javaguide.html

[2] B. W. Kernighan , D. M. Ritchie, *The C Programming Language,* 2nd ed., USA: Upper Saddle River, NJ07458: Prentice Hall P T R, 1988.

11.20. DEVOLUCIÓN DE LAS REVISIONES

Primera Revisión

Fortalezas*:*

- La temática es muy interesante
- Se nota el compromiso de los integrantes del equipo
- Se realizó una gran cantidad de trabajo hasta la fecha

Mejoras a realizar:

- Plantear mejor el problema
    - Hablar más en detalle en qué consiste el desafío
    - Explicar quiénes son los entes que regulan las actividades del suelo
    - Mejorar la introducción para que se entienda mejor el problema
- Faltó definir los objetivos del proceso
- Faltó definir al cliente
- Faltó especificar si es un servicio y a quien se le ofrece
- Si se va a usar Scrum, decir quién es el *Product Owner* y de que reuniones va a participar. Y si se descarta, explicar por qué.
- Faltó hablar de la ejecución del proceso
- ¿Cómo se va a tomar acción sobre los riesgos definidos?
    - Hablar de los planes de contingencia.
- Profundizar en calidad
    - Qué aspectos se usan para medir y cómo se mide
- Profundizar en SCM
- Profundizar en la elección de las tecnologías

Segunda Revisión

Fortalezas:

- Es un servicio muy interesante
- El equipo se expresó claramente
- No había exceso de texto en la presentación

Mejoras a realizar:

- Es importante mencionar si están trabajando con un cliente o un experto del dominio
- Hay que mencionar que es un emprendimiento
- Faltó describir quiénes eran los usuarios
- Faltó mencionar las métricas
- Profundizar en la arquitectura
- Hablar menos de SCM
- Faltó hablar más de la parte de *Testing*

Tercera Revisión

Fortalezas:

- Proyecto muy interesante
- Es una temática muy compleja y desconocida y el equipo se está enfrentando bien a la misma

Mejoras a realizar:

- Faltó mencionar donde va a estar hosteado el SaaS
- No se entró en detalle en los obstáculos que se enfrentaron y fueron difíciles de resolver
- Faltó justificar la elección de las métricas
    - La métrica de horas estimadas vs horas reales no es útil, ver de combinarla con la velocidad
- Faltó un cierre a la presentación con las lecciones aprendidas
- Hubo problemas técnicos durante la presentación, lo cual puede resultar muy problemático en una defensa final. Practicar más en varios ambientes (computadoras, salones, etc.) para que todo funcione correctamente.

11.21. PLAN DE ASEGURAMIENTO DE LA CALIDAD (SQA) [4]

Fecha	Versión	Quién	Descripción	Revisión
01/12/2014	1.0	Mauro	Creación del documento Plan SQA	07/12/2014
07/12/2014	1.1	Andrés	Actualización del Plan SQA	09/10/2014
09/10/2014	1.2	Clara	Actualización del Plan SQA	10/10/2014
20/10/2014	1.3	Mauro	Actualización del Plan SQA	21/10/2014
11/11/2014	1.4	Mauro	Actualización del Plan SQA	12/11/2014
15/12/2014	1.5	Mauro	Actualización del Plan SQA	19/12/2014
20/01/2015	1.6	Mauro	Actualización del Plan SQA	25/01/2015

1. Propósito

El propósito de este plan es el de definir todas aquellas actividades que los integrantes del equipo deben desarrollar para lograr la calidad deseada para el producto y el proceso de desarrollo. Se especificarán los estándares a utilizar, la asignación de responsabilidades y los procedimientos y guías a seguir.

Para este proyecto en particular se han definido varias actividades de aseguramiento pero no se ha profundizado en las pruebas ya que el objetivo es el de lanzar una primera versión Beta del sistema. Sin embargo se aseguró que no posea fallas graves ya que el proyecto es para la gestión de producción de grandes cantidades de cultivos, cualquier error en el sistema puede ocasionar pérdidas importantes para el productor.

Es muy importante invertir en calidad ya que ayuda a lograr los objetivos planteados en el proyecto. Se logra a su vez obtener un proceso reutilizable en el futuro.

2. Alcance

Durante el plan se establecerán actividades para el aseguramiento de la calidad como también para la gestión de la misma. Se establecen actividades desde dos puntos de vista: asegurar un producto de calidad y que el proceso genere un producto de calidad.
Todo integrante del proyecto es involucrado en el plan de calidad como también todas las áreas de gestión. Esto se realiza definiendo la actividad a realizar en cada área y quién o quiénes son los responsables de que dicha actividad se cumpla.

3. Referencias y Estándares

- IEEE 2008 std. 1028-1008, *IEEE Standard of* Software *Reviews and Audits*. [1]

Es una parte muy importante de la evaluación continua de los productos de Software a lo largo del ciclo de vida de desarrollo. Esta norma proporciona orientación para el revisor o auditor en la realización de evaluaciones. Es específico para procesos y productos del ciclo de vida del Software.

Revisión: Es una evaluación de elementos de Software o estado de un proyecto para determinar discrepancias con los resultados planeados y recomendar mejoras.

Auditoría: Es una evaluación independiente de productos o procesos para verificar el cumplimiento de estándares, guías, especificaciones y procedimientos; basándose en criterios objetivos.

- IEEE Std 730.1-1995, IEEE *Guide for* Software *Quality Assurance Planning*. [2]

Dicho Estándar es una recomendación para elaborar un Plan de Aseguramiento de la Calidad de Software para los proyectos de desarrollo. Proporciona los requisitos mínimos aceptables para la preparación y el contenido de los planes de aseguramiento de la calidad de Software. Fue escrito para ser utilizado en las fases de desarrollo y mantenimiento del Software. El plan SQA sirve como guía de las actividades de SQA en el proyecto.

Este estándar describe la preparación y los contenidos de los planes SQA. Entre las principales actividades de SQA se incluye la gestión, documentación, mediciones, revisiones, *Testing*, informes de problemas y las acciones correctivas, control de medios de comunicación, control de proveedores, gestión de registros, capacitación y gestión de riesgos. Todas las cuales se describen en este Estándar.
- IEEE Std 1012-1998, IEEE *Standard for* Software *Verification and Validation*. [3]

Determina si los productos de desarrollo de una determinada actividad se ajustan a los requisitos de esa actividad, y si el Software satisface sus usos y necesidades de los usuarios previstos. Esta determinación puede incluir el análisis, evaluación, revisión, inspección, evaluación y pruebas de los productos y procesos de Software. Procesos de Validación & Verificación apunta a evaluar el Software en el contexto del sistema, incluyendo el entorno operativo, hardware, interfaces de Software, operadores y usuarios.

4. Gerenciamiento

4.1. Organización

ROL	RESPONSABLE
Coordinador de Requerimientos	Ana Silva
Ingeniero de Requerimientos	Todos
Ingeniero de Procesos	Ana Silva
Arquitecto	Andrés Nieves

SQAer	Mauro Varela
SCMer	Clara Youdale
Administrador de tareas	Clara Youdale
Desarrollador	Todos
Tester	Todos

Ilustración 11-46 Roles y Responsabilidades

4.2. Responsabilidades [5]

A continuación se describen las actividades de cada rol:

1) Administrador de tareas es responsable de:

Se encarga de las tareas administrativas del proyecto. Elige a la persona indicada para realizar el plan de SQA para luego revisarlo y aprobarlo. Es a quien se debe reportar en caso de encontrar errores en el producto o proceso, el Administrador identifica y analiza cualquier problema de calidad que le sea reportado.

2) SQAer es responsable de:

Es el encargado en definir el plan de Calidad y de asegurar que se cumpla lo establecido en este. Todo Software producto del plan de calidad debe cumplir ciertas características y requisitos, es responsabilidad del SQAer, verificar que los posea. A lo largo del ciclo de vida del proyecto, deberá realizar revisiones, inspecciones y revisar las técnicas de los productos de Software elaborados.

3) Arquitecto

Se encargará de diseñar toda la arquitectura del sistema, es decir, tomará las decisiones estratégicas para aprovechar al máximo la tecnología. Decidirá cómo será construida la aplicación y creará todos los diagramas.

4) Diseño/desarrollo de Software son responsables de:

Debe implementar lo establecido en el plan de calidad. De esta forma podrá identificar y darle el seguimiento correspondiente a cualquier problema de calidad reportado por SQA, dentro del área de Diseño y Desarrollo del producto. Para lograr un producto de calidad implementará las prácticas, procesos y estándares de diseño y desarrollo especificado en el plan de SQA.

5) Integrantes que realizan *Testing*, son responsables de:

Se encargará de probar el sistema, que el mismo cumpla con lo establecido y reportar todas aquellas fallas que el sistema posea. Para esto implementará realizará casos de prueba,

siempre continuando con lo establecido en el plan de calidad (referido al testeo). Una vez hallado un problema, debe darle seguimiento hasta resolver el mismo.

6) SCMer es responsable de:

El SCMer se encarga de resolver los problemas de calidad relacionados con SCM. Deberá definir la línea base y asegurar el seguimiento de la misma. Controlará los cambios y planificará la configuración.

7) Análisis y requerimientos es responsable de:

Identificar, analizar y resolver los problemas de SQA relacionados con el análisis de requerimientos. De esta forma podrá asegurar la calidad en relación a los requerimientos.

8) Ingeniero de proceso es responsable de:

Administrar los recursos necesarios para asegurar la producción planeada conforme a los requerimientos del cliente. Responsable del diseño y la gestión de los procesos.

5. Estándares, prácticas y convenciones [6]

5.1. Propósito

El objetivo principal de esta sección es el de definir los estándares, prácticas y convenciones a cumplirse.
Los mismos pueden estar relacionados tanto para los documentos como para el código realizado, diseño, métricas y riesgos.

5.2. Estándares

Recopilando información que fue enseñada a lo largo de la carrera se elaboró una lista de estándares y buenas prácticas que se deben cumplir en cada una de las áreas del proyecto. Se recopiló información de autores que fueron presentados en materias como Ingeniería de Software o Gestión de proyectos.

5.2.1. Estándares de diseño:

· El desarrollo de Software se realizará en Java
· El nombre de las clases siempre comienza en mayúscula así como la segunda palabra si es que esta existe.
· El nombre de los métodos debe iniciar en minúscula y la segunda palabra debe de ser mayúscula.
· El nombre del método debe describir la función que este realiza.
· Los nombres de clases y métodos será en español.
· En cada diagrama habrá una nota especificando el o los requerimientos que satisface.

A su vez se basó en los principios de diseño para favorecer la mantenibilidad y escalabilidad del sistema. Entre estos se encuentran:

- Principio de responsabilidad única
- Principio de Abierto - Cerrado
- Principio de sustitución de Liskov
- Principio de Segregación de interfaz
- Principio de inversión de dependencia

5.2.2. Estándares de codificación

Para los estándares de codificación se basó en Google Java Style y se realizaron las modificaciones pertinentes para generar nuestro propio estándar para Java.

5.2.3. Estándares de documentación:

Para los estándares de documentación se aplicaron los definidos por los documentos de la ORT. Los mismos son 302, 303, 304, 306 y 307.

· Los documentos deben tener una tabla de versionado. En esta se proporciona información sobre cambios realizados, quién lo efectuó, cuándo y quién revisó dicho cambio.
· Deben estar en una letra común (Calibri, Arial, Time New Roman) y con un tamaño legible.
· Los documentos deben estar debidamente identificados y registrados en el ítem documentación de este plan.
· La documentación del proyecto debe estar en idioma español.
· En toda documentación debe estar identificado el nombre, quien lo desarrolla, quien lo prueba y la versión del documento.

5.2.4. Estándares SCM

· Se debe manejar siempre el versionado de los documentos pudiendo volver en cualquier momento a la versión anterior.

- Se utilizará *Google Drive* para mantener control de los cambios realizados en los documentos.
- Para el control de las versiones del código se utilizará la herramienta Github.
- Se deberá detallar el proceso de realización de un cambio en el sistema.

5.2.5. Estándar de documentos en *Google Drive*

Para mantener un formato coherente entre todos los documentos almacenados en Google Drive, los mismos deben seguir las siguientes reglas:

- Deben tener un título descriptivo de su contenido
- Deben tener una tabla llamada “Tabla de Revisión” conteniendo las columnas Fecha, Versión, Quién, Descripción y Revisión.
    - En la fecha se especifica la fecha en la cual el documento fue cambiado
    - La versión incrementa de un punto en cada modificación, empezando en 1.0 e incrementado el número después de la coma

    - En la columna “Quién”, debe aparecer el nombre de la persona que realizó la modificación
    - En la descripción debe aparecer el propósito de la modificación
    - En la columna “Revisión” se registra la fecha en la cual se revisó la modificación realizada por la persona que se encuentra en la columna “Quién”
- Deben contener un encabezado llamado Objetivo o Propósito y otro llamado Alcance (O un punto con los 3 encabezados) que describa lo que contiene el documento, cuál es su propósito y cuál es su alcance.

5.2.6. Estándar de requerimientos

Para mantener la coherencia entre los requerimientos, se definió un estándar que se implementa en el ESRE, para cada requerimiento.

5.2.7. Estándar de *Testing*

Para lograr estandarizar las pruebas de Software las mismas se basarán en el siguiente estándar. ISO/IEC/IEEE 29119 Software *Testing.*

Dicho estándar tiene como objetivo proporcionar una norma definitiva para las pruebas la cual define vocabulario, procesos, documentación, técnicas y un modelo de evaluación del proceso de pruebas de Software.

El mismo consta de cuatro partes:

1- Conceptos y vocabulario: donde se da una visión general de la norma y los conceptos generales de pruebas de Software. Además de proporcionar un vocabulario de términos para las pruebas.
2- Proceso de prueba: En la norma se define un proceso de prueba genérico que se puede utilizar en cualquier desarrollo de Software y ciclo de vida de la prueba.
3- Documentos de Prueba: En esta parte se cubrirá documentación de pruebas en todo el ciclo de vida del Software como ser plantillas.
4- Técnicas de ensayo: en la misma se cubre una variedad de técnicas dinámicas de pruebas de Software.

6. Pruebas

El propósito de realizar pruebas es el de encontrar fallas, corregirlas de forma anticipada para que las mismas no se conviertan en errores. Las mismas se detallan en el Anexo de *Testing* - Plan de Pruebas.

7. Atributos de calidad

Los atributos de calidad seleccionados se ven especificados en el Anexo de Arquitectura.

8. Métricas en SQA

Con el objetivo de evaluar la calidad del proceso y producto se definen métricas. Mediante la realización de mediciones y se registraron indicadores los cuales lograron definir si se cumplen con las especificaciones necesarias.
Se denomina indicador, a una métrica o combinación de métricas que proporcionan una visión profunda del proceso, proyecto o producto de Software.
Las mismas se detallan en el Anexo de Registro de las Métricas.

9. Revisiones

Se planteó la existencia de dos tipos de revisiones durante el proyecto. Las mismas se dividen en las revisiones internas al proyecto y las externas. Las revisiones externas son tres y se realizan por un profesor de la Universidad, el cual luego de presenciar una muestra de los avances, aconseja y asiste brindando posibles mejoras desde sus conocimientos y experiencia.

Evaluación	Modo	Revisor	Resultado	Fecha
1	Reunión	Nicolás Fornaro	Informe	01-jul-14
2	Reunión	Marcelo Cagnani	Informe	09-set-14
3	Reunión	Rafael Bentancur	Informe	16-dic-14

Ilustración 11-47 Revisiones

Por otro lado las reuniones internas serán establecidas internamente por el equipo, donde se realizarán preguntas y se evaluará internamente los errores, violaciones de los estándares establecidos y otros inconvenientes. Las mismas se realizarán cada dos semanas en cada *Sprint Review*.

En cada revisión se tendrán tres objetivos primarios. Estos serán:
- Detectar defectos.
- Elegir el camino de resolución.
- Verificar la resolución (los defectos deben ser resueltos).

Las inspecciones no sólo son útiles para corregir defectos sino también para potenciar el trabajo en equipo y obtener datos para las métricas.
En cada reunión de inspección habrá:
- Revisores: buscan defectos y toman decisiones.
- Moderador: responsable de la inspección.
- Registrador: anota.

Para los defectos hallados se deberá informar:
- Ubicación.
- descripción
- Gravedad: Mayor / Menor
-

10. Reuniones con expertos

Debido a que el proyecto tiene como temática central un área en la cual se carece de conocimiento, se decidió realizar reuniones con expertos del dominio. Varias reuniones fueron necesarias mantener, para lograr un producto de calidad.

Reunión con Ing. Agrónomo Mario Pérez: experto del dominio, catedrático de la facultad de Agronomía. Solamente se pudo concretar una reunión pero fue muy enriquecedora debido a los amplios conocimientos del experto. Mario es una de las personas con mayor conocimiento de la agronomía en el país, no sólo brindó conocimiento sino que también aportó su punto de vista respecto a la necesidad de una herramienta como SIGAMAS. Dichas palabras fueron muy motivadoras ya que auspicia una herramienta de futuro exitoso.

Reunión con Ing. Agrónomo Tabaré Alcorta: fueron realizadas reuniones periódicas. Las frecuencias de las mismas fueron bajando a medida que avanzaba el proyecto y los conocimientos del equipo crecían. Fue la persona que más aportó conocimientos en un campo desconocido. El compromiso de Tabaré con el equipo y el proyecto fue fundamental, al ser un campo en el que se carecía de conocimiento se necesitaba contar con una persona que tuviera disponibilidad constante para ampliar conocimiento y evacuar dudas, esta persona fue Tabaré.

Reunión con Pablo Montes: Coordinador de la Cartera Nacional de Proyectos Ambientales en la Dirección Nacional de Medio Ambiente, a lo largo del proyecto fue muy importante contar con el apoyo de Pablo. Fue quién informaba de las actividades importantes que se realizarían, como ser el evento de Suelos o generaba los contactos con personas claves del Ministerio MGAP. Las reuniones con los principales representantes de las entidades estatales, fueron gracias a Pablo. También aportó conocimiento de dónde se encuentra la información sobre la situación actual de los campos del Uruguay o información sobre las fases de los cultivos.´

Participación en el 7mo. Congreso Uruguayo de Suelos: en dicho evento se tuvo una participación activa presentando frente a grandes autoridades e importantes productores las principales características del proyecto. Se obtuvo un *Feedback* muy importante y de acuerdo a este se realizaron modificaciones en el proyecto. Nuevamente fue muy motivante el importante número de personas que se acercaron para informarse sobre el proyecto y que mostraron su interés de contar con la herramienta. Se aprovechó la oportunidad para realizar encuestas entre los interesados, analizados los resultados, se realizaron pequeños cambios.

Reunión con FUCREA: esta entidad es “La Federación Uruguaya de los Grupos Crea (FUCREA) es la organización que nuclea a todos los Grupos CREA que desarrollan sus tareas dentro del territorio nacional. FUCREA es una Institución de larga trayectoria en el sector agropecuario, que desde 1966 cuenta con gran reconocimiento a nivel nacional e internacional.” Autoridades de la entidad acudieron al equipo para informarse más sobre el proyecto, luego de tener noticias del mismo en el Congreso de Suelos. Contactaron al equipo con expertos del dominio (Ingenieros Agrónomos) los cuales brindaron más información sobre los planes de uso de suelos. Se planteó la posibilidad de poder trabajar con ellos una vez el proyecto haya concluido.

Reunión con Mónica Barbazán Ing. Agrónoma del departamento de suelos y aguas de la facultad de Agronomía. Mónica fue la encargada y anfitriona del Congreso de Suelos. Le otorgó al equipo la posibilidad de presentar el proyecto en el Congreso frente a todos los presentes. Posteriormente organizó una reunión con la Mesa Tecnológica de Oleaginosos y con Mariana Hill Directora General de RENARE.

Reunión con Mesa Tecnológica de Oleaginosos. Es una iniciativa pública privada que agrupa empresas productoras, industriales y comercializadoras de oleaginosos, así como instituciones de tecnología e investigación. Gracias a esta reunión se coloca al proyecto en conocimiento de las empresas del rubro. Es muy importante ya las empresas serán unos de los canales de distribución del servicio. Desde un principio se notaron muy interesados.

Reunión con Nicolás Fornaro para discutir de la arquitectura del sistema. Dado el conocido *expertise* de Nicolás en arquitectura, se acudió a él para obtener consejos y evacuar dudas. El arquitecto consideró fundamental dicha reunión con Nicolás.

Reunión CIE (Centro de Innovación y Emprendimientos) con Rosana Fernández. Se acudió a Rosana, coordinadora de proyectos para que obtener consejos en los primero pasos y cómo presentar el mismo.

Reunión con Mariana Hill Ing. Agrónoma Directora General RENARE (Dirección General de Recursos Natural Renovables) en MGAP. Mariana se mostró interesada en el proyecto luego de hablar con ella en el Congreso de Suelos en Colonia. Se concretó una reunión con ella en el Ministerio donde se planteó el beneficio que tiene para el estado la utilización por parte de los productores de dicha herramienta.

REFERENCIAS BIBLIOGRÁFICAS

[1] IEEE 2008 Std. 1028-2008. [En linea]. Disponible: http://segoldmine.ppi-int.com/content/standard-ieee-std-1028-ieee-standard-Software-reviews-and-audits

[2] IEEE Std 730.1-1995, IEEE Guide for Software Quality Assurance Planning. [En linea]. Disponible: http://standards.ieee.org/findstds/standard/730.1-1995.html

[3] IEEE Std 1012-1998, IEEE Standard for Software Verifcation and Validation. [En linea]. Disponible: http://standards.ieee.org/findstds/standard/1012-1998.html

[4] A. Álvarez, V. Filipovich. 2010. Plantilla del Plan de SQA. ORTsf

[5] Responsable de SCM. [En línea]. Disponible: http://www.fing.edu.uy/inco/cursos/ingsoft/pis/proceso/MUM/roles/respscm.htm

[6] Proyecto: Zeugnis - Universidad ORT Uruguay

11.22. PLAN DE SCM

Fecha	Versión	Quién	Descripción	Revisión
01/06/2014	1.0	Clara	Creación del documento Plan SCM	07/06/2014
07/07/2014	1.1	Clara	Actualización del Plan SCM	17/07/2014
15/07/2014	1.2	Clara	Actualización del Plan SCM	31/07/2014
15/08/2014	1.3	Clara	Actualización del Plan SCM	19/08/2014
24/08/2014	1.4	Clara	Actualización del Plan SCM	28/08/2014
20/11/2014	1.5	Clara	Actualización del Plan SCM	21/11/2014
12/12/2014	1.6	Clara	Actualización del Plan SCM	15/12/2014
03/01/2015	1.7	Clara	Actualización del Plan SCM	06/01/2015

Propósito

El propósito de la gestión de la configuración (SCM) es de establecer y mantener la consistencia de las piezas del Software a lo largo de su ciclo de vida. Se aplican procedimientos técnicos y administrativos para identificar, definir las piezas de Software; controlar modificaciones y versiones de estas piezas; registrar y reportar el estado de cada pieza y las solicitudes de modificaciones.

Objetivos

Establecer una línea base para cada proceso del ciclo de vida del Software.
Realizar un seguimiento para tener más control sobre los cambios.
Mantener y controlar la línea base a lo largo del ciclo de vida.

Roles

Solo se tiene un rol, el SCMer que se encarga de mantener las distintas versiones, controlar y verificar la autenticidad de las mismas. Sin embargo, todos los integrantes son responsables de conocer y aplicar el plan SCM elaborado.

Herramientas y repositorio

Para llevar a cabo el proceso SCM se debe hacer uso de varias herramientas para las distintas facetas del mismo.

- ***Un repositorio de documentos***

Para tener todos los documentos y archivos necesarios para la realización del proceso SCM se debe hacer uso de un repositorio que permita mantener las versiones y tener un seguimiento de los cambios realizados en los distintos documentos. Los documentos y carpetas deben seguir una estructura particular que se define a continuación:

Ilustración 11-48 Organización de las carpetas

Documentos de apoyo:

Allí se encontrarán todos los documentos, encuestas que no hacen parte del proceso pero que sirven para su funcionamiento. Es decir, todos los documentos que influyen indirectamente en el proceso.

Entregas:

Se encontrarán las entregas que se fueron realizando a lo largo del proyecto para mantener las distintas versiones del producto, para poder observar la evolución del mismo.

Etapa 1 – Planificación

Se encontrarán los documentos relacionados con los requerimientos (prototipos y descripciones), el cumplimiento de los mismos y otros documentos que apoyen a la planificación.

Etapa 2 – Diseño

Se encontrarán los diagramas y documentos que estén relacionados con el diseño del producto.

Etapa 3 – Desarrollo

Se encontrarán los documentos que apoyen al desarrollo del producto.

Etapa 4 - Revisión y prueba

Se encontrarán los documentos que permiten realizar las distintas pruebas sobre el producto, el plan de prueba y los resultados de las mismas.

