Universidad ORT Uruguay
Facultad de Ingeniería

S.C.E – Solución de Comunicación Empresarial

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

Diego Lussich - 144074
José Luis Obes - 152132
Rafael Etcheverry - 168490

Tutor: Marcelo Cagnani

2014

Declaración de autoría

Nosotros, José Luis Obes, Rafael Etcheverry y Diego Lussich, 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 final de carrera de Ingeniería en Sistemas.

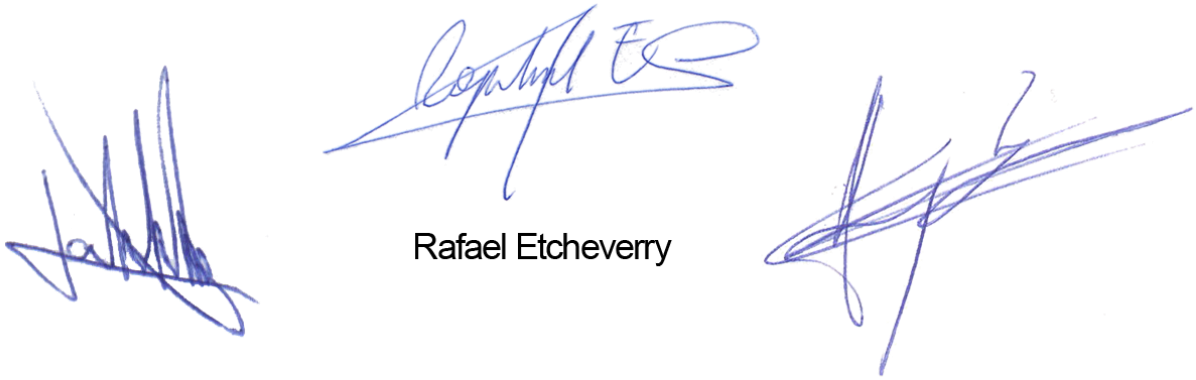
- Cuando hemos consultado el trabajo publicado por otros, lo hemos atribuido con claridad.

- Cuando hemos citado obras de otros, hemos indicado las fuentes. Con excepción de estas citas, la obra es enteramente nuestra.

- En la obra, hemos acusado recibo de las ayudas recibidas.

- Cuando la obra se basa en trabajo realizado conjuntamente con otros, hemos explicado claramente que fue contribuido por otros, y que fue contribuido por nosotros.

- Ninguna parte de este trabajo ha sido publicada previamente a su entrega, excepto donde se han realizado las aclaraciones correspondientes.



Rafael Etcheverry

José Luis Obes

Diego Lussich

Agradecimientos

En primer lugar, queremos agradecer a nuestro tutor Marcelo Cagnani por ayudarnos y guiarnos a lo largo de todo el proyecto, reuniéndose semanalmente con nosotros.

Al laboratorio ORT Software Factory y los docentes de la cátedra, que de una manera u otra nos ayudaron, ya sea como guía o verificación técnica experta en cuestiones ligadas directamente al proyecto.

En tercer lugar queremos agradecer a familiares y amigos quienes nos apoyaron de forma incondicional.

Agradecemos también, a los empleados de la empresa Moon Ideas quienes colaboraron directamente con el proyecto, en la ejecución de las pruebas del sistema.

Por ultimo pero no menos importante queremos agradecer a Matías Burgos, el referente de nuestro cliente, por apoyarnos en todo momento y colaborar para que el proyecto se llevara adelante.

Abstract

El proyecto surge a partir de una propuesta realizada por el cliente Moon Ideas, quien presentaba la necesidad de contar con una herramienta para mejorar la comunicación y la gestión interna de la empresa. Al analizar la posibilidad de dicha herramienta, se decidió ir más allá y crear una herramienta que además de ayudar en la gestión interna, permita acercar a la empresa a sus clientes y a otras empresas de interés.

Entre sus principales funcionalidades se encuentra la comunicación instantánea entre los empleados de una empresa, la comunicación instantánea entre los clientes de una empresa y sus empleados y el acercamiento entre empresas de interés común.

El producto hace foco en las plataformas móviles, por lo que fue de gran importancia crear versiones nativas para los principales sistemas (iOS y Android). A la vez se utilizaron tecnologías estándar para la creación del servidor (PHP) y la comunicación instantánea (XMPP).