Negocio

Se encontrarán los documentos que se relacionan con el negocio, es decir el CANVAS, y otros documentos vinculados con el futuro de SIGAMAS.

Resumen y estadísticas

Se encontrarán los documentos que recolectan la información que se generó en cada *Sprint* y la acumulación de las mismas que permitirán luego generar las métricas.

SQA

Se encontrarán las actividades SQA del proceso así como las revisiones y los estándares.

- ***GitHub***

Para el control de las versiones del código del producto así como otros elementos necesarios para la construcción del producto, se usará el repositorio GitHub ya que permite un control rápido, eficiente y fácil de manejar. Además permite el seguimiento y reporte de errores y un manejo conciso de los *Branches*.

Identificación de los elementos de configuración de Software (ECS)

Los elementos considerados para la gestión de la configuración se separaran en las siguientes categorías:

- Códigos fuente
    - Comprende el código de la aplicación
        - Elementos web (xhtml, html, css, js)
        - .java
- Datos
    - Es lo que comprende los datos y la base

- Ejecutables
- Documentación
    - Comprende todos los documentos que realizados durante el ciclo de vida del proyecto
        - Diagramas
        - Documentos web
        - Imágenes
        - Documentos

Proceso SCM - Petición de Cambio

A continuación se describirá el proceso que debe ejecutarse al surgir una petición de cambio.

Una petición de cambio puede afectar los diferentes ECs definidos anteriormente. Se entiende por petición de cambio a las siguientes posibles modificaciones:

- Cambios en los requerimientos
- Cambios en la Arquitectura
- Cambios en el Diseño
- Cambios en el Código
- Cambios en los Documentos del proyecto

A continuación se detalla el proceso de petición de cambio:

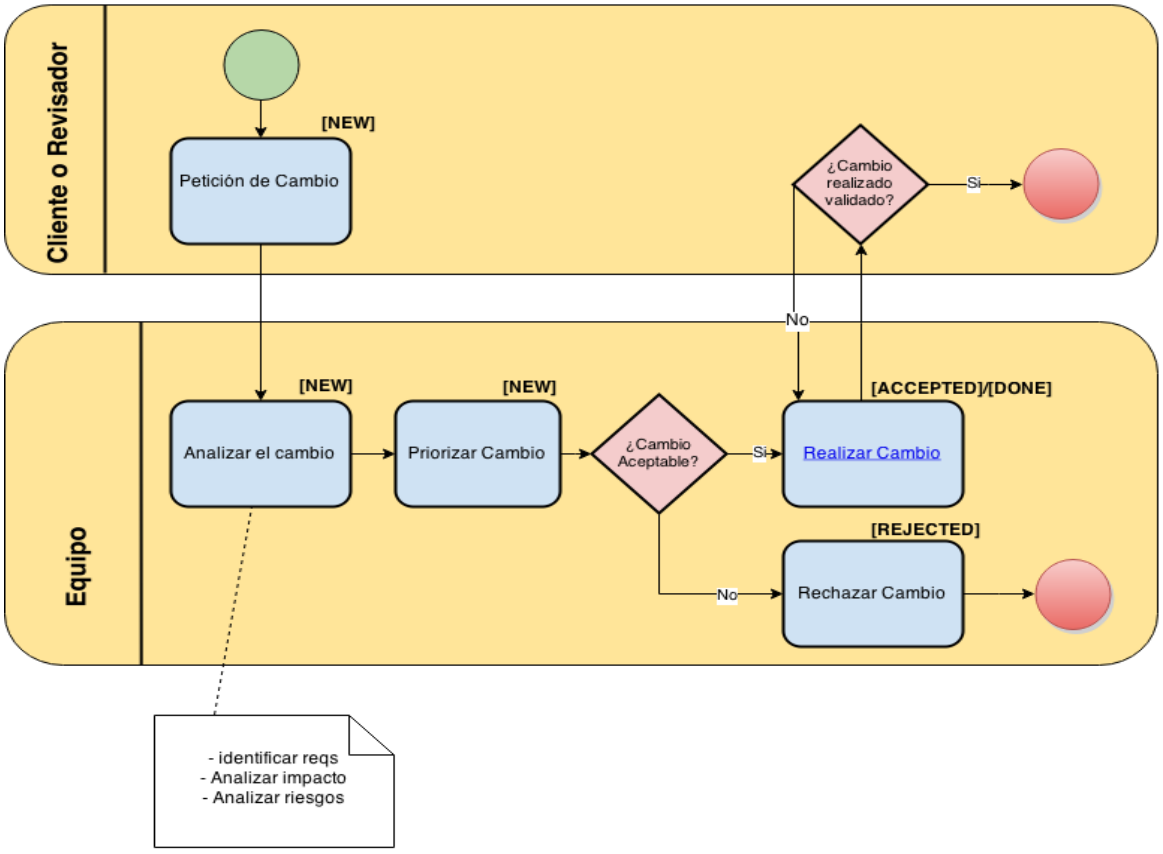


Ilustración 11-49 Proceso general de control de cambios

Petición del cambio

Sucede cuando el cliente o revisor sugiere que se realice una modificación. En esta etapa, se crea la petición de cambio y se le asigna el estado [*NEW]* (nuevo). Se decidió nombrar los estados de los cambios en inglés, porque en GitHub los estados de los *Issues* están en inglés y se quería unificar ambos procesos.

Por una cuestión de organización, se solicita que el título de las solicitudes de cambio tenga el formato:

Sprint (Si corresponde) - Priorización - Fecha - Nombre - Descripción – Estado

Analizar el cambio

En esta actividad, se analiza los requerimientos que el cambio afecta, el impacto que tiene sobre la solución y los riesgos de realizar este cambio. Todavía no se le atribuye una priorización y no se le cambia de estado.

Priorizar el cambio

En esta actividad, a partir del análisis anterior, se le atribuye una prioridad al cambio y

además un nuevo estado. Si el cambio tiene prioridad No Urgente y No Importante, se le asigna el estado [*REJECTED*], en otro caso con el estado [*ACCEPTED*].

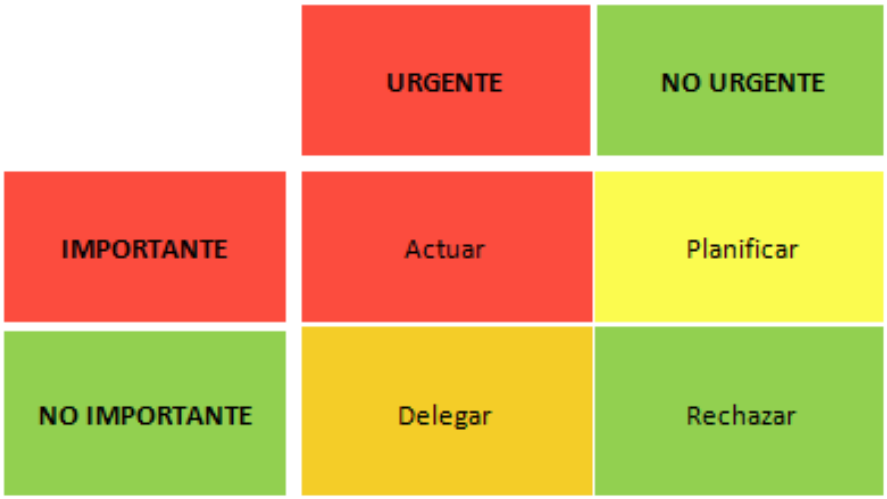


Ilustración 11-50 Actuación sobre Cambios

Cambio [Urgente, Importante] – Actuar

Los cambios urgentes e importantes se resolverán lo antes posible. En la planilla "Control de Cambios" estos cambios se posicionarán al principio de la lista, con máxima prioridad.

Cambio [Urgente, No importante] – Delegar

Los cambios urgentes pero no importantes, no necesitarán ser resueltos con la máxima prioridad. Se les atribuirá la acción "Delegar" que significa realizarlos después del último cambio Urgente e Importante.

Cambio [No Urgente, Importante] – Planificar

Los cambios no urgentes pero importantes, no necesitarán ser resueltos rápidamente, pero sí con precaución. Al ser importantes, el impacto en el sistema será alto, por lo que será necesario una buena planificación.

Cambio [No Urgente, No Importante] – Rechazar

Los cambios ni urgentes ni importantes, serán rechazados por el equipo ya que se considera retrasarán el cronograma innecesariamente.

Realizar el cambio

Esta etapa corresponde al sub proceso SCM del proceso principal

Rechazar el cambio

En caso que el cambio sea rechazado (estado [*REJECTED*]) se terminará el proceso de control

de cambio y se guardará registro del cambio en la planilla cómo rechazado. Luego de que la solicitud sea resuelta se cerrará.

Manejo de *Branches*

Al surgir un cambio que resulta en modificaciones en el código, esto implica un manejo especial de *Branches* en el repositorio de GitHub de SIGAMAS. El proceso es el siguiente:

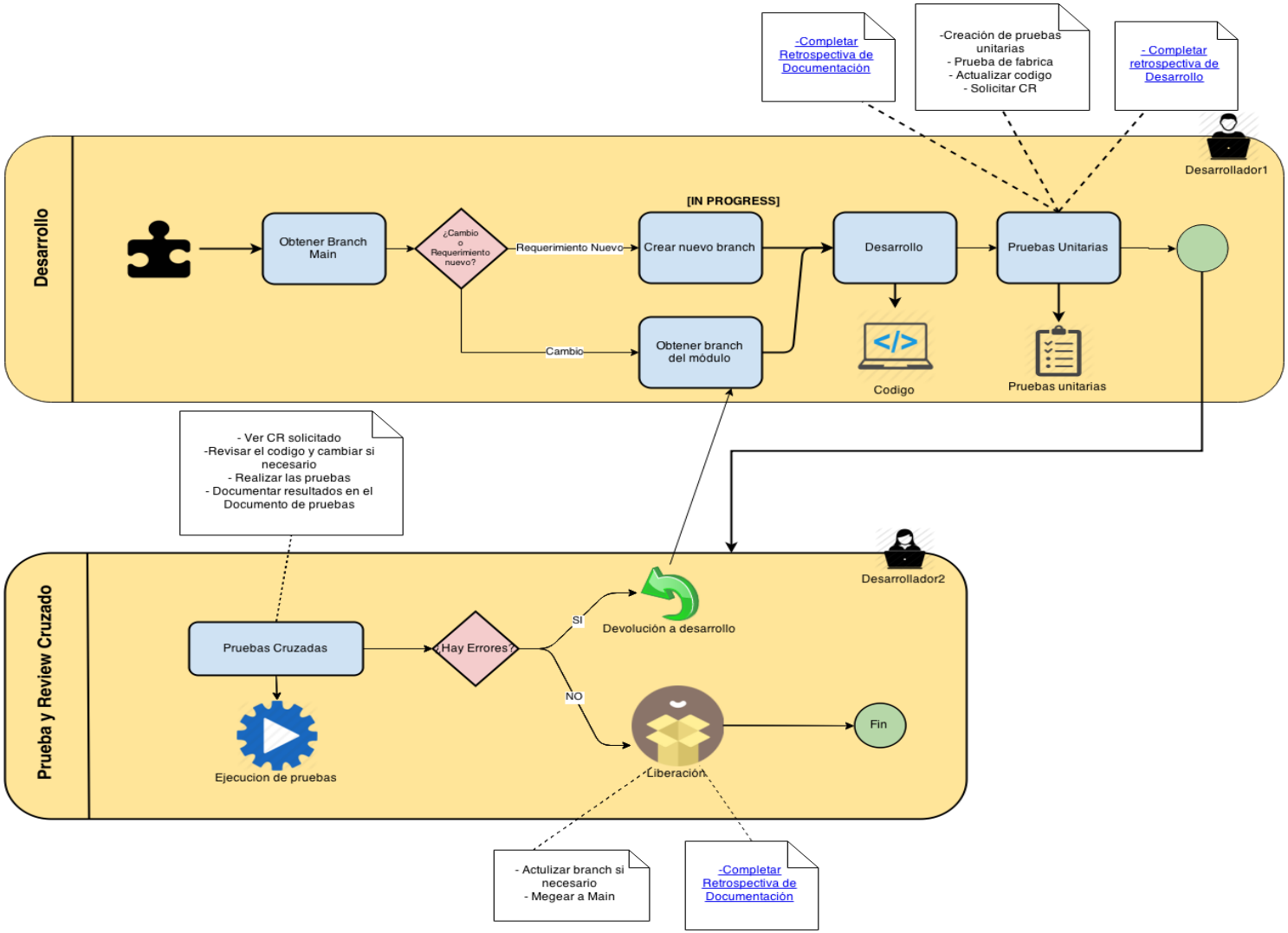


Ilustración 11-51 Manejo de *Branches*

Para ejemplificar el proceso recién mencionado se adjunta un ejemplo de línea de tiempo de manejo de *Branches*.

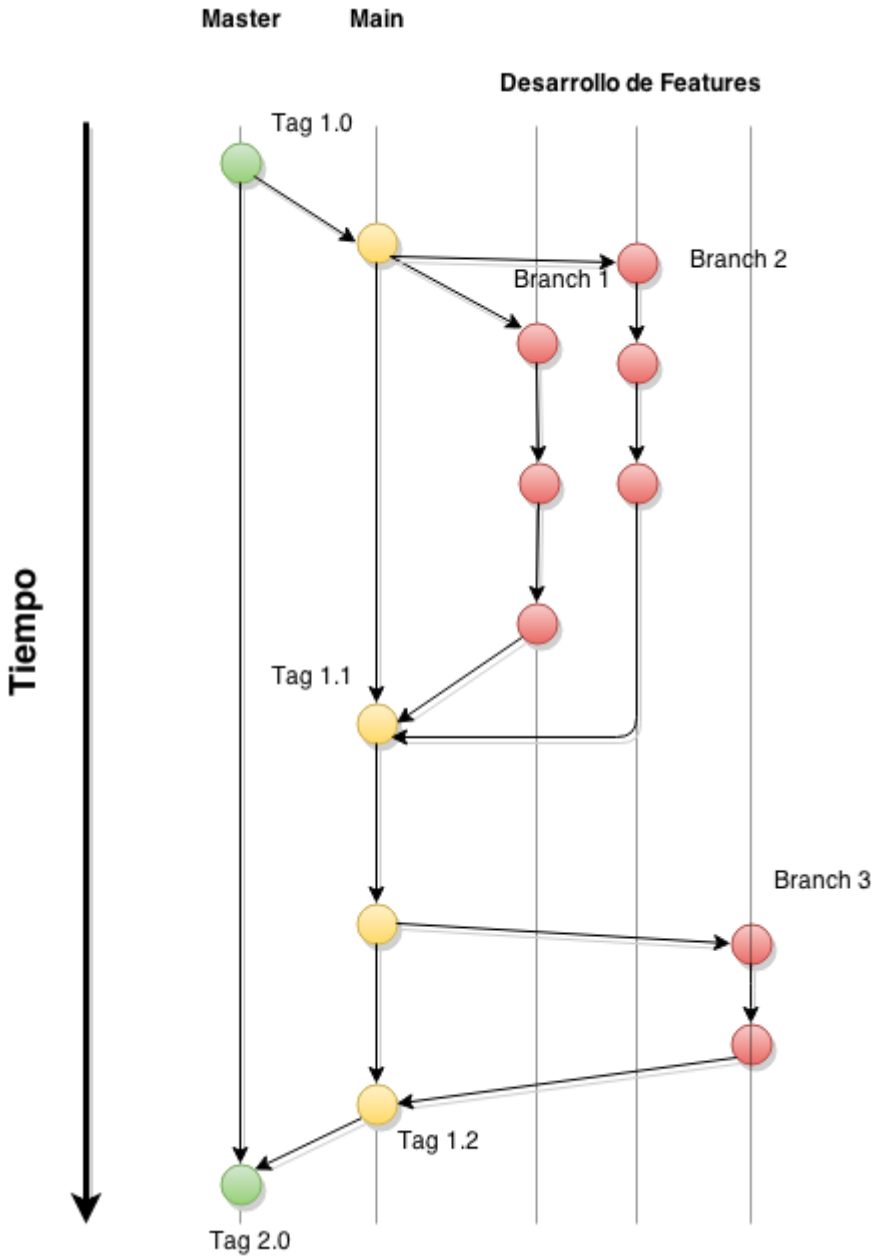


Ilustración 11-52 Línea de tiempo de Manejo de *Branches*

Todos los cambios que se pedirán por parte de los posibles usuarios, expertos del dominio o de cualquier miembro del equipo deben registrarse en planillas *Google Sheet*. Existirá una planilla para el Control de Cambios en Requerimientos, para cambios en código, para cambios en diseño y para cambios en los documentos.

11.23. SVN vs GIT

En este Anexo se desea explicar las diferencias entre SVN (*Subversion*) y GIT y porqué se eligió GIT.

Primero se describirán cada una de estas herramientas y luego se listarán las ventajas y desventajas de cada una.

SVN [1]:

Es una herramienta que permite el control de versiones y es de código libre. Permite controlar las versiones de archivos, pudiendo acceder a los mismos a través de la red. SVN es ampliamente usado en el mundo del Software libre, sobre todo para los lenguajes de licencia gratuita cómo lo es Java.

GIT [2]:

Git es un sistema de control de versión de código libre. Permite un manejo de archivos de forma rápida y segura. También es muy utilizado en el mundo del Software libre y los lenguajes de licencia gratuita cómo Java.

SVN vs GIT

Cómo se puede ver por las definiciones, las prestaciones de ambas herramientas son muy similares. Sin embargo, si se analizan en detalle, se pueden encontrar notables diferencias.

Ventajas de usar SVN:

- Fácil de usar, el IDE Netbeans tiene todos los comandos necesarios para realizar el manejo de *Branches* y versiones.
- Existen más herramientas de gestión (clientes) SVN que para GIT.
- SVN solo permite un repositorio lo que no genera confusiones a la hora de saber dónde se encuentran los archivos.
- SVN permite *checkouts* parciales (no se necesita bajar todo el código, solo la revisión que se precisa)
- Permite realizar *commits* a varios *Branches*

Ventajas de usar GIT:

- Sistema distribuido. Cuando un usuario desea utilizar el código publicado, GIT le crea una copia local para que pueda trabajar sobre ella. Esto permite tener el historial sobre todo lo que sucede en la copia sin tener que pasar por el repositorio central cómo se necesita con SVN.
- Al ser un sistema distribuido, cada usuario puede cambiar, realizar *commits*, sobre su repositorio local, sin afectar el repositorio central. Los usuarios mismos deciden cuándo realizar el *merge* con el repositorio central.
- GIT ofrece un excelente y robusto manejo de *Branches*.
- Los comandos GIT se ejecutan más rápidos que los de SVN.

- Los repositorios locales con GIT ocupan menos espacio en disco que si fueran administrados con SVN (debido a las carpetas de Git que ocupan menos espacio que las carpetas generadas por SVN).
- Se pueden ignorar archivos gracias al .gitignore. Dentro del mismo, se escriben los *paths* de los archivos que se desean ignorar. Esto es muy útil, ya que muchos archivos de configuración se generan al momento de compilar la solución y el *commit* de los mismos es innecesario.

Desventajas de usar SVN

- La ejecución de sus comandos es más lenta.
- El procedimiento de renombrar archivos no es completo. Se realiza como una copia del archivo a renombrar y luego el borrado del mismo. Esto puede generar confusiones en los reportes y el trazado histórico del archivo.
- Los archivos con extensión .svn son muy susceptibles a corromperse por errores del usuario.

Desventajas de usar GIT

- El IDE de Netbeans también tiene los comandos necesarios para realizar el manejo de *Branches* y versiones, sin embargo para muchos comandos se necesita utilizar la consola, lo cual puede agregar complejidad al manejo de archivos. Esto se da generalmente a la hora de realizar un *merge*, para los cuales se necesita resolver conflictos mediante líneas de comando.
- La curva de aprendizaje de GIT no es lineal. Es una herramienta más compleja.

Después de analizar las ventajas y desventajas de cada de las herramientas, se decidió optar por GIT. A pesar de ser una herramienta más compleja y difícil de aprender, era una herramienta que el equipo nunca utilizado a lo largo de la carrera. Uno de los objetivos del equipo era aprender nuevas tecnologías. Se consideró entonces que aprender a utilizar GIT era enriquecedor para el equipo mismo si esto presentaba mayores dificultades y tiempo de aprendizaje.

Se decidió entonces apartar tiempo para aprender esta nueva herramienta y elaborar un proceso para el manejo de *Branches* y *merges*.

REFERENCIAS BIBLIOGRÁFICAS

[1] Wikipedia, “Subversion”, marzo 2015. [En línea]. Disponible: http://es.wikipedia.org/wiki/Subversi%C3%B3n

[2] Git. [En línea]. Disponible: http://git-scm.com/

11.24. MANUAL NETBEANS PARA EJECUTAR EL PLAN SCM

Bajar la versión "*main*" del repositorio

Seguir los siguientes pasos:

1. **Clonar solución**

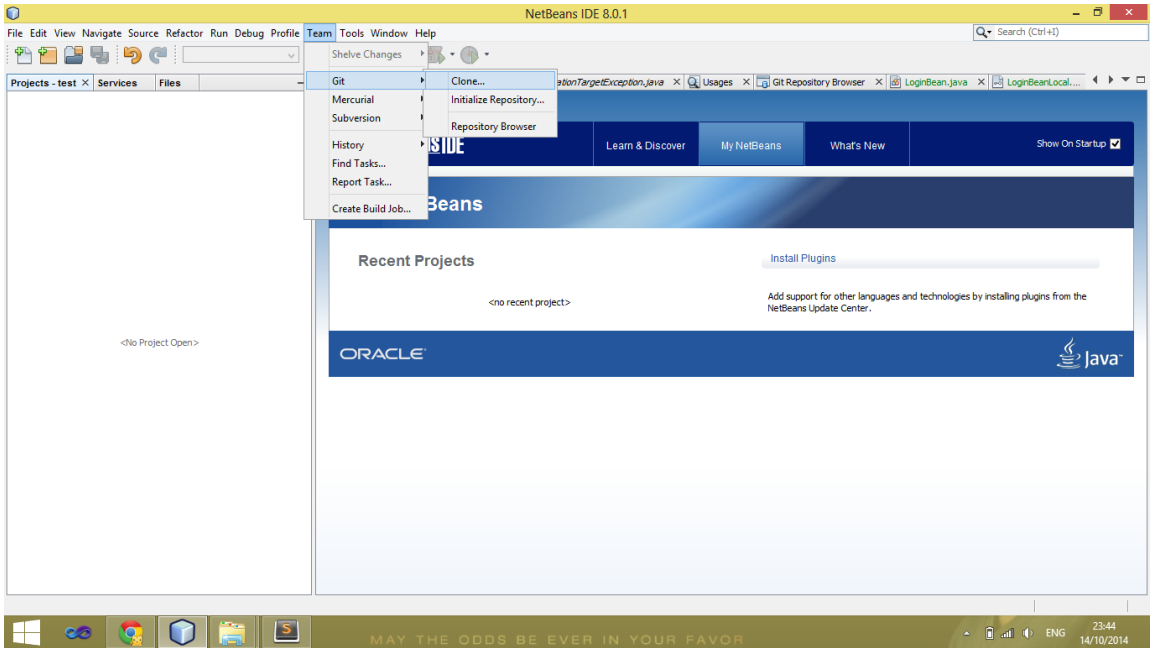


Ilustración 11-53 Clonar la solución

Datos:
Repository URL: http://github.com/clari182/SIGAMAS.git
(Se necesita un usuario en www.github.com)
Elegir la carpeta a donde bajar la solución y seleccionar *next*

2. Datos de clonación

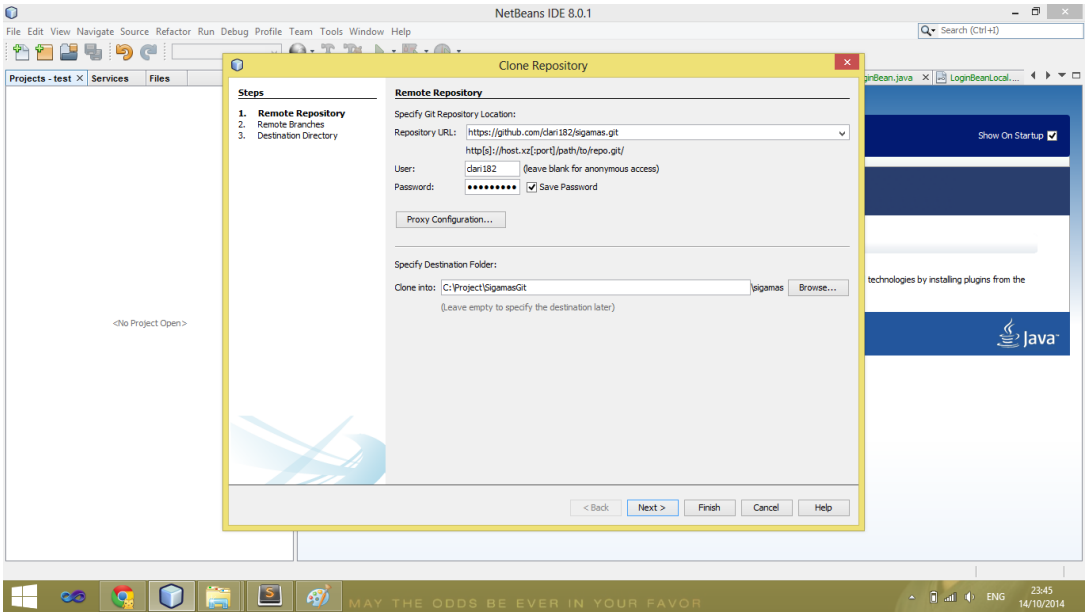


Ilustración 11-54 Ingresar datos para la clonación

3. Selección de *Branches*

Se seleccionan todos los *Branch* que hay y seleccionar *next next* y luego *finish*

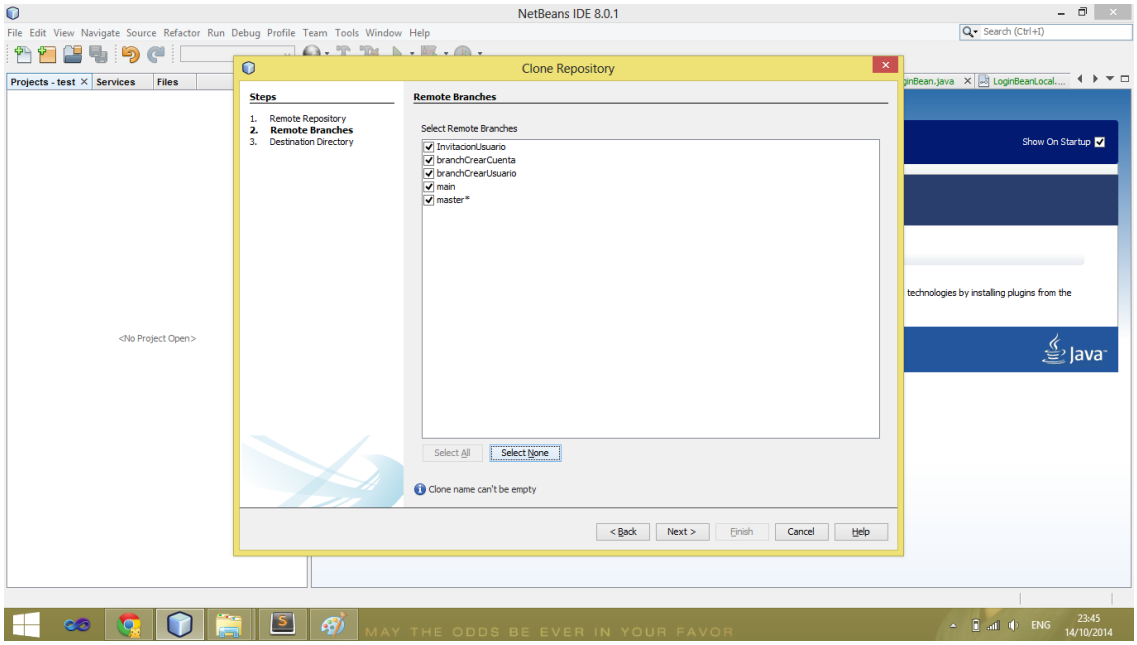


Ilustración 11-55 Seleccionar *Branches*

4. Seleccionar el proyecto

Una vez que se bajaron todos los archivos se van a abrir los proyectos. Seleccionar el proyecto SIGAMAS, luego open *required* y seleccionar *open*

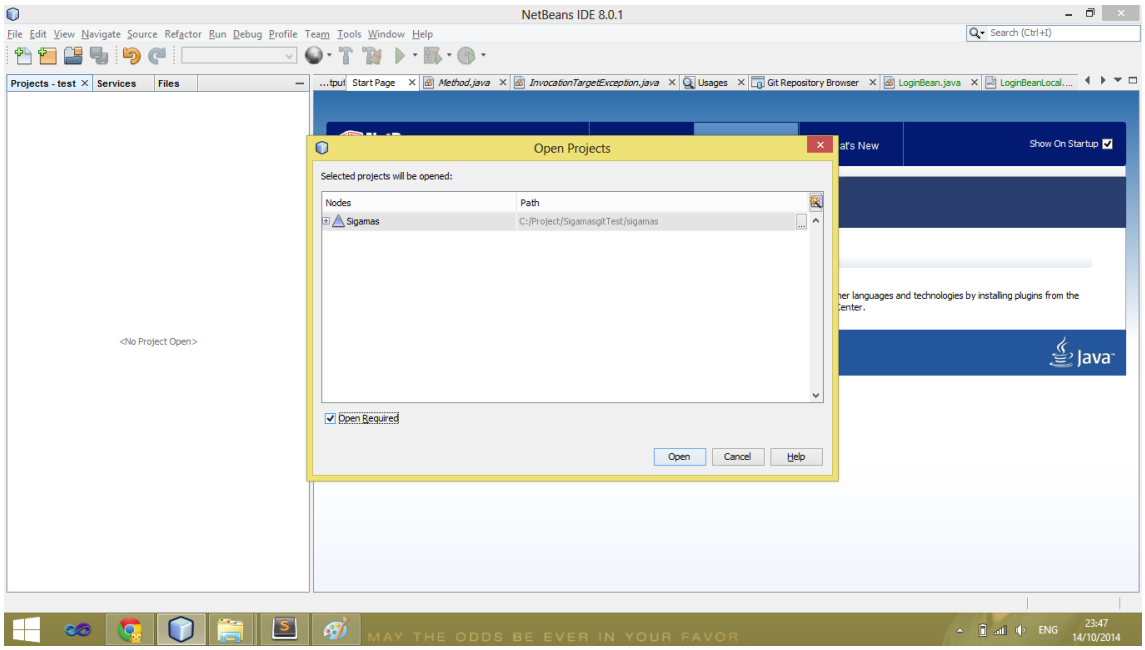


Ilustración 11-56 Seleccionar Proyecto

5. Selección del *Branch* "*Main*"

Luego se debe cambiar de *Branch* y pasarse al *Branch Main*. Seleccionar el proyecto SIGAMAS → *Git* →*branch/tag* → *switchToBranch*

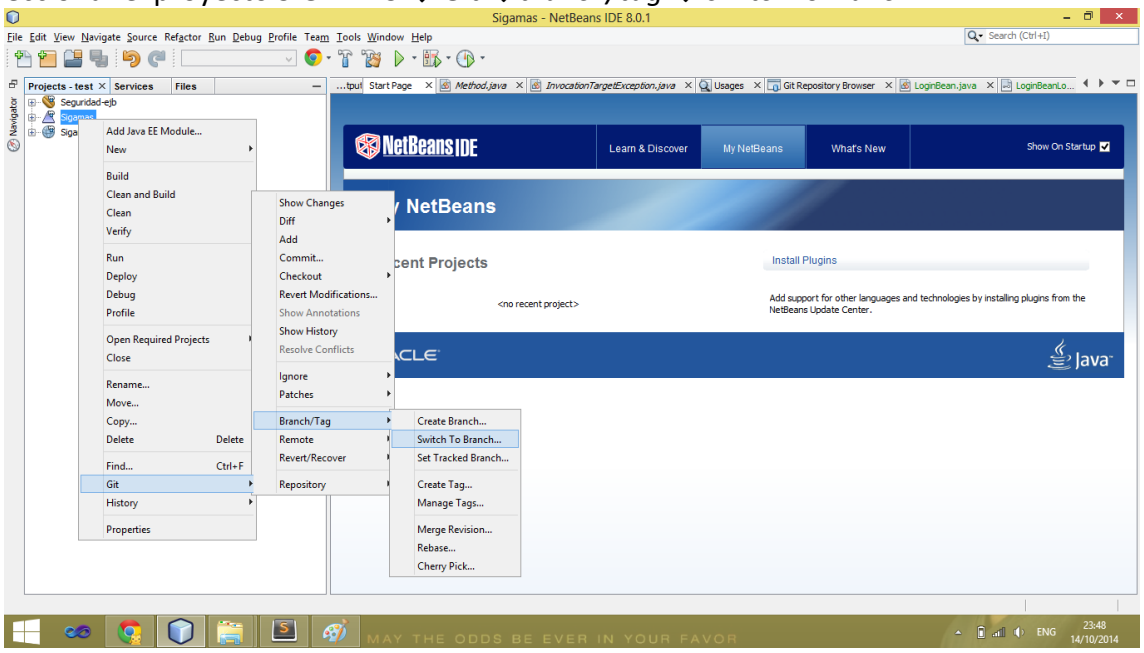


Ilustración 11-57 Seleccionar *Branch Main*

6. **Hacer *Checkout* de "*Main*"**

Aparecerá la lista de *Branches*. Seleccionar *origin/main*, y seleccionar "*checkout on new Branch*" y finalmente, seleccionar *switch*.

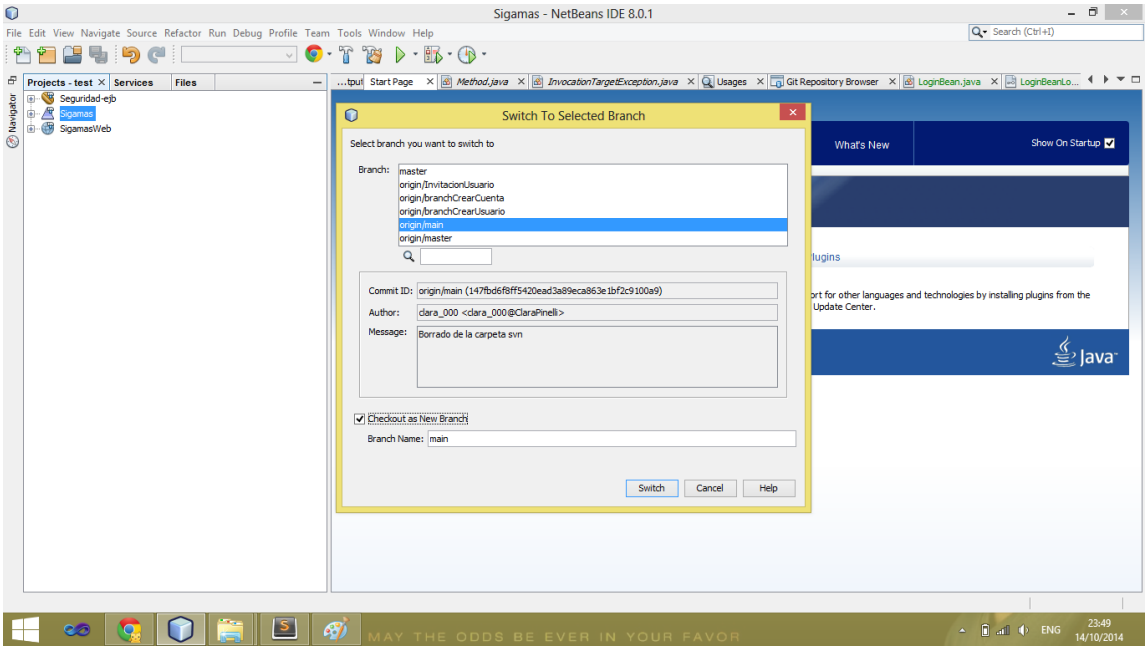


Ilustración 11-58 Checkout del *Branch Main*

Cómo crear un nuevo *Branch*

1. **Crear *Branch***

Git → Branch/Tag → Create Branch

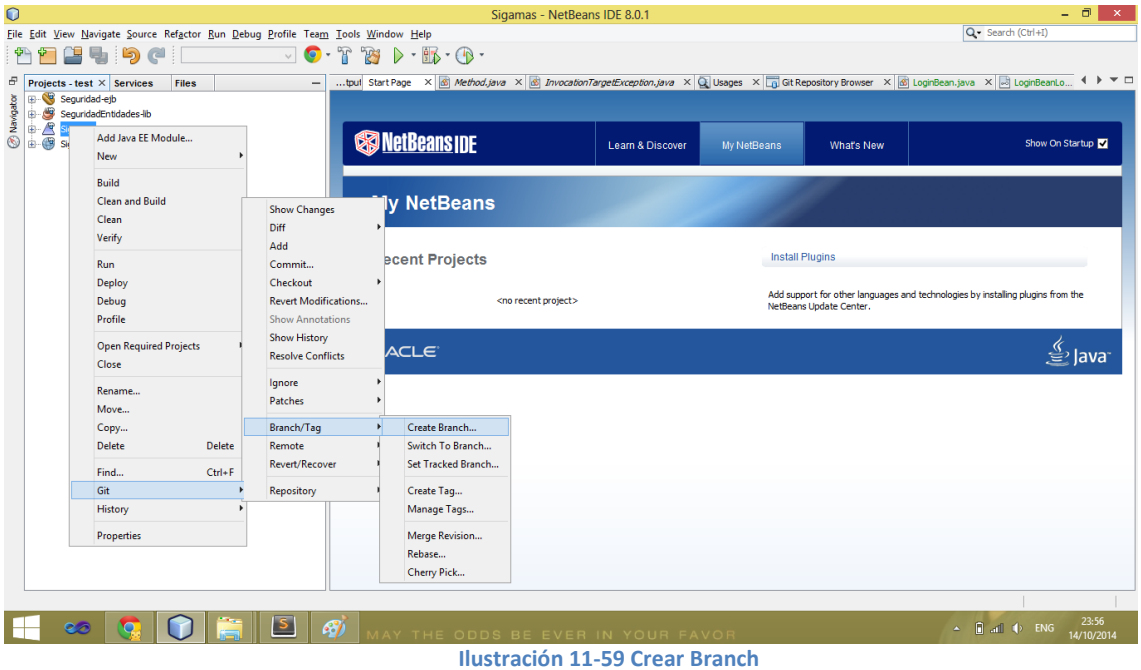


Ilustración 11-59 Crear Branch

2. Ingresar información del *Branch*

Branch name: ingresar el nombre para el nuevo *Branch*
Seleccionar *Checkout Commit Branch*

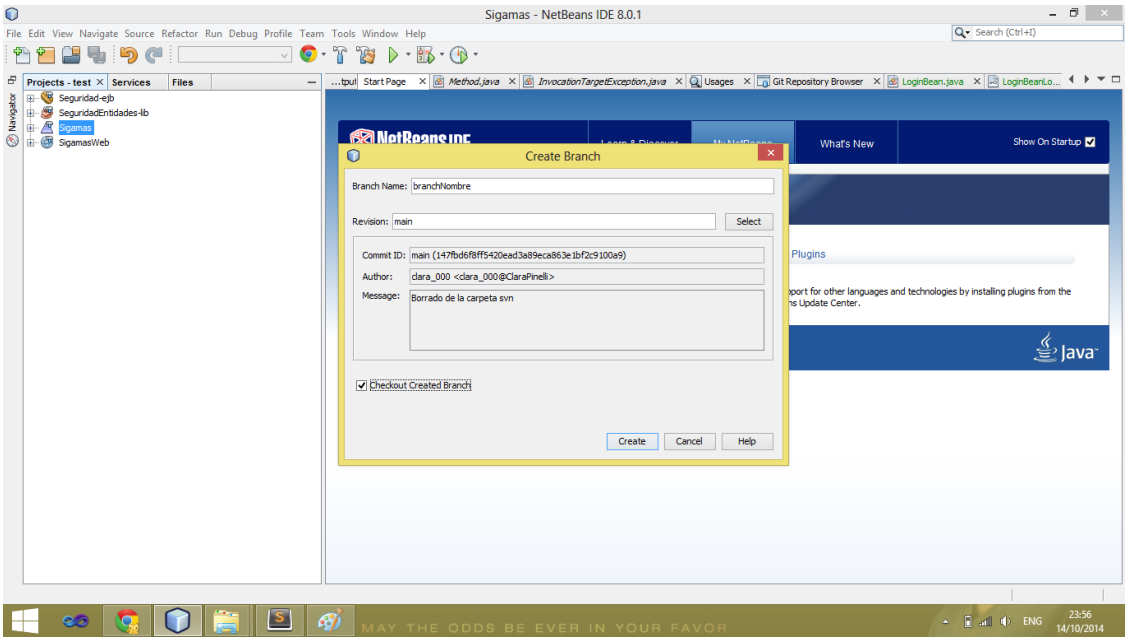


Ilustración 11-60 Ingresar información del *Branch*

3. Realizar *commit*

- Si surge un error de *merge*, seleccionar *revert changes*
- Realizar cambios.
- Subir los cambios
- *Team → Add*
- *Team → Commit*

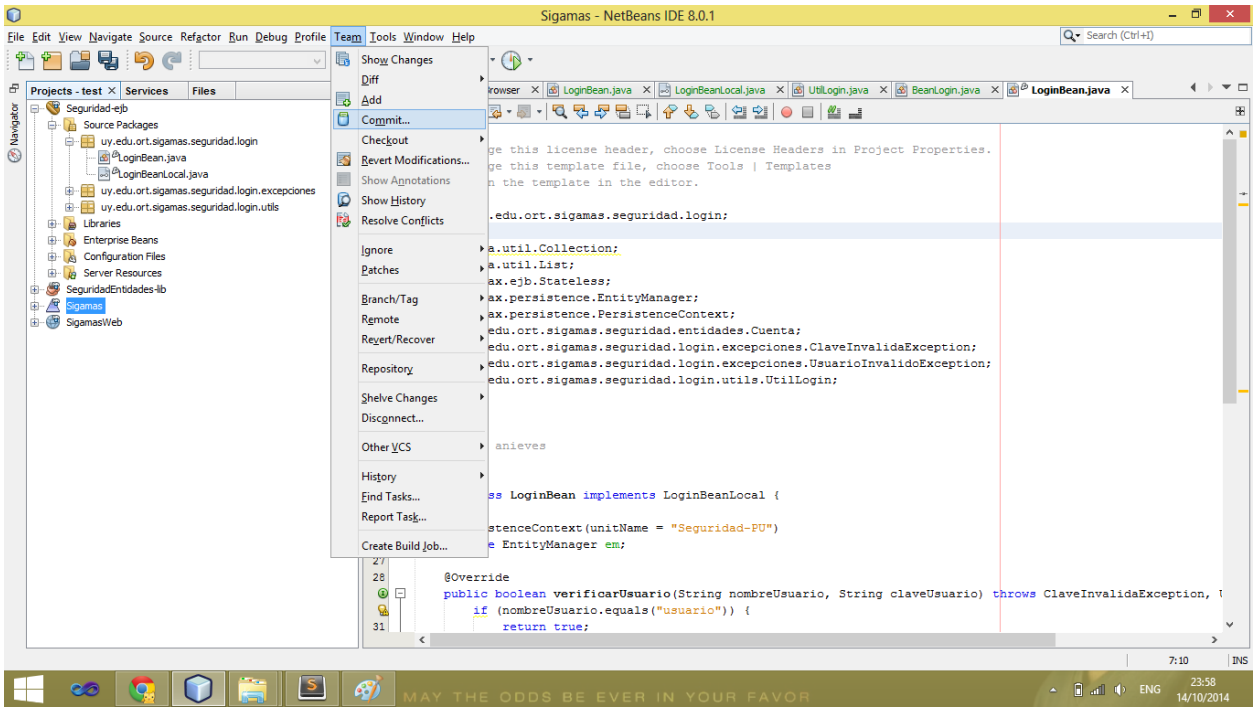


Ilustración 11-61 *Realizar commit*

4. Realizar *push*

Team → Remote → Push

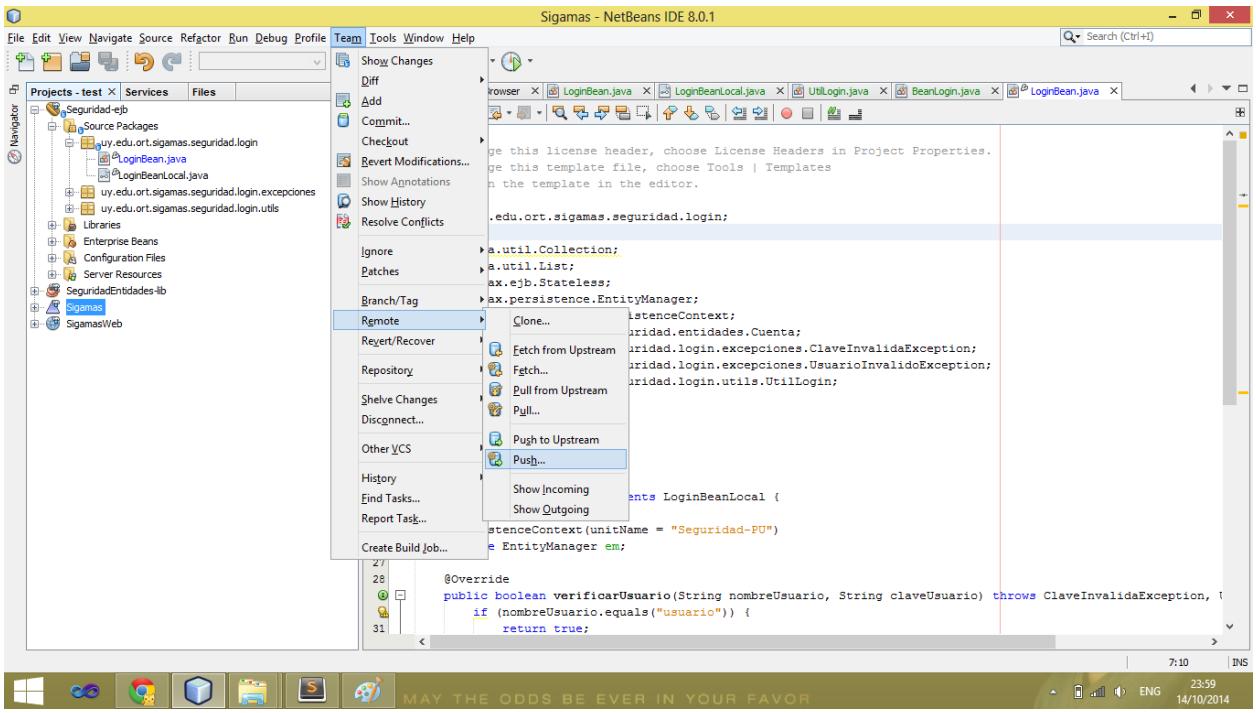


Ilustración 11-62 Realizar *Push*

11.25. BITÁCORA DE CAMBIOS DEL PROYECTO

A continuación se detallan dichas planillas.

Priorización	Fecha	Nombre	Descripción	*Status*
Actuar	12/8/2014	Eliminar plan de rotación automático	Según las encuestas realizadas en Colonia, se descubrió que las rotaciones automáticas no eran de interés para los productores e ingenieros agrónomos. Estos preferían un sistema de gestión de sus cultivos ya que ya usaban el programa EROSION 6.0 para la parte de rotaciones.	*Done*
Delegar	12/8/2014	Renombrar cultivos y proyectos	Actualmente el sistema llama a "Cultivo" a la gestión de un padrón con un cultivo. Tabaré dejó claro que esto podía ser confundido con el Cultivo del padrón (trigo, soja). Renombrar a Proyecto	*Done*
Delegar	10/11/2014	Consumir *Web Service* de Noticias	Consumir desde un sitio de noticias para mostrar las principales noticias agros	*Done*
Planificar	6/12/2014	Cuentas multi-clientes y clientes multi-cuentas	Cambiar el hecho que un cliente solo pueda pertenecer a una cuenta. Hacer que un cliente pueda pertenezca a varias cuentas	*Done*
Planificar	9/10/2014	Invitar usuarios a una cuenta	Permitir a un usuario invitar a otro por su mail o su nombre de usuario	*Done*
Rechazar	5/8/2014	Permitir adjuntar foto de perfil	Permitirle a los usuarios asignarse una foto de perfil	*Rejected*
Rechazar	9/9/2014	Campos personaliza	Permitirle al usuario ingresar campos personalizados para	*Rejected*

		dos	crear recetas personalizadas	

Ilustración 11-63 Bitácora de cambios del proyecto

11.26. CASOS DE PRUEBA

Los casos de pruebas que se realizaron se tuvieron en cuenta solo para los requerimientos funcionales con clasificación crítica. Se recuerda que un requerimiento crítico depende de la complejidad del mismo. Con esto se refiere a la cantidad de transacciones a la base de datos que realice como a la lógica que contenga.

A continuación se listan los casos de pruebas.

Escenarios de prueba

Id	**Nombre**	**Descripción**
E01	Aplicación *Backoffice*	Ingreso normal en la aplicación de *Backoffice*
E02	Aplicación	Ingreso normal en la aplicación

Especificación de Casos de prueba

Id del requerimiento	RF10
Id del caso de prueba	CP01
Descripción	Iniciar proyecto de cultivo
Roles involucrados	Usuario administrador
Escenario	E02
Pasos a seguir	1) El usuario selecciona la opción de iniciar proyecto. 2) El usuario completa la información. 3) El usuario selecciona "crear".
Resultado esperado	Se crea un nuevo proyecto.
Criterio de aprobación	

Id del requerimiento	RF10
Id del caso de prueba	CP02
Descripción	Iniciar proyecto de cultivo fallido
Roles involucrados	Usuario administrador
Escenario	E02
Pasos a seguir	1) El usuario selecciona la opción de iniciar proyecto. 2) El usuario no completa toda la información. 3) El usuario selecciona "crear".
Resultado esperado	El sistema muestra mensaje alertando la falta de información.
Criterio de aprobación	

Id del requerimiento	RF13
Id del caso de prueba	CP03
Descripción	Realizar planificación de presupuesto
Roles involucrados	Usuario administrador
Escenario	E02
Pasos a seguir	1) El usuario selecciona la opción de modificar durante la planificación. 2) El usuario completa la información. 3) El usuario selecciona “aceptar”.
Resultado esperado	El sistema actualiza los datos.
Criterio de aprobación	

Id del requerimiento	RF13
Id del caso de prueba	CP04
Descripción	Realizar planificación de presupuesto fallida
Roles involucrados	Usuario administrador
Escenario	E02
Pasos a seguir	1) El usuario selecciona la opción de modificar durante la planificación. 2) El usuario no completa toda la información 3) El usuario selecciona “aceptar”.
Resultado esperado	El sistema muestra mensaje advirtiendo al usuario que debe completar toda la información.
Criterio de aprobación	

Id del requerimiento	RF13
Id del caso de prueba	CP05
Descripción	Realizar planificación de presupuesto
Roles involucrados	Usuario administrador
Escenario	E02
Pasos a seguir	1) El usuario selecciona la opción de modificar con proyecto en ejecución. 2) El usuario completa la información. 3) El usuario selecciona “aceptar”.
Resultado esperado	El sistema actualiza los datos, guardando las diferentes versiones.
Criterio de aprobación	

Id del requerimiento	RF14
Id del caso de prueba	CP06
Descripción	Realizar planificación de presupuesto
Roles involucrados	Usuario administrador
Escenario	E02
Pasos a seguir	1) El usuario selecciona la opción de modificar receta inicial durante la planificación. 2) El usuario completa la información. 3) El usuario selecciona “aceptar”.
Resultado esperado	El sistema actualiza los datos.
Criterio de aprobación	

Id del requerimiento	RF14
Id del caso de prueba	CP07
Descripción	Realizar planificación de presupuesto fallida
Roles involucrados	Usuario administrador
Escenario	E02
Pasos a seguir	1) El usuario selecciona la opción de modificar receta inicial durante la planificación. 2) El usuario no completa toda la información 3) El usuario selecciona “aceptar”.
Resultado esperado	El sistema muestra mensaje advirtiendo al usuario que debe completar toda la información.
Criterio de aprobación	

Id del requerimiento	RF14
Id del caso de prueba	CP08
Descripción	Realizar planificación de presupuesto
Roles involucrados	Usuario administrador
Escenario	E02
Pasos a seguir	1) El usuario selecciona la opción de modificar receta inicial con proyecto en ejecución. 2) El usuario completa la información. 3) El usuario selecciona “aceptar”.
Resultado esperado	El sistema actualiza los datos, guardando las diferentes versiones.
Criterio de aprobación	

Id del requerimiento	RF16
Id del caso de prueba	CP09
Descripción	Realizar planificación de cronograma
Roles involucrados	Usuario administrador
Escenario	E02
Pasos a seguir	1) El usuario selecciona el cronograma. 2) Modifica las tareas. 3) Selecciona guardar.
Resultado esperado	El sistema actualiza los datos de la planificación del cronograma del proyecto.
Criterio de aprobación	

Id del requerimiento	RF16
Id del caso de prueba	CP10
Descripción	Realizar planificación de cronograma
Roles involucrados	Usuario administrador
Escenario	E02
Pasos a seguir	1) El usuario selecciona el cronograma, proyecto en ejecución. 2) Modifica las tareas. 3) Selecciona guardar.
Resultado esperado	El sistema actualiza los datos de la planificación del cronograma del proyecto. El sistema guarda las diferentes versiones.
Criterio de aprobación	

Id del requerimiento	RF17
Id del caso de prueba	CP11
Descripción	Realizar planificación de cronograma
Roles involucrados	Usuario administrador
Escenario	E02
Pasos a seguir	1) El usuario selecciona el cronograma. 2) Modifica el cronograma. 3) Selecciona guardar.
Resultado esperado	El sistema actualiza los datos de la planificación del cronograma del proyecto.
Criterio de aprobación	

Id del requerimiento	RF17
Id del caso de prueba	CP12
Descripción	Realizar planificación de cronograma
Roles involucrados	Usuario administrador
Escenario	E02
Pasos a seguir	1) El usuario selecciona el cronograma, proyecto en ejecución. 2) Modifica el cronograma. 3) Selecciona guardar.
Resultado esperado	El sistema actualiza los datos de la planificación del cronograma del proyecto. El sistema guarda las diferentes versiones.
Criterio de aprobación	

Id del requerimiento	RF21
Id del caso de prueba	CP13
Descripción	Ver lista de proyectos
Roles involucrados	Usuario administrador
Escenario	E01
Pasos a seguir	1) El usuario selecciona la pestaña de proyectos
Resultado esperado	El sistema despliega la lista de proyectos asociados a la cuenta del usuario
Criterio de aprobación	

Id del requerimiento	RF22

Id del caso de prueba	CP14
Descripción	Insertar nueva tarea al cronograma
Roles involucrados	Usuario administrador
Escenario	E01
Pasos a seguir	1) El usuario ingresa a un proyecto activo 2) El usuario inserta una nueva tarea dentro de un proyecto activo, registrando su nombre y fecha.
Resultado esperado	El sistema recalcula el cronograma y lo ajusta según la nueva tarea ingresada
Criterio de aprobación	

Id del requerimiento	RF22
Id del caso de prueba	CP15
Descripción	Insertar nueva tarea al cronograma
Roles involucrados	Usuario administrador
Escenario	E02
Pasos a seguir	1) El usuario ingresa a un proyecto activo 2) El usuario cambia la fecha de una tarea existente
Resultado esperado	El sistema recalcula el cronograma y lo ajusta según la nueva fecha
Criterio de aprobación	

Id del requerimiento	RF22
Id del caso de prueba	CP16
Descripción	Insertar nueva tarea al cronograma
Roles involucrados	Usuario administrador
Escenario	E03
Pasos a seguir	1) El usuario modifica o elimina el uso de un insumo en alguna tarea
Resultado esperado	El sistema actualiza la tarea y el presupuesto del cronograma
Criterio de aprobación	

Id del requerimiento	RF22
Id del caso de prueba	CP17
Descripción	Insertar nuevo insumo
Roles involucrados	Usuario administrador
Escenario	E04

Pasos a seguir	1) El usuario inserta un nuevo insumo a una tarea
Resultado esperado	El sistema actualiza la tarea y el presupuesto del cronograma
Criterio de aprobación	

Id del requerimiento	RF22
Id del caso de prueba	CP18
Descripción	Ingresar fecha estado fenológico
Roles involucrados	Usuario administrador
Escenario	E05
Pasos a seguir	1) El usuario modifica la fecha del próximo estado fenológico
Resultado esperado	El sistema actualiza la etapa del modelo fenológico y las fechas de las próximas etapas.
Criterio de aprobación	

Id del requerimiento	RF23
Id del caso de prueba	CP19
Descripción	Reajustar plan
Roles involucrados	Usuario administrador
Escenario	E01
Pasos a seguir	1) El usuario ingresa al sistema 2) El usuario ingresa a uno de sus proyectos activos
Resultado esperado	El sistema ajusta y notifica a los usuarios los cambios necesarios tanto en cronograma como en insumos para lograr un mejor rendimiento del cultivo.
Criterio de aprobación	

Id del requerimiento	RF20
Id del caso de prueba	CP
Descripción	Notificar hitos del cronograma
Roles involucrados	Usuario administrador
Escenario	E01
Pasos a seguir	1) El usuario ingresa al sistema
Resultado esperado	El sistema muestra a los usuarios los mensajes correspondientes según lo ingresado en la administración de notificaciones.
Criterio de aprobación	Se notifican hitos del cronograma

11.27. DOCUMENTO DE ARQUITECTURA

Fecha	Versión	Quién	Descripción	Revisión
28/07/2014	1.0	Andrés	Creación del documento de arquitectura	28/07/2014
13/08/2014	1.1	Andrés	Actualización del documento de arquitectura	13/08/2014
19/09/2014	1.2	Andrés	Actualización del documento de arquitectura	20/09/2014
12/10/2014	1.3	Clara	Actualización del documento de arquitectura	12/10/2014
01/12/2014	1.4	Andrés	Actualización del documento de arquitectura	05/12/2014

Sobre este documento

Objetivo y alcance del documento

El objetivo de este documento es presentar la arquitectura del sistema y dar un panorama de cuáles son las decisiones de arquitectura más relevantes. Se espera poder brindarle a los distintos *stakeholders* una visión lo suficientemente abstracta para su comprensión y detallada para contemplar la representación de los intereses de los mismos.

Es en este sentido que el presente documento se basa en el enfoque *Views and Beyond* propuesta en la bibliografía [1]. La premisa es documentar aquellos aspectos de la arquitectura relevantes para su comprensión, en términos apropiados para que el lector encuentre la información que necesita según sea su rol respecto a la arquitectura. Esto plantea un desafío doble, por un lado ser suficientemente descriptiva como para que los aspectos relevantes de la arquitectura no se pierdan ni caigan en la ambigüedad a la hora de ser interpretada por los desarrolladores, mientras que por otro lado debe funcionar como una pilar de comunicación con el cliente. Para lograr estos objetivos, se utiliza una combinación de vistas con detalle técnico y vistas informales, de fácil lectura para el cliente.

Además de la documentación de vistas arquitectónicas, este documento contiene las principales decisiones relevantes a la construcción de la solución y el despliegue y mantenimiento de la misma. Siendo estas decisiones estrechamente relacionadas con la definición de arquitectura, se entiende que agregan valor a este documento, convirtiéndolo en un punto de referencia sobre la arquitectura durante toda la vida útil del producto.

Los diseños presentados en forma de vistas en este documento se corresponden con los lineamientos que la arquitectura define a alto nivel, dejando las implementaciones concretas no especificadas a criterio de los desarrolladores.

Se asume que el lector está familiarizado con el problema y los requerimientos relevados.

Introducción al problema

SIGAMAS se presenta como un sistema de apoyo a la toma de decisiones, capaz de resolver una variedad de requerimientos, entre ellos:

- Gestionar planes de cultivos.
- Modelar distintos cultivos.
- Generar alertas y notificaciones basadas en el seguimiento de los planes.
- Centralizar información de fuentes externas para uso interno.

Resulta natural inferir que es posible modularizar el sistema basándose en las responsabilidades particulares de cada funcionalidad.

SIGAMAS presenta otra característica particular, que es la capacidad para extender el sistema agregando nuevos modelos de cultivos. Esto nos lleva a un escenario en el que distintos módulos o subsistemas deben interactuar entre sí, de forma desacoplada e independiente.

Los usuarios definidos de SIGAMAS no son particularmente afines a uso de tecnologías de apoyo a su labor, en la teoría de difusión de las innovaciones pueden clasificarse como “Rezagados” por las características que fueron relevadas. Para poder romper esta barrera el sistema debe proporcionar características de usabilidad y calidad de solución que fomenten su uso, donde se debe cuidar particularmente la interacción con el usuario y la respuesta del sistema.

El contexto de uso de la aplicación exige que su acceso sea descentralizado, dado que los usuarios deben poder hacer uso de esta en el lugar en que se encuentren, lo que lleva a pensar en una aplicación web publicada en Internet. Debe considerarse también que en muchas ocasiones el usuario puede estar ubicado en una zona de poca o nula conectividad a Internet, y debe brindarse una alternativa para que pueda operar hasta volver a tener conexión.

Organización del documento de arquitectura

Se presentó en la sección anterior una definición básica del problema y algunos conceptos importantes que surgen de un análisis preliminar.

A continuación, se presentaran los *stakeholders* del proyecto, y cuál es su relación con este documento.

Luego se presentara un análisis basado en los requerimientos no funcionales relevados que permite determinar que atributos de calidad se van a priorizar. En base a este análisis se desarrolla el resto del documento.

Se detalla en el siguiente apartado el proceso de selección de tecnologías, basado en los atributos de calidad definidos. Luego de este análisis se muestra el análisis y gestión de riesgos tecnológicos.

Una vez definidos claramente los atributos de calidad a cumplir y las tecnologías de apoyo, se pasa a la definición de arquitectura y representación de vistas. A continuación, se detalla las tácticas y patrones utilizados.

Luego se detalla la infraestructura sugerida para montar la solución, para luego pasar a la justificación de la arquitectura en términos de los atributos de calidad definidos.

Para cerrar el análisis se realiza una evaluación de la arquitectura.

Luego se anexa un plan de mantenimiento acorde a las definiciones realizadas y un catálogo que muestra como la arquitectura resuelve cada requerimiento no funcional.

Stakeholders

A continuación se detallan los *stakeholders* del proyecto y que rol cumplen con respecto a la definición de arquitectura. Este rol determina el uso que van a hacer los distintos involucrados del presente documento:

Stakeholder	**Descripción**
Arquitecto Experto	Evalúa que el documento de arquitectura sea correcto en cuanto a expresividad, forma y contenido.
Desarrolladores	Utiliza el documento como guía para el desarrollo
Cliente	Verifica en el documento que las ideas principales estén acordes a sus necesidades

Atributos de calidad a cumplir

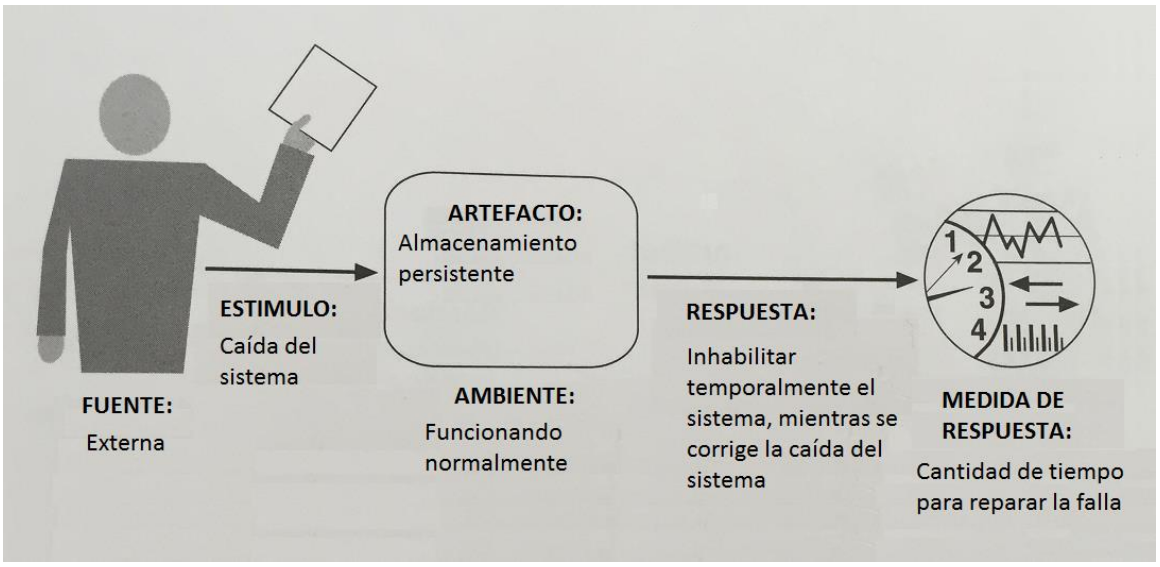
En esta sección se expone un análisis de los atributos de calidad que el sistema debe satisfacer, basado en el relevamiento de requerimientos realizado.

La metodología definida para el análisis de requerimientos determinó que se agruparan según el atributo de calidad con el que están relacionados, de modo que se pueda reutilizar ese trabajo en el análisis arquitectónico.

Los atributos de calidad que se van a considerar son:

Disponibilidad

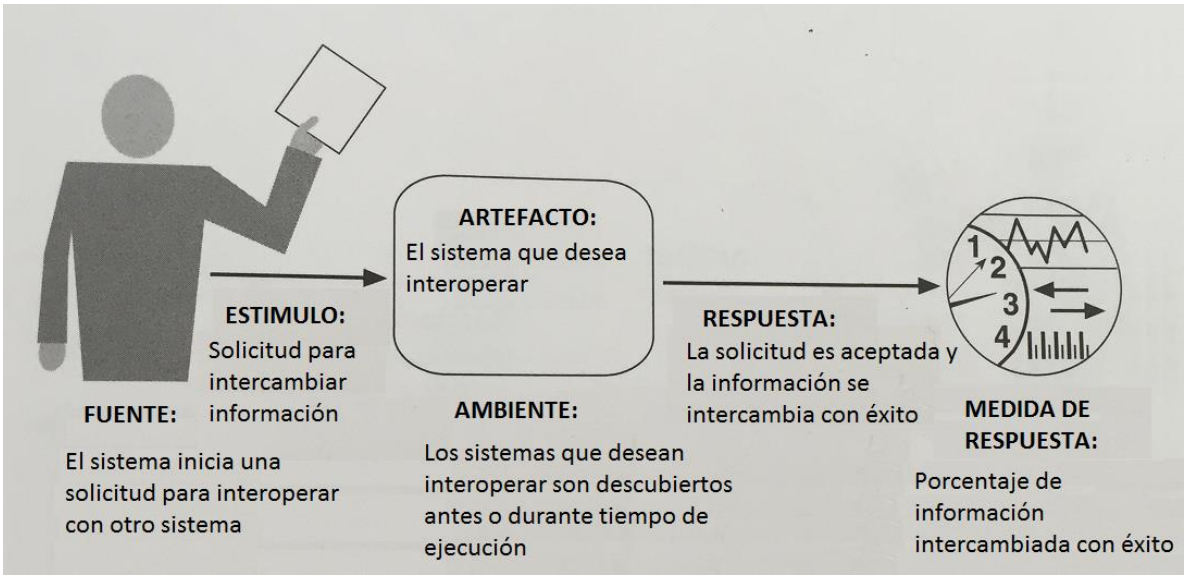
Determinado por los requerimientos no funcionales 13 y 14. El sistema debe tener alta disponibilidad y retomar el funcionamiento en caso de fallas en el menor tiempo posible Escenario tipo:



Interoperabilidad

El sistema debe ser capaz de comunicarse con otras aplicaciones, principalmente servicios externos que provean de datos como ser el clima a SIGAMAS. Requerimiento no funcional asociado número 23.

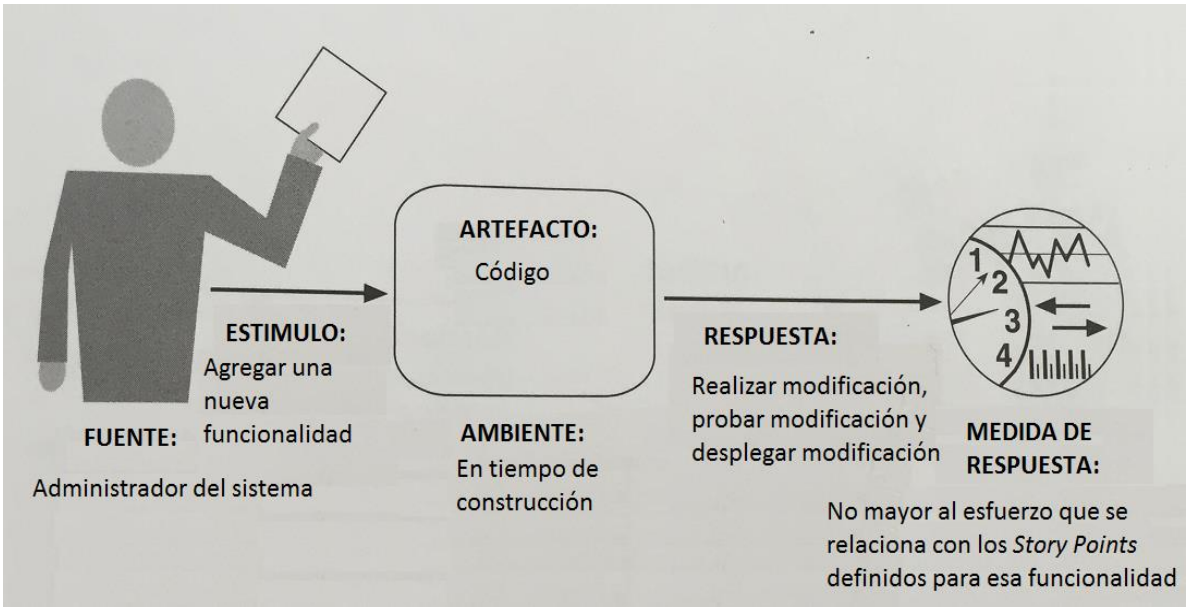
Escenario tipo:



Modificabilidad

Es imperativo que el sistema presente características de diseño que favorezcan la modificabilidad, dado que el desconocimiento del área que se está modelando genera la posibilidad que se requieran cambios sobre el programa para adaptarlo. Determinado los requerimientos no funcionales 27 a 29.

Escenario tipo:

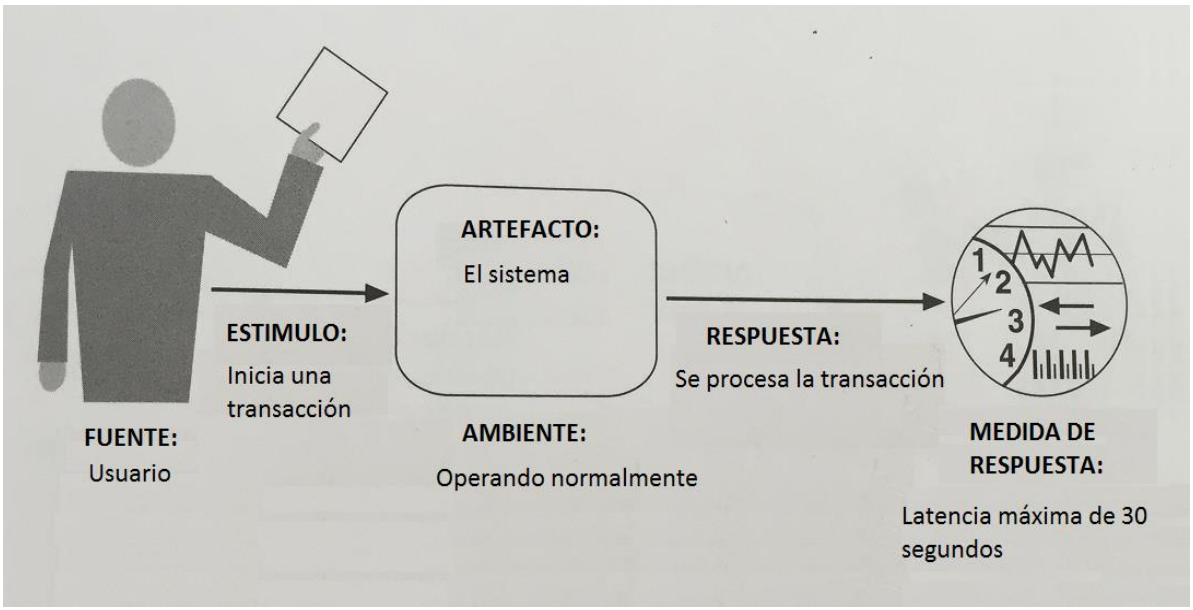


Performance

Dadas las características de los usuarios, es imperativo que el sistema responda de manera rápida y eficiente para no decepcionarlos y fomentar así que adopten el sistema.

Requerimientos no funcionales 6 a 9.

Escenario tipo:

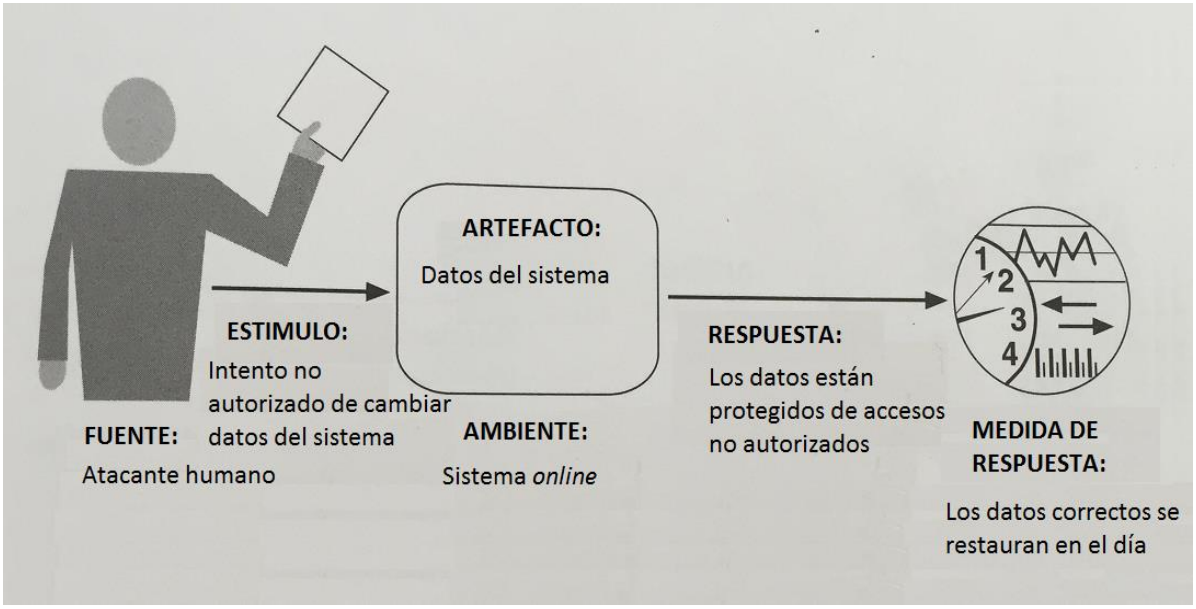


Seguridad

Dado que el sistema maneja información de gestión y de negocio, es fundamental asegurar la seguridad de la información.

Requerimientos no funcionales 3 y 4.

Escenario tipo:

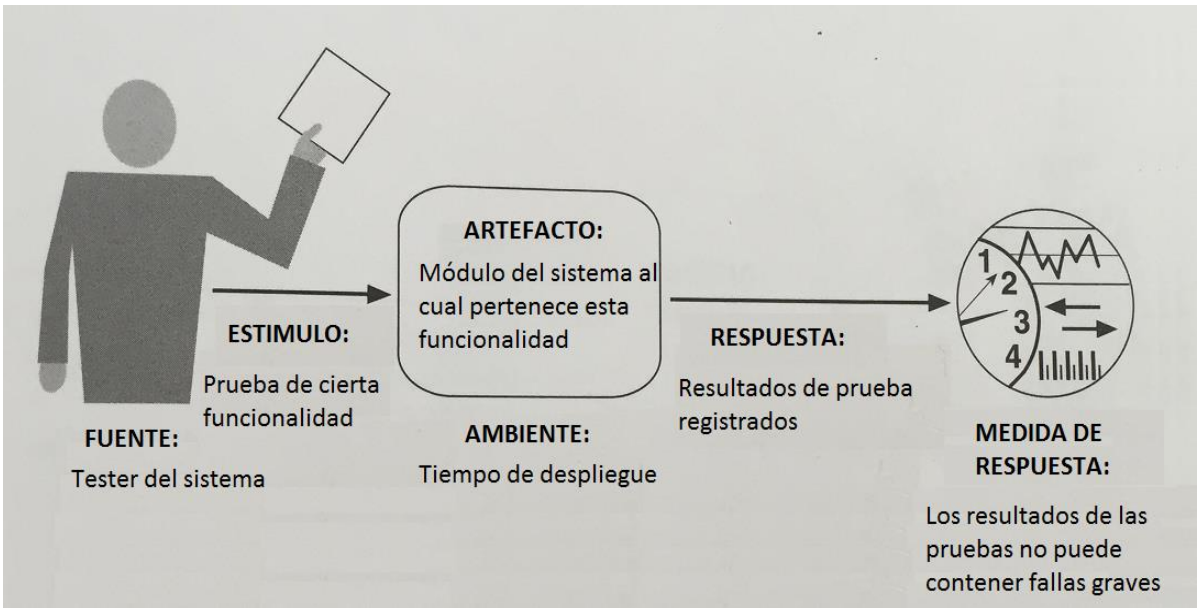


Testeabilidad

Con el fin de asegurar la calidad del producto final dentro de las restricciones de tiempo dadas por el proyecto, se requiere favorecer la testeabilidad.

Requerimientos no funcionales 32 al 35

Escenario tipo:

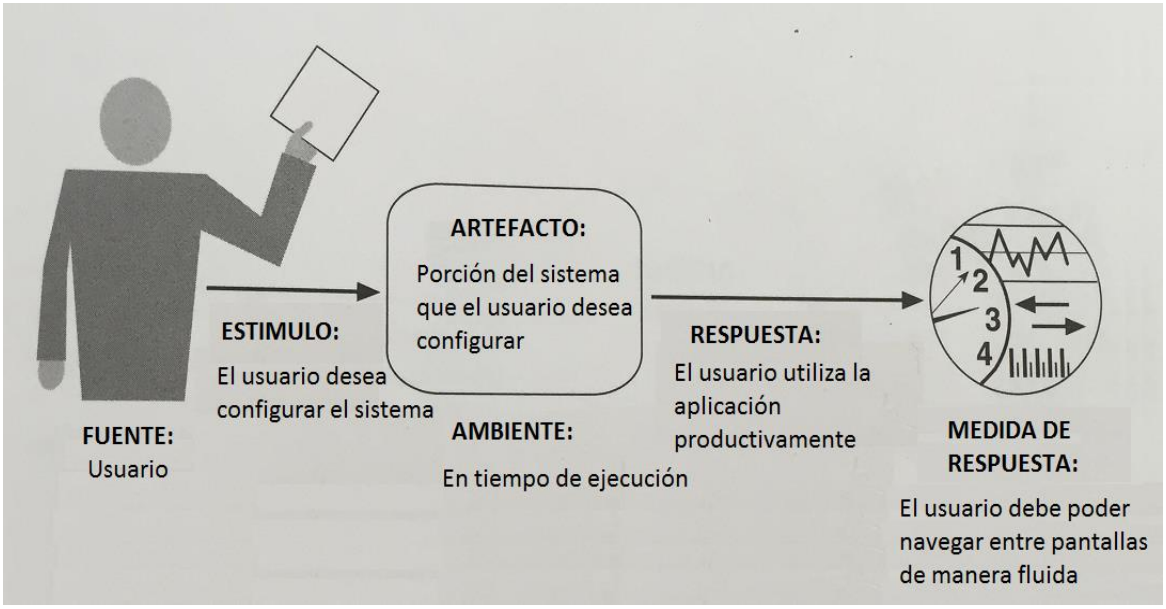


Usabilidad

Es fundamental, por la naturaleza de los usuarios, favorecer en todo sentido la usabilidad de modo que se fomente el uso de la aplicación de forma natural y fluida.

Requerimiento no funcional 30.

Escenario tipo:



Elección de tecnologías de apoyo

A continuación se exponen los análisis que definen las tecnologías en las cuales se apoya el desarrollo del sistema SIGAMAS. Esta elección estuvo basada en la premisa de minimizar el tiempo de desarrollo sin perjudicar la calidad de la solución final, puesto que el proyecto requiere de una gran carga de investigación previa para poder construir una solución de software. Por este motivo, factores como la madurez y la robustez de la tecnología analizada, así como la experiencia de los miembros del equipo, son conceptos primordiales en la toma de decisiones planteada.

Elección de Lenguaje

La elección del lenguaje de desarrollo se basó en un análisis comparativo de aquellos lenguajes bien conocidos por el equipo. Se analizaron los lenguajes propuestos por cada integrante en base a tres factores que fueron considerados clave:

-Experiencia previa, tanto académica como laboral

-Soporte y documentación disponible

-Experiencia en uso de *Frameworks* de apoyo al desarrollo

Los lenguajes evaluados fueron C# (.NET), Python y Java. Se muestra a continuación una tabla comparativa con resumen del análisis realizado en base a los factores considerados, donde en cada celda se detalla la cantidad de integrantes que cumplen la premisa:

Lenguaje	Experiencia previa(académica)	Experiencia previa(laboral)	Experiencia en *Frameworks* de apoyo
C# (.NET)	4	0	0
Python	0	1	0
Java	4	2	2

El lenguaje en el que el equipo suma más experiencia es Java, que además posee una gran comunidad que da soporte. El hecho de que sea una tecnología libre también es importante, dado que existen entornos de desarrollo gratuitos con lo que no es necesario realizar ninguna inversión económica para desarrollar.

El paradigma de programación orientada a objetos es manejado por todos los miembros del equipo, quienes además poseen experiencia tanto laboral como académica en desarrollo de aplicaciones JEE (Java *Enterprise Edition*), por lo que Java es la elección de preferencia para minimizar la curva de aprendizaje de tecnologías y favorecer un desarrollo ágil.

JEE o Java Empresarial, es una plataforma que permite el desarrollo que permite arquitecturas de N capas distribuidas y se basa en el uso de componentes de software modulares que se ejecutan en un servidor de aplicaciones. A su vez, JEE define en su especificación un *stack* (conjunto de APIs) de tecnologías que dan soporte a una variedad de funcionalidades, como son el acceso a base de datos unificado o la capacidad de operación transaccionales. De esta manera JEE permite una gran capacidad de expresión en cuanto a lo que el diseño arquitectónico refiere, y se adapta a las necesidades del proyecto SIGAMAS, favoreciendo que se satisfagan de los atributos de calidad definidos.

El *stack* JEE provee también APIs para el desarrollo web, para este proyecto se utilizará la tecnología JSF (Java Server Faces), que da soporte a la generación de páginas web dinámicas. Además provee tecnología para la inyección de dependencias y recursos gestionados por el contenedor de aplicaciones de JEE.

Servidor de aplicaciones

En el contexto de construir una aplicación JEE, un punto significativo es la elección de un servidor de aplicaciones para ejecutar y contener los artefactos de Software creados.

El mencionado servidor debe poseer, además de las capacidades estándar, algunas características particulares que son prioridad para el equipo de desarrollo:

- Ser Liviano
- De fácil instalación
- De uso intuitivo
- Con posibilidad de despliegue en caliente (Sin reiniciar el servidor)
- Que permita la configuración de d*atasources* (orígenes de datos) y colas de mensajes sin mayor dificultad.

Opciones del mercado

Se listan aquellos servidores con compatibilidad JEE (Java *Enterprise Edition*), preferentemente en versión 6 o más cuya licencia sea libre:

Servidor	**Empresa**	**Versión**	**Fecha de lanzamiento**	**Versión JEE**
Geronimo	ASF	3.0.1	2013-05-28	6
Glassfish	Oracle	4.0	2013-05	7
JBoss	Red Hat	6.0.2	2013-12-05	6
Jetty	Eclipse	9.1.5	2014-05-05	7(Parcial)
JEUS	TmaxSoft	7	2012-06	6
TomEE	ASF	1.6.0	2013-11-20	6(WEB)

Ilustración 11-64 Servidores con compatibilidad JEE

Glassfish y factores que llevan a esta elección

Dentro de la amplia gama de servidores disponibles, Glassfish no solo tiene el soporte de Oracle y una gran comunidad, sino que es un servidor que se utiliza habitualmente en trabajos de materias en la facultad, por lo que existe en el equipo experiencia en el servidor que se puede aprovechar.

Este servidor no tiene soporte comercial, sin embargo se mantiene vigente en la comunidad dada su completa y madura implementación del *Stack* JEE.

Framework de persistencia

Dadas las características del proyecto y las tecnologías utilizadas, el paradigma de programación a usar va a ser Orientación a Objetos y junto con esto y la tecnología de soporte que es JEE, el estándar **JPA** (Java *Persistence* API) es un enfoque que reúne los siguientes atributos:

- Estándar maduro
- Gran soporte en la comunidad Java
- Gran cantidad de implementaciones
- Desacoplado de las implementaciones
- Soporte integrado en la mayoría de los IDEs

Ventajas de desarrollo

El estándar JPA es de uso muy simple, y permite una gran capacidad de expresión mediante el uso de anotaciones. Dentro de JPA se maneja el concepto de Entidades para denominar un conjunto de datos relacionados, que pude traducirse por ejemplo en una tabla que refleja el estado de diferentes objetos. De esta manera la Entidad se vuelve una suerte de proxy

entre la representación como objeto pertinente al mundo POO (Programación Orientada a Objetos) y la representación como tupla en el mundo, por ejemplo, relacional.

Una Entidad no es más que un POJO con anotaciones específicas. Un ejemplo clásico puede ser el siguiente:

```
@Table
 public class Empleado{
        @Id
        @Column
        private long pk;
        @Column
        private String nombreEmpleado;
        @Column
        private String direccionEmpleado;
        //Código
 }
```

Mediante estas anotaciones la implementación de JPA traduce el objeto a tabla y viceversa.

La implementación de JPA que elegida para este proyecto es EclipseLink, de gran aceptación en la comunidad Java y por tanto con gran soporte.

Ambiente de desarrollo

Para el desarrollo de la aplicación se utilizará NetBeans 8.0.2. Se comparó con Eclipse y se llegó a la conclusión de elegir NetBeans porque provee más automatizaciones en las tareas que se iban a realizar. Ya viene preparado con *Plugins*, sin la necesidad de agregarlos, lo que puede generar más configuraciones y dificulta llegar a un ambiente de desarrollo unificado para todo el equipo.

El equipo no posee experiencia en otras herramientas populares como ser IntelliJ, por lo que no se exploraron esas opciones.

Librerías de terceros

Se utilizará la librería PrimeFaces que permite facilitar la creación de aplicaciones web, dado que provee un conjunto de componentes que extienden las funcionalidades de los componentes básicos de JSF (Java Server Faces) y acelera la tarea de desarrollo.

Asimismo se utilizó la librería Log4J para gestionar *Logs* de auditoría del sistema.

Adicionalmente se utilizó el Framework JUnit para la automatización de pruebas unitarias.

Base de datos

Para la persistencia de datos se utilizó una base de datos MySql, en su versión 6.2. Se optó por esta base de datos siguiendo la misma línea de elección de tecnologías de desarrollo fundamentada en dos pilares: ser una herramienta *Open Source*, experiencia del equipo en la tecnología y gran soporte por la comunidad de la misma.

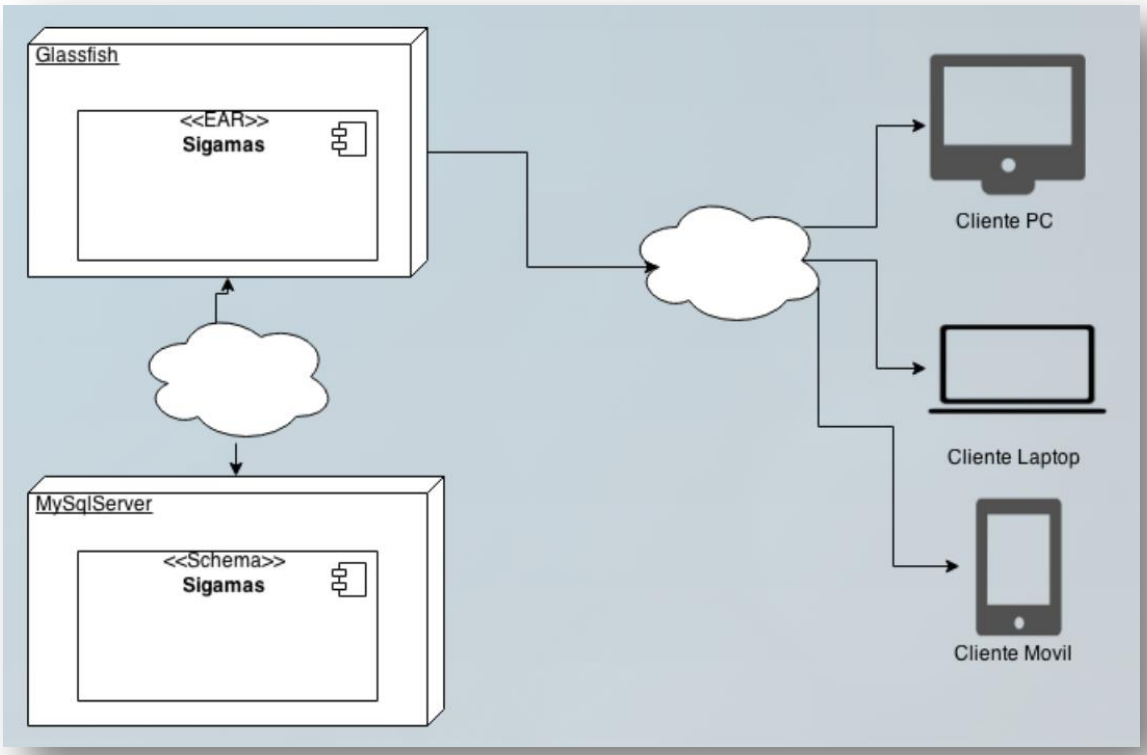
La base está alojada en el mismo servidor que el ambiente de desarrollo, no obstante en un futuro se pretende mover la base a un servidor físico diferente de modo de favorecer la seguridad e integridad del sistema.

Riesgos tecnológicos

Si bien las tecnologías utilizadas fueron elegidas con foco en los talentos del grupo, esto no exime de la posibilidad de que surjan problemas propios del desarrollo y el entendimiento de las mismas, por lo que se hace relevante prever los riesgos que puedan ocasionar estos problemas y gestionarlos. Dado el proceso definido y la metodología de trabajo del equipo de desarrollo, se debe considerar la gestión de riesgos tecnológicos dentro de la gestión de riesgos del proyecto.

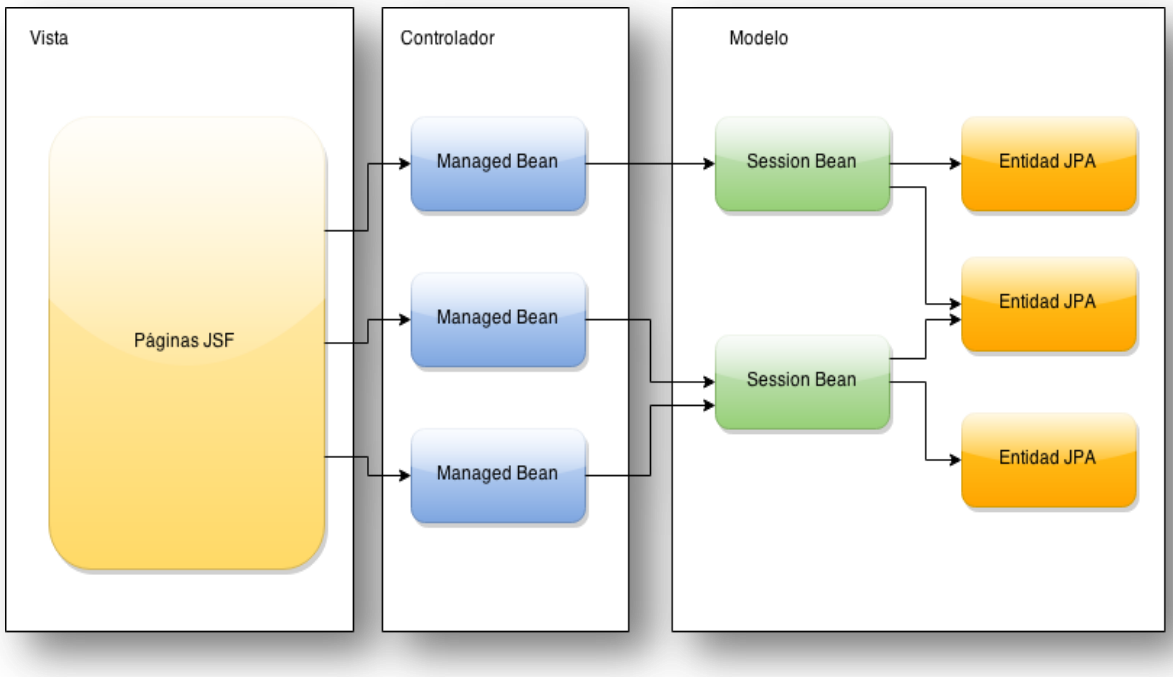
Definición de arquitectura

Se plantea una solución arquitectónica basada en el patrón cliente-servidor, a través de internet mediante páginas web.



La arquitectura definida se corresponde con un diseño separado en tres capas físicas, favoreciendo el bajo acoplamiento y la alta cohesión. Las capas seleccionadas fueron la de Persistencia, Negocio y Presentación, manteniendo la dependencia en un solo sentido y apoyándose en las APIs del *Stack* de JEE (Java *Enterprise Edition*) para mantener esta relación. Asimismo se utilizó el patrón de diseño MVC para bajar a tierra estas tres capas y mapearlas con componentes propios de la tecnología JEE. Se muestra a continuación un diagrama de capas simplificado que ejemplifica lo planteado.

Diagrama que muestra la separación de elementos que se definen para el uso del patrón MVC:



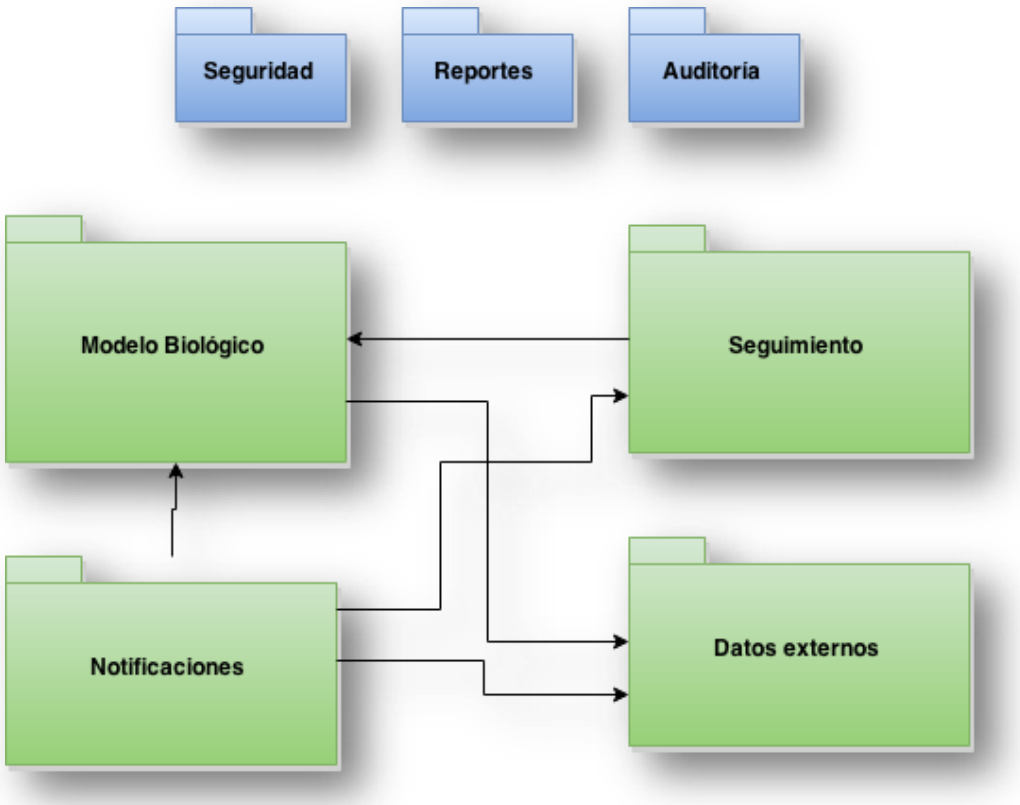
Estrictamente hablando, los *Managed Beans* se pueden considerar parte del modelo, y el controlador es un elemento llamado *FacesServlet* quien es el encargado de procesar las solicitudes de los pedidos (*requests*) que hace el cliente en las páginas web. Se opta por esta

representación informal porque se adapta al concepto que se quiere plasmar en el desarrollo del producto: La lógica de negocio esta relegada a los EJB (*Session Beans*) y los *Managed Beans* solo deben articular entre ellos y la interfaz.

Vista Lógica

Se describen a continuación los sucesivos refinamientos de la descomposición del sistema en módulos desde la perspectiva de la vista lógica.

Descomposición en subsistemas (módulos)



SIGAMAS está subdividido en cuatro módulos principales y tres de apoyo. Los módulos principales encapsulan la funcionalidad principal de SIGAMAS en lo referido a la resolución de la lógica de negocio, mientras que los de apoyo engloban tareas propias de la solución de software.

El módulo de Seguridad tiene como responsabilidad la autenticación y autorización de los usuarios del sistema, siendo el encargado de la gestión de usuarios, cuentas y permisos dentro de la aplicación. El módulo de Auditoria por su parte se encarga de la persistencia de trazas de auditoría del sistema, siendo clave para el mantenimiento y la recuperación del mismo. La responsabilidad del módulo Reportes es la recopilación de información relevante para el usuario, en un formato que le sea de utilidad previamente definido.

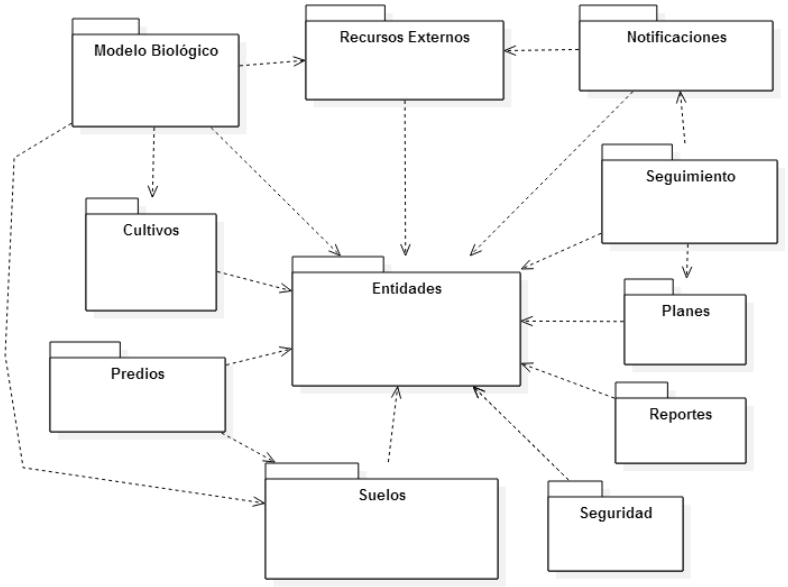
El modulo Modelo Fenológico encapsula toda la lógica referida al modelado de crecimiento de los cultivos que SIGAMAS debe implementar. Le corresponde a este módulo actualizar el estado en base a la información provista por el modulo Datos Externos.

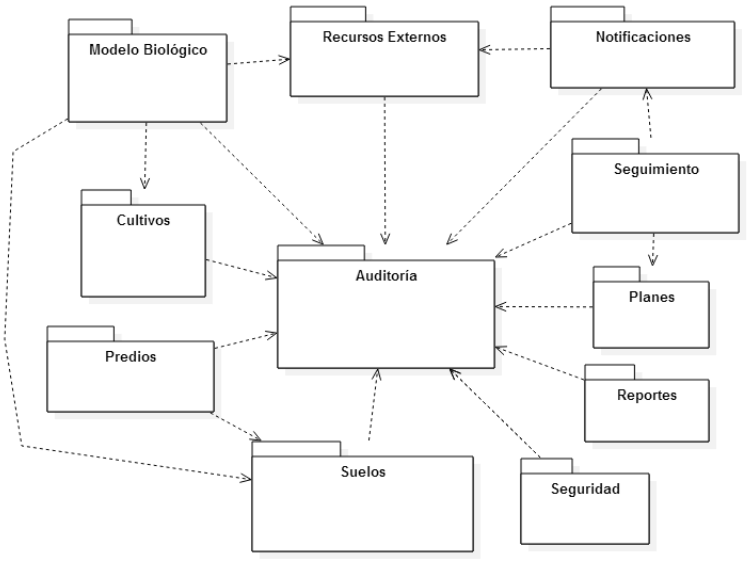
Datos externos se encarga de recopilar información de distintos orígenes (*Web Services* externos) y procesarla para que sea de utilidad en el sistema. Este módulo debe asegurar la información, por lo que puede usar redundancia para obtener los mismos datos de distintos orígenes si alguno de ellos falla.

Por su parte, el modulo Seguimiento es la implementación del BPM (*Business Process Model*) relevado para cada cultivo, integrando las tareas que se deben llevar a cabo. El módulo de seguimiento depende del módulo Modelo Fenológico, dado que las tareas a realizarse están fuertemente relacionadas con la fase actual del cultivo con el que se está trabajando.

Por último, el módulo de Notificaciones se encarga de evaluar indicadores que permiten generar avisos de interés del usuario. Estas notificaciones pueden provenir de datos externos, el desarrollo del modelo fenológico o el seguimiento del proyecto.

Vista de paquetes





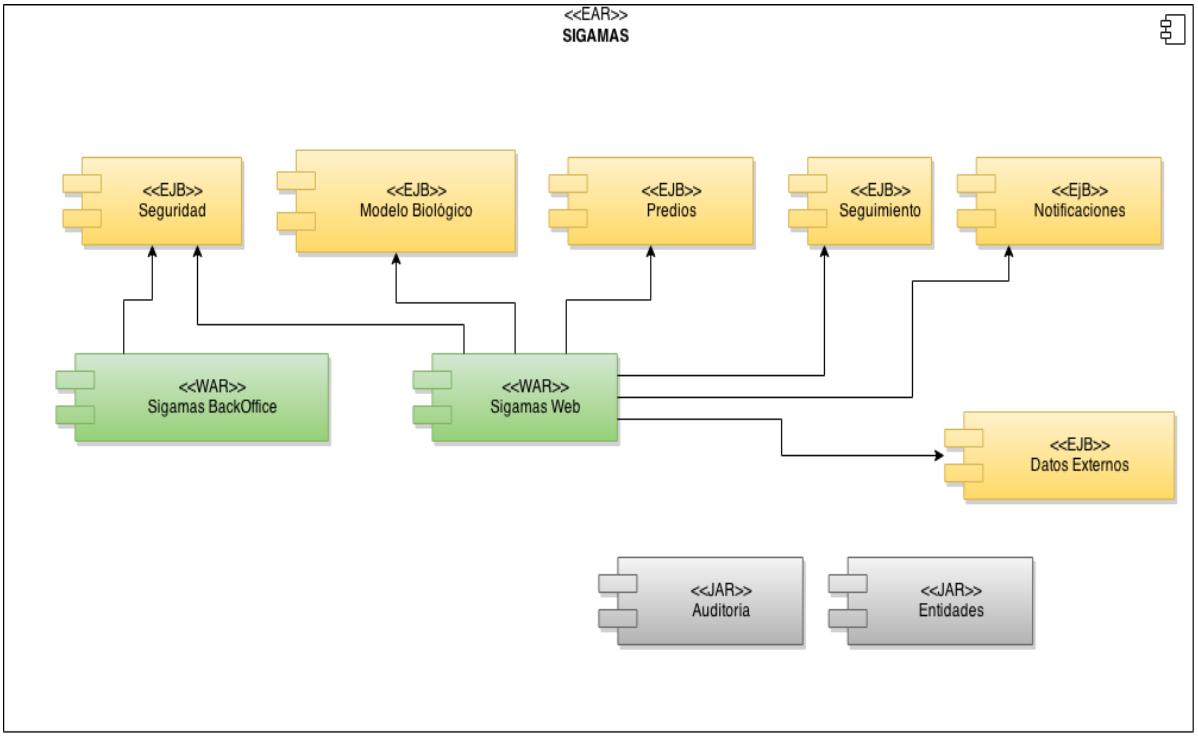
Catálogo de elementos

Elemento	**Responsabilidades**
Recursos Externos	Recoge y transforma información relevante al sistema (como ser datos climáticos, económicos, de suelos, etc.) desde orígenes de datos externos (*Web Services* por ejemplo)
Notificaciones	Gestiona las notificaciones de los usuarios. Se encarga de controlar los indicadores definidos que disparan una notificación y manejar la notificación una vez que se dan las condiciones para que se cree.
Modelo Biológico	Responsable de la simulación de los modelos teóricos de crecimiento de los cultivos que permite saber en qué fase fenológica se encuentran según las paramétricas que estos modelos manejan.
Cultivos	Guarda y gestiona información de los cultivos y sus fases.
Predios	Guarda y gestiona información geo-referenciada de los predios que el sistema maneja.
Suelos	Mantiene la información de interés biológico de los suelos que componen un predio. Es de vital importancia para el Modelo Biológico
Seguridad	Gestiona permisos y usuarios.
Reportes	Genera reportes con la información del sistema
Planes	Maneja los planes de trabajo predefinidos en el sistema. Adicionalmente permite la creación de planes personalizados. Los planes son el conjunto de tareas base para el desarrollo de un proyecto de cultivo
Seguimiento	Se encarga de monitorear la ejecución de los proyectos de cultivo y determinar que tareas están pendientes. Este módulo es también el encargado de calcular que tareas nuevas implica un desvío y se basa en el Modelo Biológico para hacerlo.

Entidades	Mantiene todas las entidades del sistema, suborganizadas según a que modulo dan soporte.
Auditoria	Provee funcionalidad para generar trazas de auditoría útiles para el análisis del funcionamiento del sistema

Vista de Componentes

Representación preliminar

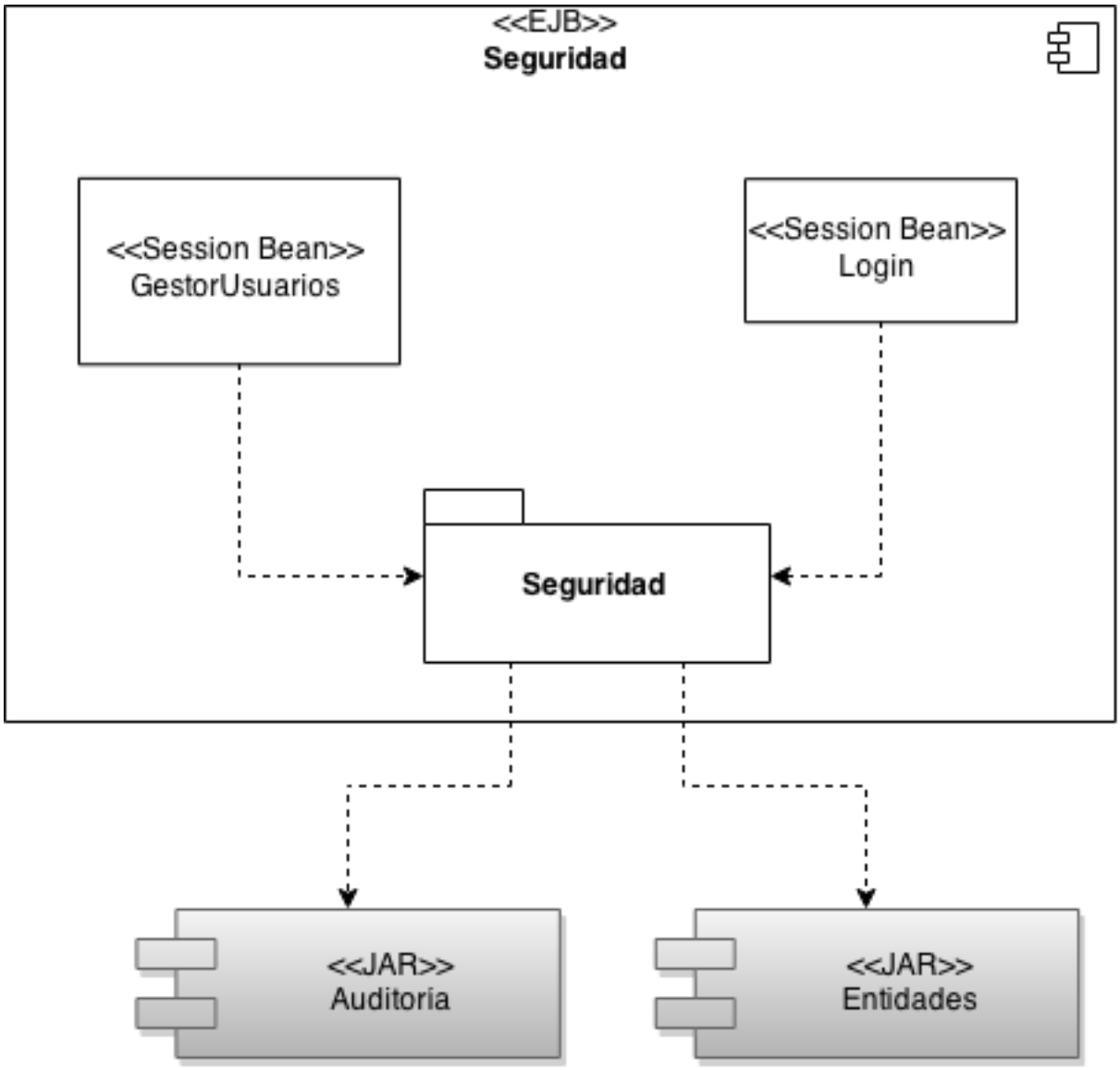


El componente principal de la solución es la aplicación JEE, que físicamente se mapea a un binario de extensión EAR (*Enterprise Archive*). Este componente engloba todos los demás y es el que se despliega en el servidor de aplicaciones.

Se detallan a continuación los componentes principales que integran la aplicación:

EJB Seguridad

Representación primaria



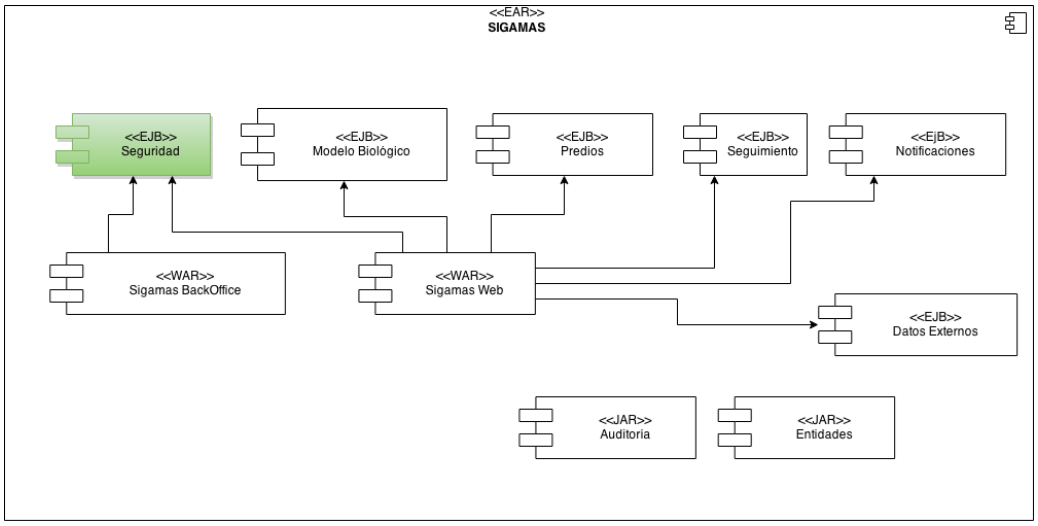
Catálogo de elementos

Elemento	Responsabilidades
GestorUsuarios	*Session Bean* que dispone los servicios para la creación, modificación y borrado de usuarios y cuentas.
Login	*Session Bean* responsable de la validación de accesos al sistema
Seguridad	Modulo que contiene la lógica de negocio que utilizan los EJB

Justificación

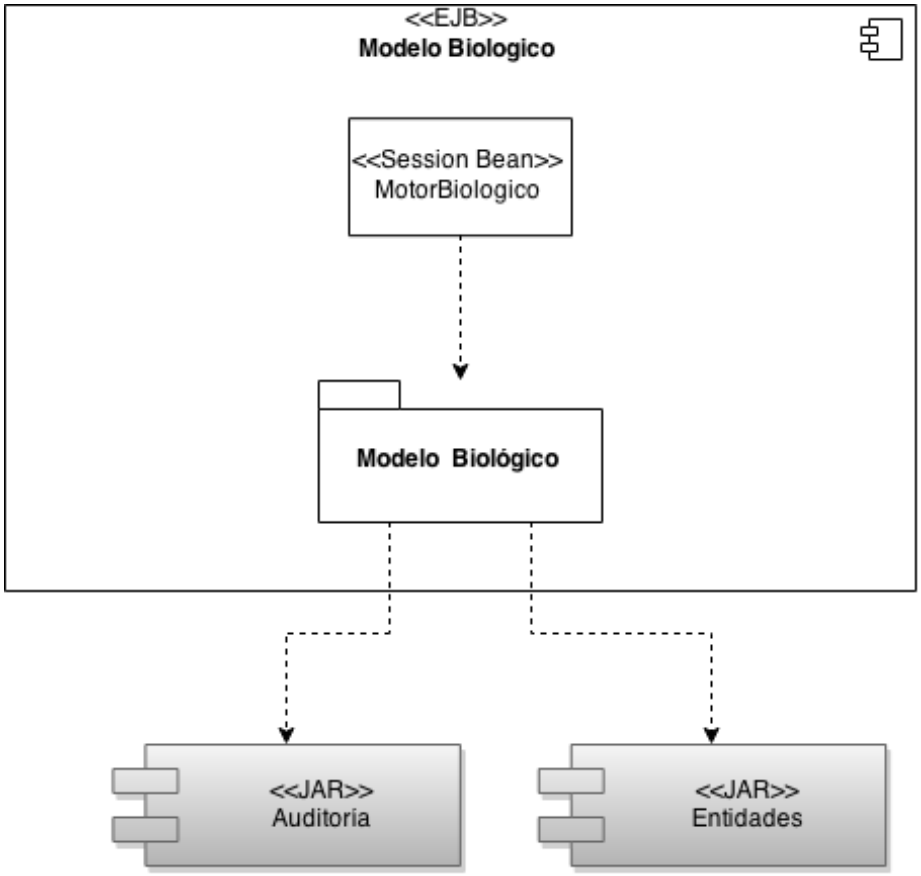
Se exponen dos *Session Beans* que encapsulan responsabilidades bien definidas, de modo de favorecer el principio de aislamiento.

Contexto



EJB Modelo Fenológico (o biológico)

Representación primaria



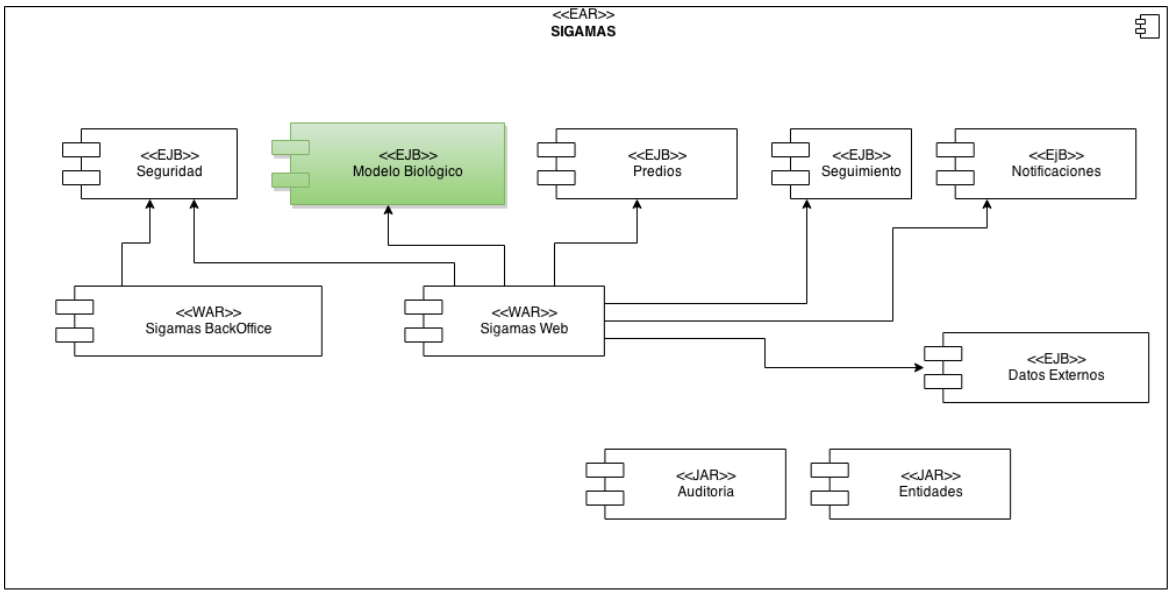
Catálogo de elementos

Elemento	Responsabilidades
MotorBiologico	*Session Bean* que expone las funcionalidades relacionadas con la actualización y mantenimiento de los modelos biológicos de los diferentes proyectos de cultivo
Modelo Biológico	Modulo que provee la lógica de negocio de los modelos biológicos

Justificación

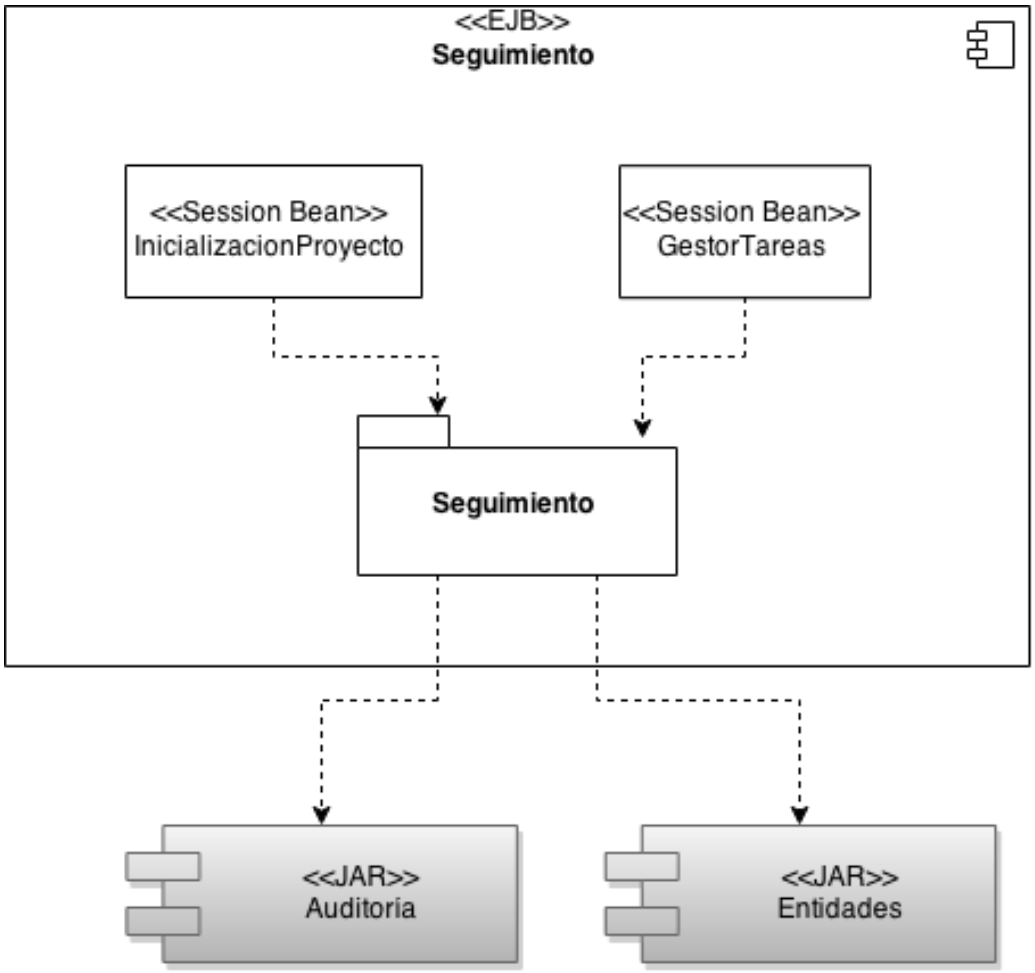
Con el fin de favorecer el bajo acoplamiento y sacar provecho de la gestión de recursos del contenedor JEE, se exponen mediante un *Session Bean* las funcionalidades del módulo.

Contexto



EJB Seguimiento

Representación primaria



Catálogo de elementos

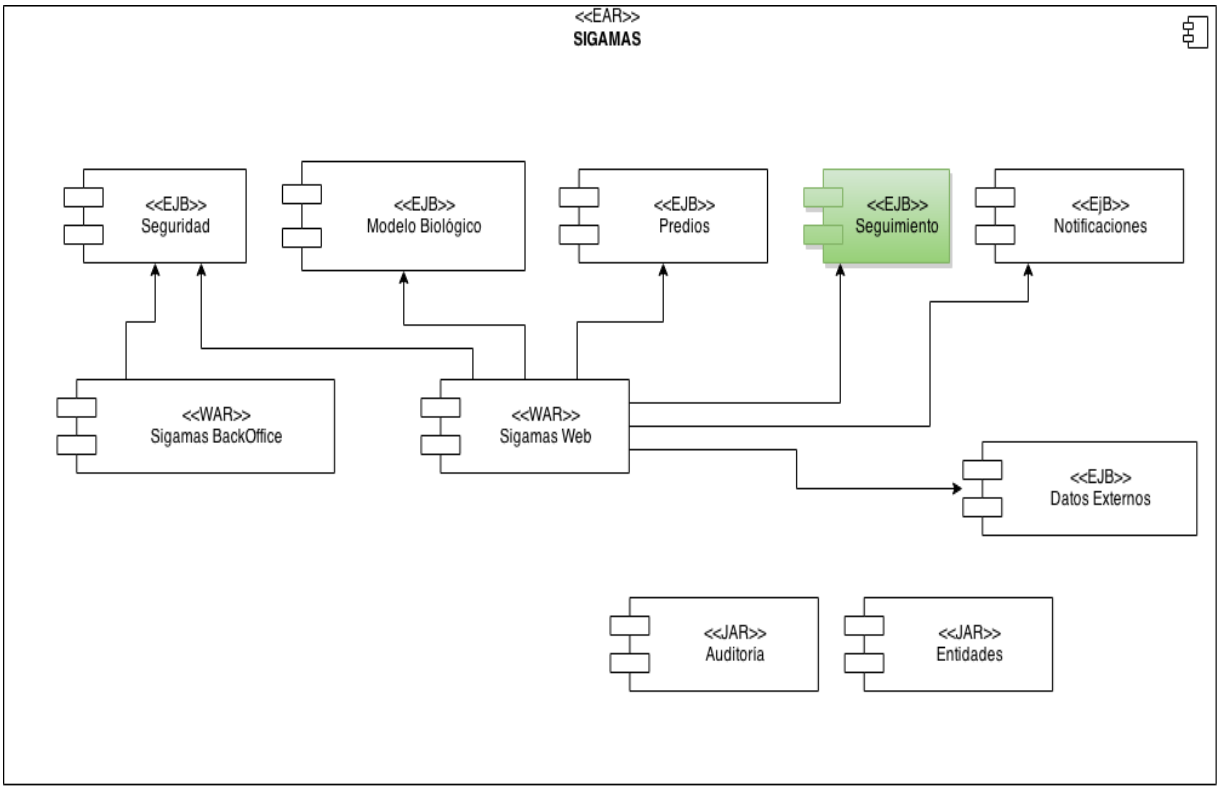
Elemento	Responsabilidades
InicializacionProyecto	Responsable de todo lo que implica la creación de un nuevo proyecto
GestorTareas	*Session Bean* encartado de la gestión de las tareas de un proyecto de cultivo, así como del cálculo de los desvíos que se pueden generar.
Seguimiento	Modulo que contiene la lógica de negocio subyacente en el contexto del seguimiento de proyectos de cultivo

Justificación

Se divide en dos *Session Bean*s la funcionalidad expuesta, puesto que la inicialización de un proyecto de cultivo es una operación compleja que implica no solo inicializar las tareas, sino inicializar el modelo de cultivo asociado.

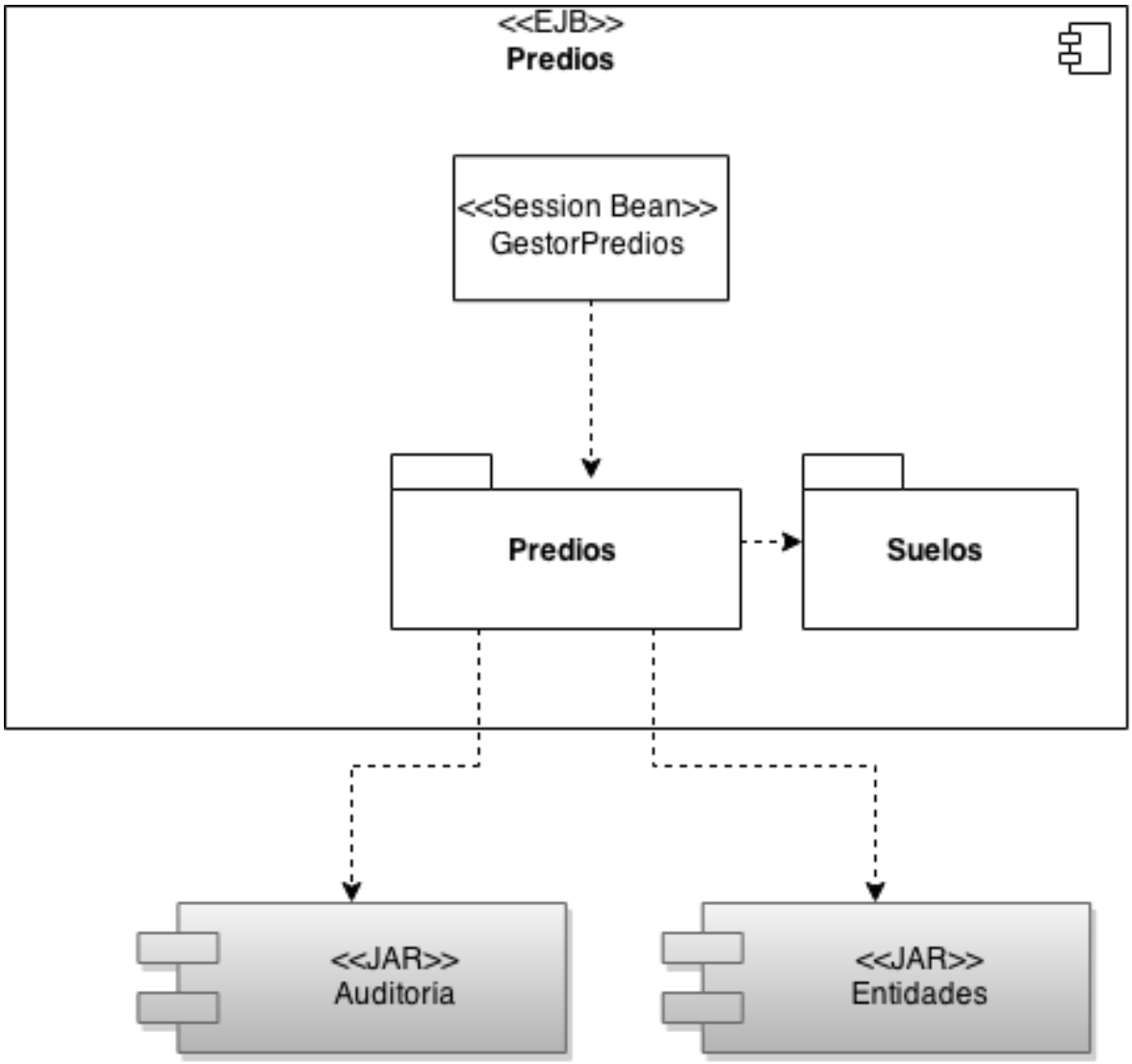
La responsabilidad del GestorTareas engloba el seguimiento de un proyecto en curso, considerando las desviaciones que se puedan dar tanto por las tareas previstas como por las tareas que surgen de los cambios de fase en el modelo biológico.

Contexto



EJB Predios

Representación primaria



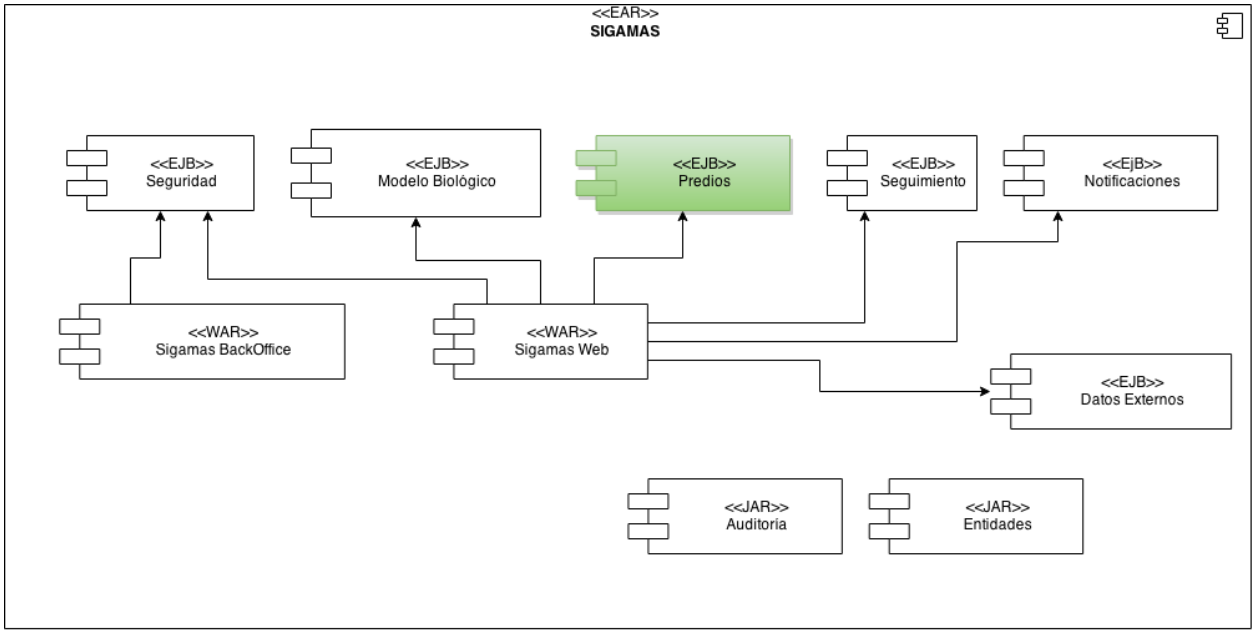
Catálogo de elementos

Elemento	Responsabilidades
GestorPredios	*Session Bean* encargado del mantenimiento de predios y las características de estos, entre ellas las propiedades de los suelos
Predios	Lógica de negocio asociada a la gestión de la información de los predios
Suelos	Logia de negocio asociada a la gestión de la información de las propiedades de los suelos

Justificación

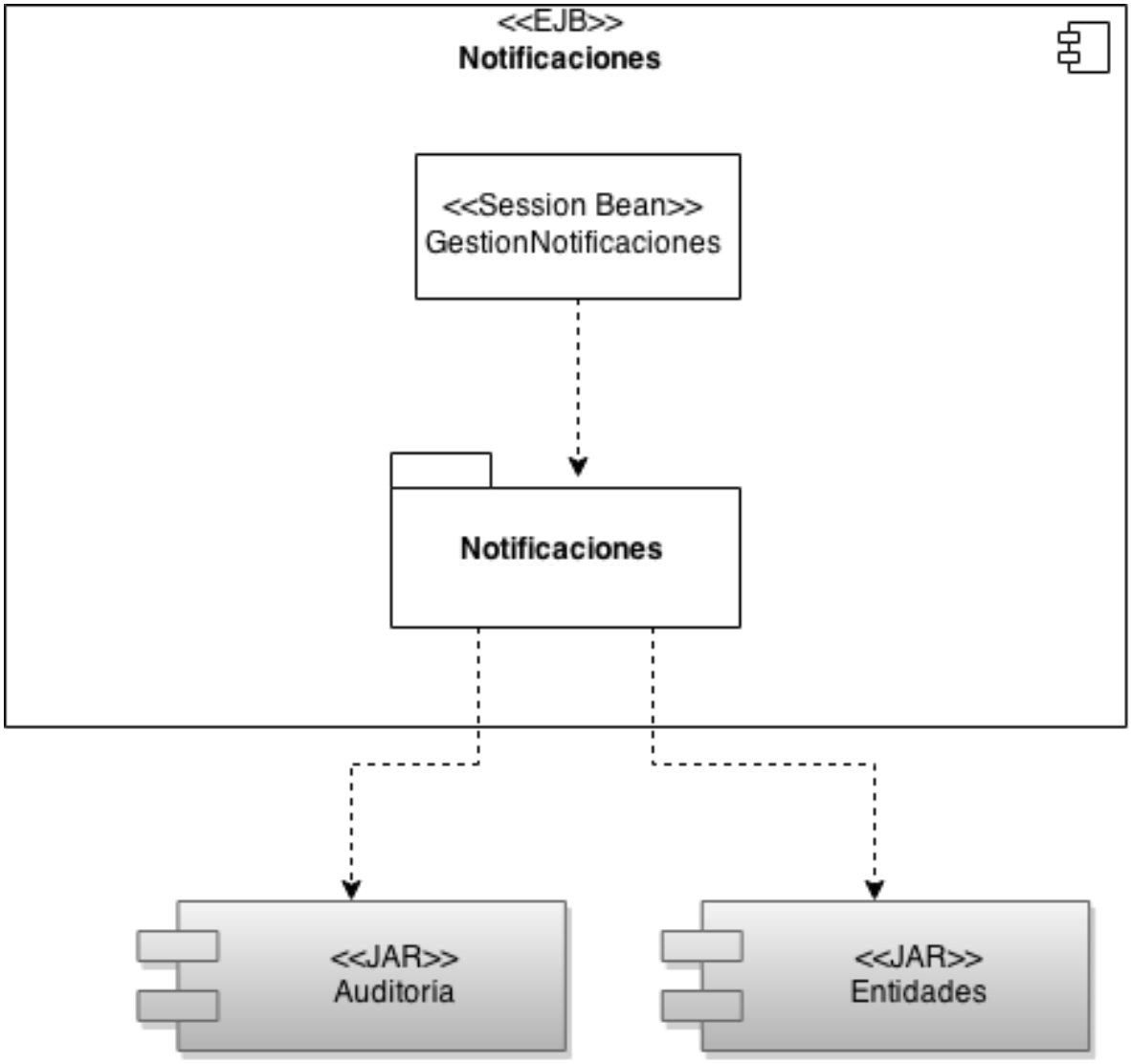
La información de los predios y sus suelos está estrechamente relacionada, por lo que se expone mediante un único *Session Bean*.

Contexto



EJB Notificaciones

Representación primaria



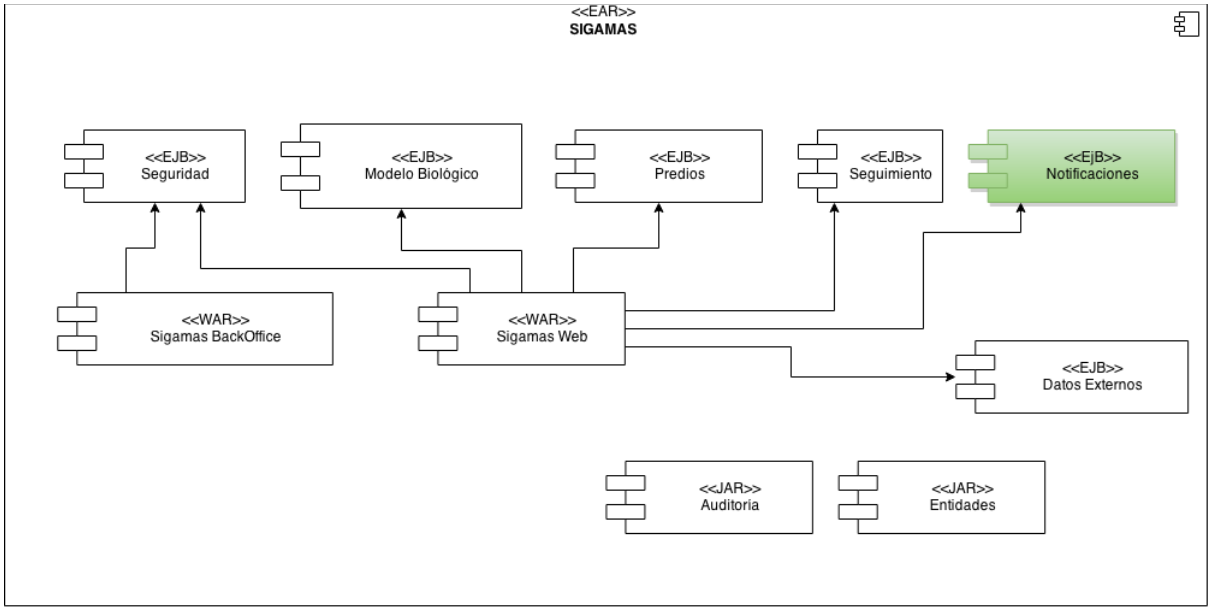
Catálogo de elementos

Elemento	Responsabilidades
GestionNotificaciones	*Session Bean* encargad de exponer las funcionalidades de la gestión de notificaciones, incluyendo la consulta de notificaciones pendientes y la modificación de estado de las notificaciones de pendientes a leídas
Notificaciones	Modulo que implementa la lógica de negocio del mecanismo de notificaciones

Justificación

Se decidió implementar un sistema de notificaciones propio sin la utilización de tecnologías de apoyo como por ejemplo JMS (Java *Message System*) que permite el uso de colas de mensajes. Esta decisión se tomó en vista de que no se utiliza la funcionalidad de mensajería asíncrona, y se evaluó que en este escenario es menos costoso implementar el sistema que utilizar JMS.

Contexto



EJB Datos Externos

Representación primaria

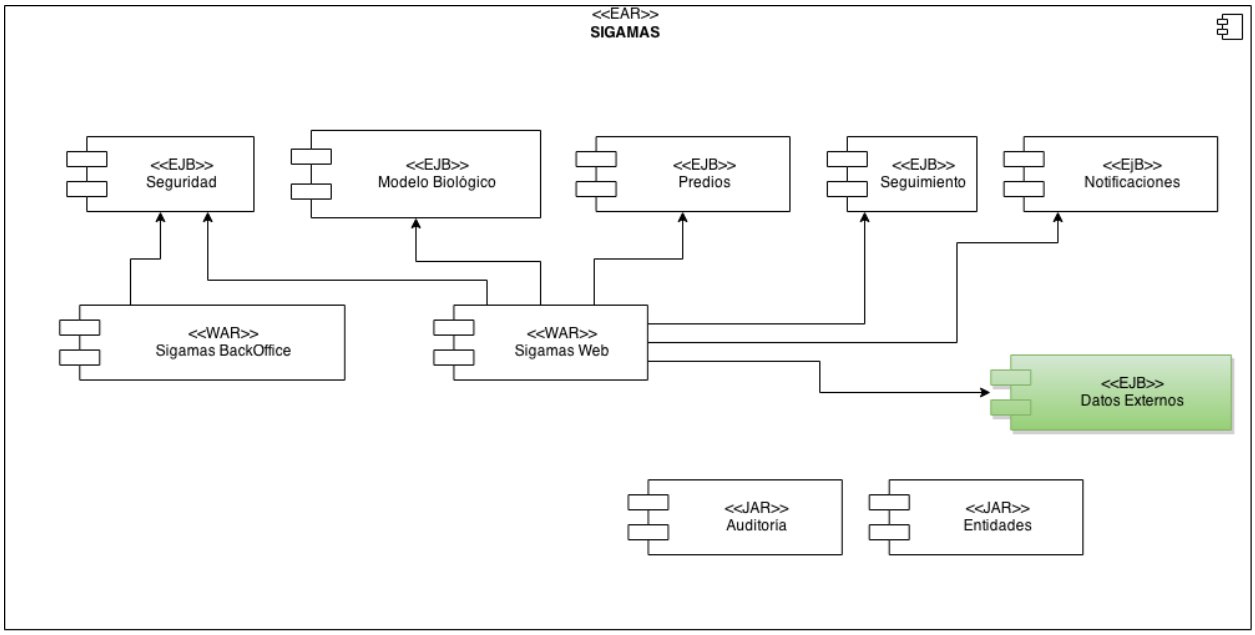
Catálogo de elementos

Elemento	Responsabilidades
DatosExternos	*Session Bean* que provee la funcionalidad de consulta de datos externos
Datos Externos	Modulo encargado de la transformación de datos y resolución de comunicación con las fuentes externas

Justificación

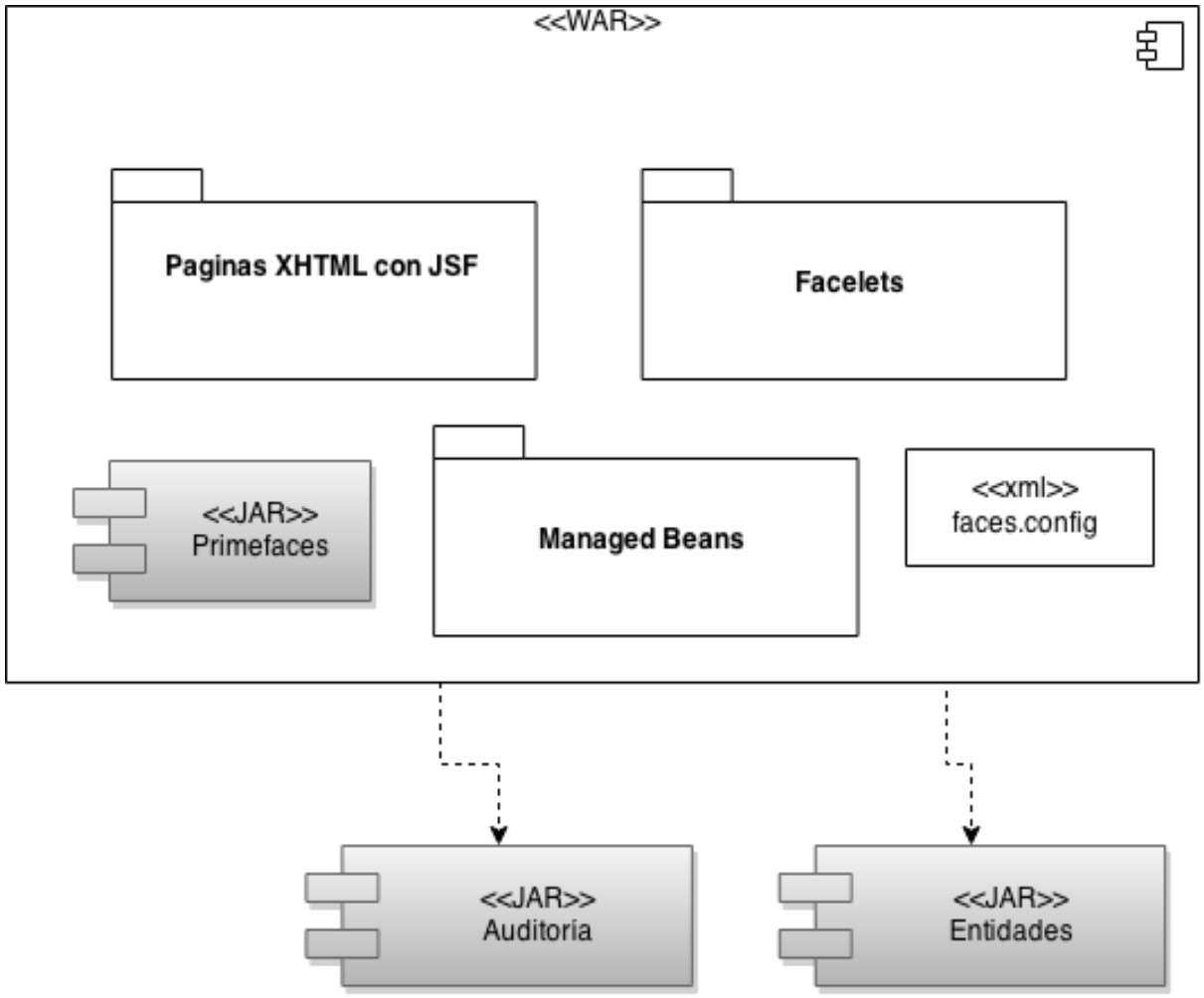
El modulo datos externos debe resolver para cada solicitud la información necesaria consultando en orígenes de datos externos a la aplicación. Al proveer esta única funcionalidad, se expone a través de un solo *Session Bean*.

Contexto



Componentes WAR

Representación primaria



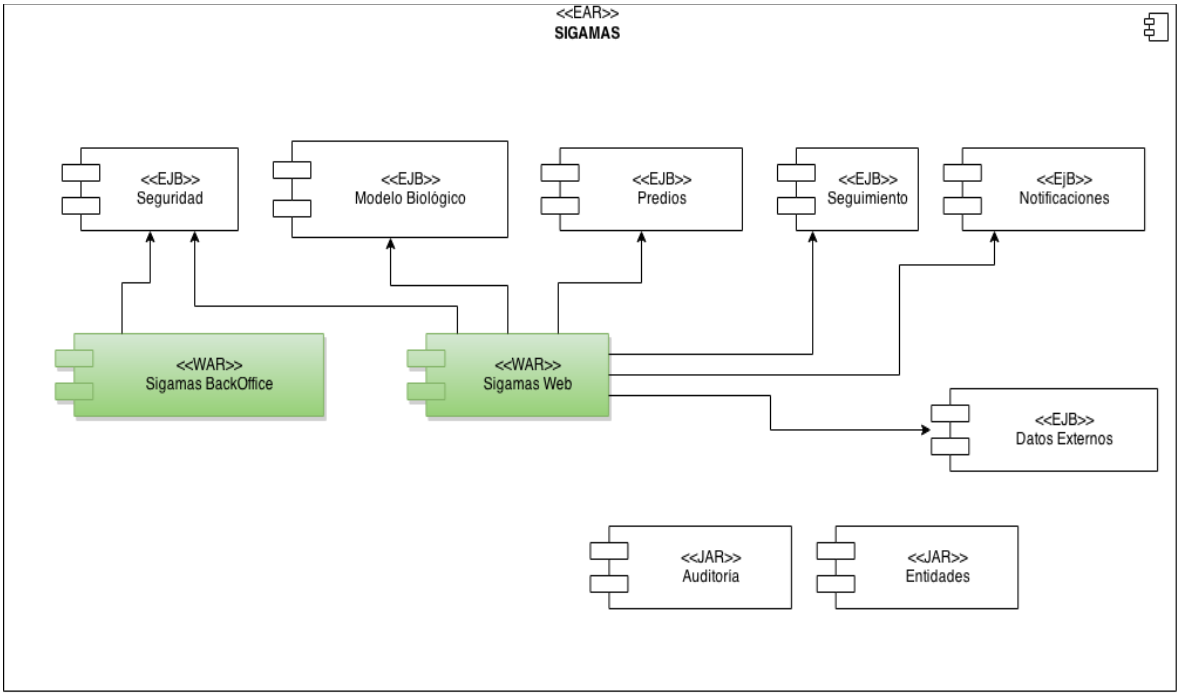
Catálogo de elementos

Elemento	**Responsabilidades**
Páginas XHTML con JSF	Paginas dinámicas codificadas en JSF (Java *Server Faces*).
Facelets	Paginas que utilizan la tecnología de Facelets para definir *templates* que permiten reutilizar componentes y acelerar el desarrollo.
Managed Beans	Tecnología de *Beans* gestionados por el contenedor
Primefaces	Jar con de Primefaces, que contiene una librería de componentes JSF mejorados con gestión AJAX automática y estilos intercambiables
Faces.config	Archivo de configuración de JSF que contiene entre otras configuraciones los casos de navegación

Justificación

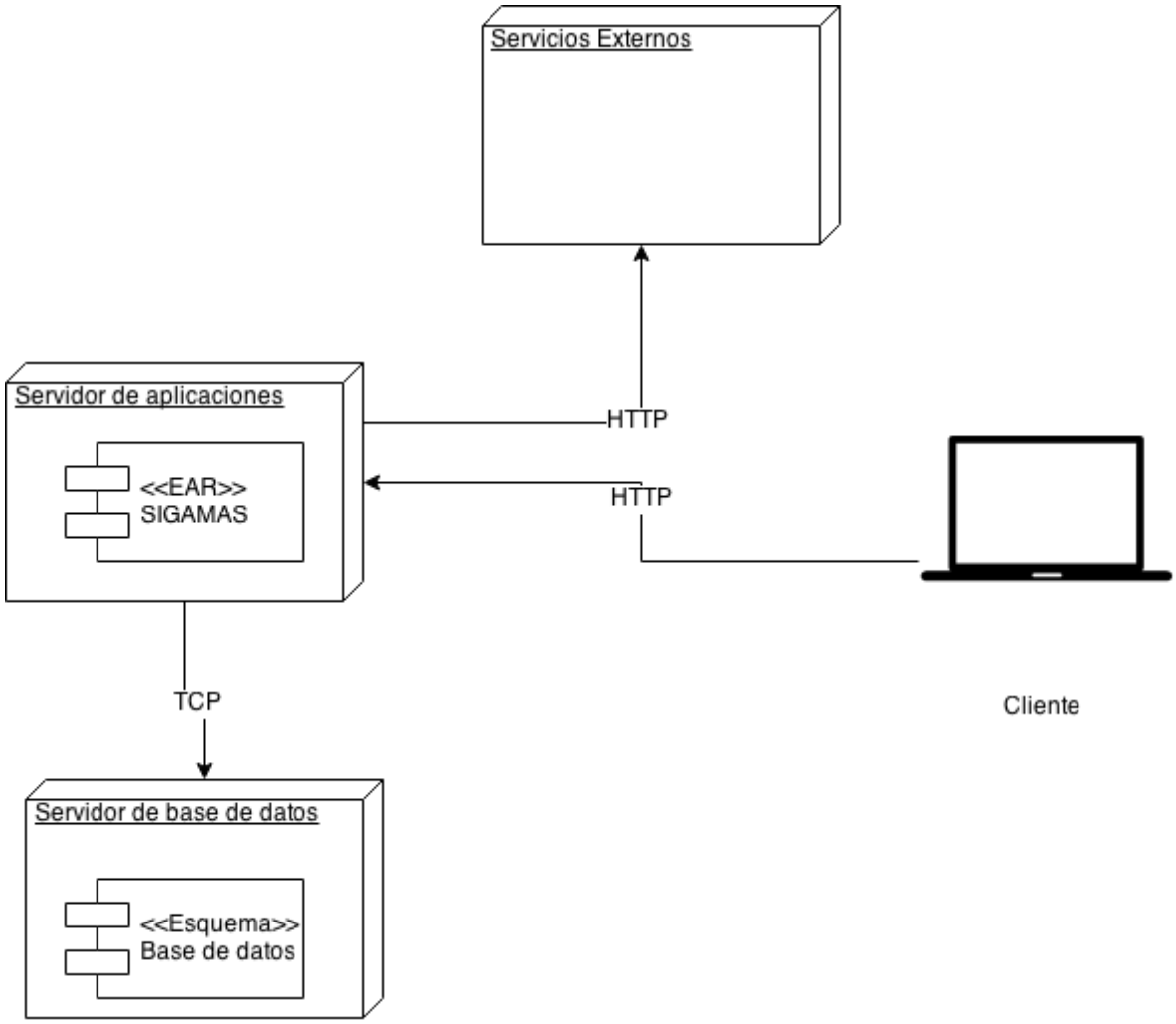
Esta disposición permite la reutilización de código, y la gestión de la navegación a través de casos de navegación JSF. De esta manera se favorece la modificabilidad del sistema y la seguridad del mismo, restringiendo los accesos mediante las mencionadas reglas de navegación. La librería de Primefaces facilita la creación de páginas dinámicas, dado que provee componentes que extienden los componentes básicos de JSF.

Contexto



Vista de despliegue

Representación primaria



Catálogo de elementos

Elemento	Responsabilidades
Servidor de Aplicaciones	Servidor con soporte JEE donde va a estar alojada la aplicación
Servidor de Base de datos	Servidor donde va a estar alojado el esquema de datos de la aplicación
Cliente	Cliente del sistema que se conecta a través de internet
Servicios externos	Servicios de datos que consume la aplicación
SIGAMAS	La aplicación SIGAMAS en si misma

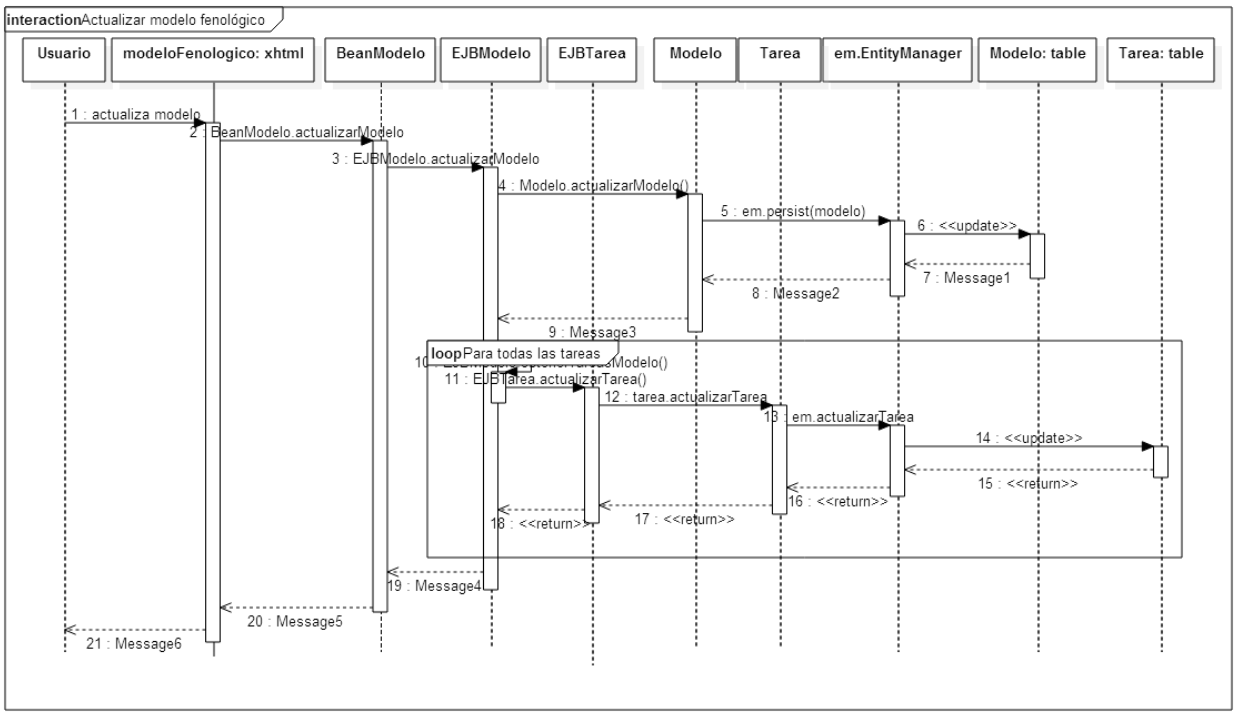
Justificación

Esta disposición de elementos representa las decisiones arquitectónicas tomadas en cuanto al modelo cliente servidor a través de internet.

Vistas de interacción

Actualización del modelo fenológico

Representación primaria

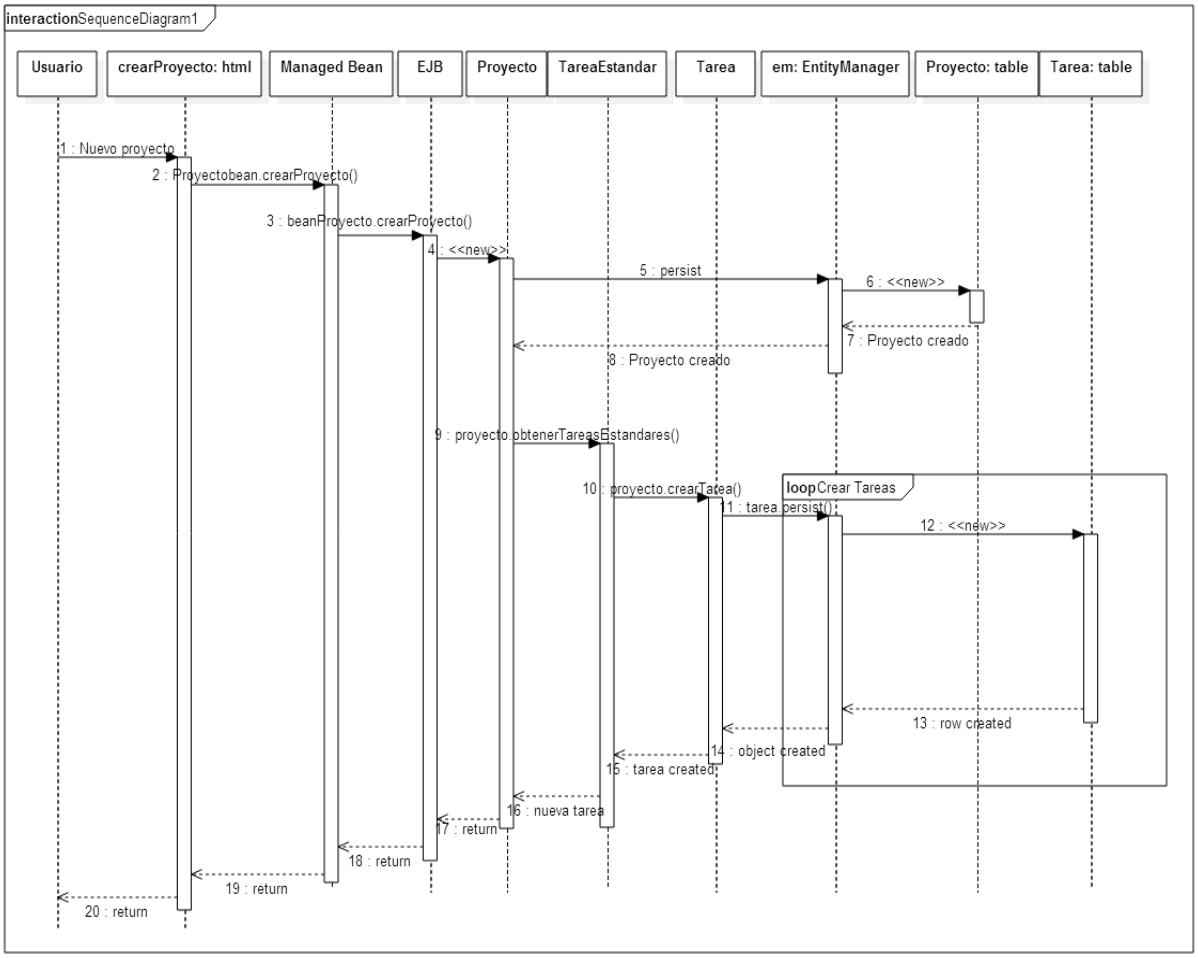


Justificación

El diagrama muestra la interacción que ocurre cuando el modelo biológico cambia de fase. Este actualiza el seguimiento del proyecto en sus tareas y agrega si fuera necesario las tareas que la fase del cultivo determine.

Crear Proyecto

Representación primaria

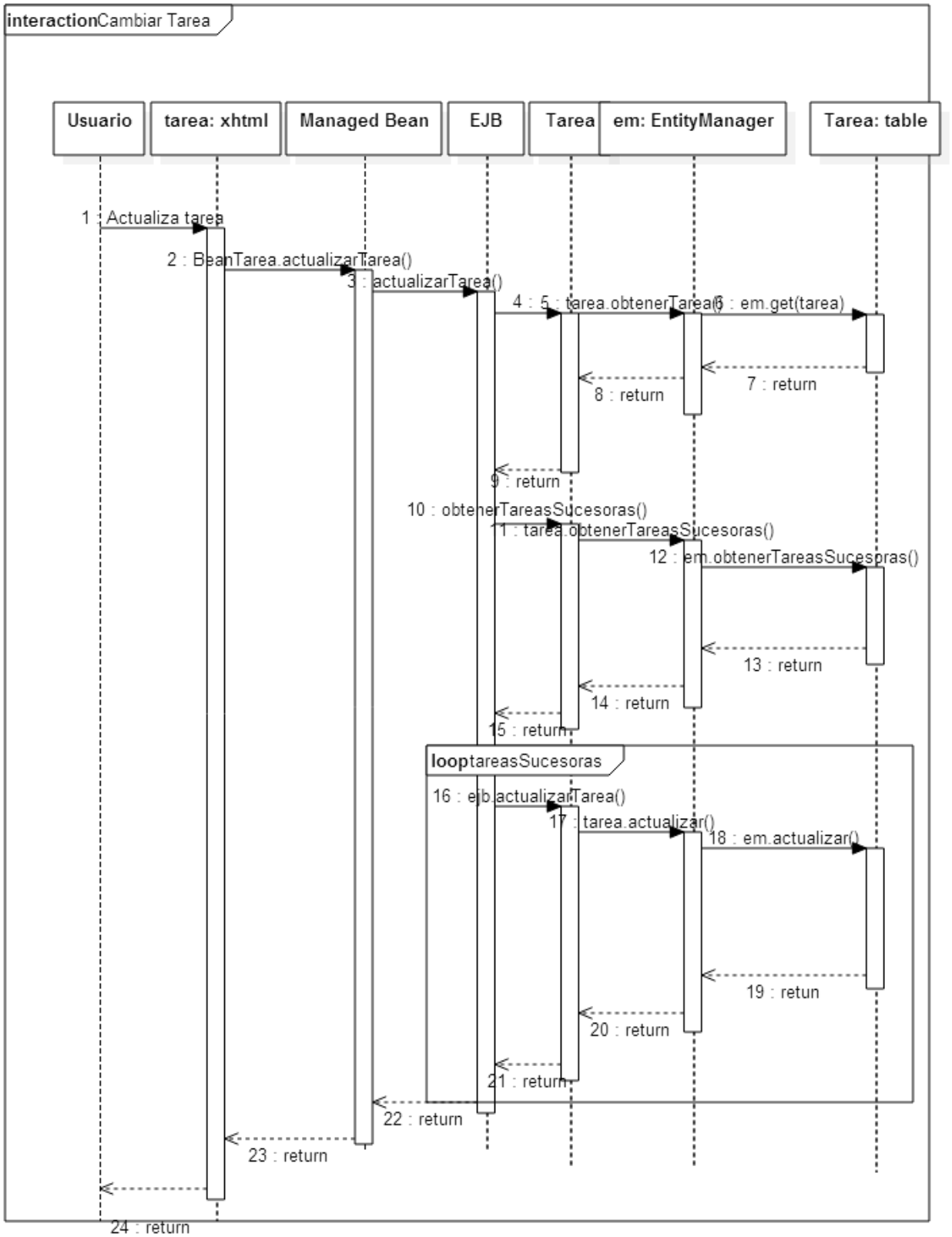


Justificación

Como se indicó anteriormente, la creación de un proyecto de cultivo es una tarea compleja, que amerita ser descrita en detalle.

Actualización de Tareas

Representación primaria



Tácticas y patrones utilizados

Se aplicaron distintas tácticas y patrones arquitectónicos para asegurar el cumplimiento de los atributos de calidad que se definieron, se listan a continuación los más relevantes según atributo de calidad:

Escalabilidad

Para asegurar la escalabilidad del sistema y que la performance se mantenga ante una demanda que se prevé incremental, la arquitectura definida se nutrió de tácticas específicas como incrementar la eficiencia de los recursos en el marco del control de demanda de recursos. Esto se logró mediante un cuidadoso uso de las operaciones críticas del sistema como el acceso a base de datos, minimizando la complejidad de las operaciones llevadas a cabo y optimizando las mismas mediante cálculo de complejidad de algoritmos y consultas a base de datos por ejemplo.

La tecnología de base, JEE (Java *Enterprise Edition*), provee también una gestión de recursos centralizada en los componentes EJB (*Enterprise* Java *Beans*), introduciendo concurrencia y manejo de la disponibilidad de recursos.

Disponibilidad
Para asegurar la disponibilidad y la integridad de los datos, se vale principalmente de un API del *Stack* JEE llamado JTA (Java *Transaction* API), que se encarga de encapsular en transacciones las operaciones llevadas a cabo en la capa de negocio permitiendo la vuelta atrás si estas fallan. Asimismo, los EJB (Enterprise Java *Beans*) en un contexto gestionado como lo es el de una aplicación JEE son instanciados por el contenedor JEE a demanda según un pool de EJBs que puede variar en tamaño para atender más solicitudes.

Seguridad
En materia de seguridad el sistema está provisto con una gestión de usuarios basada en roles. Fue diseñado de esta manera para introducir los beneficios de las técnicas de identificación de Actores, Autenticación de Actores y Acceso Limitado [REFERENCIA].

Se ideó un sistema de seguridad interno compuesto por usuarios, cuentas y roles. Determinado usuario puede tener asociadas varias cuentas y dentro de cada una de ellas tiene un rol particular.

Un usuario operando el sistema está restringido a los contenidos disponibles para el rol que tiene dentro de una cuenta, por lo que sus acciones quedan limitadas.

Accesibilidad
El sistema está basado en el acceso mediante internet, en una arquitectura cliente-servidor. En este marco, se implementan las funcionalidades en la capa de presentación siguiendo buenas prácticas de diseño responsivo, lo que permite el acceso desde cualquier dispositivo con un navegador web.

Modificabilidad
Como se mencionó anteriormente, la modificabilidad fue un atributo considerado crítico para el buen desarrollo del sistema. Se aplicaron varias técnicas así como buenas prácticas

para llegar a un producto que permitiera una gran flexibilidad a la hora de extender sus funcionalidades con el menor impacto posible. Se Aplicó táctica de reducción del tamaño de los módulos en conjunto con la incrementación de cohesión para llegar a piezas de Software minimales y auto contenidas, de modo que los cambios en estas se propaguen lo menos posible.

Se aprovechó también que la tecnología de base, JEE (Java *Enterprise Edition*), provee mecanismos para favorecer la técnica "*Defer Bindings*" dado que los contendores son quienes gestionan la creación y ciclo de vida de los recursos, como por ejemplo los EJB o *Managed Beans* mediante la inyección de dependencias usando la tecnología CDI (*Contexts and Dependency Injection*) de Java, que además favorece la reducción del acoplamiento.

Infraestructura sugerida

Se investigaron diferentes proveedores de IaaS (*Infrastructure* as a *Service*), dentro del marco de lo que son los PaaS (*Provider as a Services*) para resolver problemas con respecto a la disponibilidad dado que el sistema presenta características de SaaS (Software *as a Service*).

Como la versión que este proyecto genera no es una versión de producción, se sugiere alojarla en los servidores de Amazon, donde se realizó una prueba de concepto y no se encontraron mayores dificultades para desplegar la aplicación.

Evaluación de la arquitectura

Para la evaluación de la arquitectura se evaluó utilizar el modelo ATAM (*Architecture Tradeoff Analysis Method*). Este modelo de evaluación no exige que los evaluadores tengan conocimiento de los objetivos de negocio o la arquitectura. Sin embargo, encontramos que este modelo requiere un gran esfuerzo de trabajo y no es compatible con el proceso definido para la realización de este proyecto, dado que el tiempo que insume en cada iteración donde se modifique la arquitectura es significativo.

Se optó entonces por la validación con expertos y clientes, que permite mayor flexibilidad al no exigir un análisis tan abarcativo y no demanda demasiado tiempo de los evaluadores.

El cliente es el primero en validar cada versión de arquitectura, que luego se valida con un arquitecto experto quien analiza los aspectos más técnicos de la misma.

Plan de Mantenimiento

Se sugieren algunos lineamientos en cuanto al mantenimiento de la aplicación, que cubren requerimientos no funcionales que la arquitectura no resuelve por sí misma:

- *Backups* de la base de datos en forma diaria, que se mantengan por una semana.
- Verificación diaria de los niveles de servicio del servidor donde está alojada la aplicación.
- Revisión de los *logs* de auditoría en busca de mensajes de error.

Estos lineamientos se enmarcan dentro del aseguramiento de disponibilidad.

Resolución de Requerimientos No Funcionales

	Agrupación	Nombre	Solución
RNF1	Arquitectura	Modelo cliente-servidor	Definición de arquitectura
RNF2	Arquitectura	Cliente Aplicación Web	Definición de arquitectura
RNF3	Seguridad	Sesiones	Contenedores JEE
RNF4	Seguridad	*Login*	Diseño de la solución
RNF5	Seguridad	Encriptación de la información	No se realizo
RNF6	Seguridad	Conexiones seguras	No se realizo
RNF7	Auditoria	*Logs* del sistema	log4j
RNF8	Performance	Tiempos de respuesta	Optimizaciones
RNF9	Performance	Tiempos de procesamiento	Optimizaciones
RNF10	Performance	Tiempos de ejecución de consultas	Optimizaciones
RNF11	Performance	Tiempos de ejecución de reportes	Optimizaciones
RNF12	Capacidad	Capacidad de procesamiento	Contenedores JEE
RNF13	Capacidad	Almacenamiento	Definición de infraestructura
RNF14	Escalabilidad	Crecimiento previsto	Definición de infraestructura
RNF15	Escalabilidad	Previsión de crecimiento de base de datos	Definición de infraestructura
RNF16	Escalabilidad	Previsión de crecimiento de sistemas	Definición de

			infraestructura
RNF17	Disponibilidad	Horarios de operatividad	Plan de mantenimiento
RNF18	Disponibilidad	Lugares de operatividad	Definición de arquitectura
RNF19	Disponibilidad	Requerimientos de conexión	Definición de arquitectura
RNF20	Confiabilidad	Tiempo máximo caído	Plan de mantenimiento
RNF21	Confiabilidad	Tiempo máximo de recuperación	Plan de mantenimiento
RNF22	Confiabilidad	Técnicas para confiabilidad de los datos	JTA
RNF23	Integridad	Manejo de errores de Entrada/salida	Buenas prácticas de programación
RNF24	Integridad	Datos con fallas	Buenas prácticas de programación
RNF25	Recuperación	Tiempo de recuperación	Plan de mantenimiento
RNF26	Recuperación	Frecuencia de *Backups*	Plan de mantenimiento
RNF27	Recuperación	Táctica de *Backups*	Plan de mantenimiento
RNF28	Interoperabilidad	Intercomunicación con otras aplicaciones	*WebServices*
RNF29	Compatibilidad	Compatibilidad con distintos sistemas operativos	Buenas prácticas de programación
RNF30	Compatibilidad	Compatibilidad con distintos *Browser*	Buenas prácticas de programación
RNF31	Compatibilidad	Compatibilidad con distintas plataformas de Hardware	Buenas prácticas de programación

RNF32	Modificabilidad	Estándares de arquitectura que la permiten	Buenas prácticas de programación
RNF33	Modificabilidad	Estándares de diseño	Buenas prácticas de programación
RNF34	Modificabilidad	Estándares de codificación	Buenas prácticas de programación
RNF35	Usabilidad	*Look and Feel*	Buenas prácticas de Usabilidad
RNF36	Documentación	Requerimientos de documentación para cada audiencia	Buenas prácticas de Documentación
RNF37	Testeabilidad	Existencia de *Unit Test* para cada *Feature*	Definición de arquitectura
RNF38	Testeabilidad	Existencia de Pruebas de Integración	Plan de *Testing*
RNF39	Testeabilidad	*Record/Playback*	Herramientas de *Testing*
RNF40	Testeabilidad	*Sandbox*	Definición de arquitectura

REFERENCIAS BIBLIOGRÁFICAS

[1] L. Bass, P. Clements, R. Kazma, *Software Architecture in Practice*, USA: Addison-Wesley, 2012.