El proyecto fue desarrollado principalmente en un marco de metodología ágil, utilizando conceptos de metodologías más tradicionales en aquellos aspectos a los cuales el desarrollo ágil no hace referencia. Se trabajó en comunicación constante con el cliente, validando con el mismo y evolucionando los requerimientos de manera de atender sus necesidades de forma óptima.

Este documento describe en detalle al producto construido y el proceso utilizado para su construcción.

Palabras Clave

Comunicación, empresarial, *chat,* mensajería, instantánea, iOS, Android, XMPP

Tabla de contenido

1. Glosario

A

Android: Sistema operativo para dispositivos móviles creado por la empresa Google.

API: Especificación de cómo interactúan dos componentes de software.

APNS: Servicio de notificación de Apple.

App Store: Tienda electrónica de la empresa Apple.

C

Criptografía asimétrica: Algoritmo de encriptación que involucra una llave publica y una llave privada.

Chat rooms: Sesiones de mensajería instantánea multiusuario (grupos).

Color coding: Permite identificar distintos grupos de palabras claves pertenecientes a un lenguaje de programación, según su color.

D

Dropbox: Servicio de alojamiento de archivos multiplataforma en la nube.

E

E-commerce: Comercio electrónico; intercambio de productos o servicios a través de internet.

ESRE – Documento de especificación de requerimientos.

F

Fantasy sport gaming: Tipo de juegos en los cuales el participante actúa como dueño de un equipo, el cual se basa en las estadísticas reales de jugadores o equipos de ligas profesionales.

FTP(File transfer protocol): Protocolo de red utilizado para transferir archivos desde una entidad a otra (generalmente de un cliente a un servidor) mediante una red basada en TCP, como por ejemplo internet.

Front : Término utilizado para referirse a la página web visible al usuario final de un sistema.

Feedback: Retroalimentación.

Framework: Estructura de software compuesta por componentes personalizables e intercambiables que se utilizan para el desarrollo de aplicaciones.

Freemium: Estrategia de cobro, en la cual un servicio o producto se ofrece de manera gratuita, pero se cobra por ciertos beneficios adicionales.

G

Gaming: Industria de los videojuegos.

GCM (Google Cloud Messaging): Servicio de notificación de Google.

Google Play: Tienda electrónica de la empresa Google.

Google Docs: Programa gratuito basado en web para crear documentos en línea con la posibilidad de colaborar en grupo.

H

Hosting: Alojamiento web; generalmente consiste en uno o más servidores que hospedan aplicaciones o páginas web.

HTTP: Protocolo utilizado por la web que define el formato de los mensajes enviados a través de esta.

I

iOS: Sistema operativo para dispositivos móviles creado por la empresa Apple.

IDE: Entorno de programación que ha sido empaquetado como un programa de aplicación; es decir, que consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica (GUI). Los IDEs pueden ser aplicaciones por sí solas o pueden ser parte de aplicaciones existentes.

Intellisense: Capacidad incluida en algunos ambientes de programación que acelera el proceso de codificación mediante mecanismos de autocompletado.

IP: Etiqueta numérica que identifica, de manera lógica y jerárquica, a una interfaz (elemento de comunicación/conexión) de un dispositivo dentro de una red que utilice el protocolo IP (Internet Protocol).

L

Layers: Patrón de diseño arquitectónico que procura organizar los distintos servicios de una aplicación en capas lógicas, agrupándolos según su funcionalidad.

M

MVC: Patrón arquitectónico que divide las aplicaciones en tres componentes interconectados, para lograr separar la información interna, de la presentación de dicha información.

P

Páginas Amarillas: Base de datos que lista empresas, organizadas según el tipo de producto o servicio que ofrecen. Generalmente provee información telefónica de las empresas.

Plugin: Módulo de software que se instala en un sistema para proveer cierta funcionalidad.

Polling: Técnica para obtener datos, en la cual una aplicación cliente realiza solicitudes periódicas a una aplicación remota para obtener la información necesaria.

Push Notifiacation: Tipo de comunicación en la que es el servidor el que inicia la petición al cliente (el móvil, por ejemplo) cuando tiene una información o notificación nueva, permitiendo un importante ahorro de recursos y tiempo respecto a la tecnología convencional *pull*.

Product backlog: Lista de user stories necesarias para construir el producto.

Product owner: Persona de contacto para evacuar dudas sobre los requerimientos.

R

REST: Estilo arquitectónico para exponer servicios mediante la web.

Registration id: Identificador provisto por Apple y Google para el uso de *push notifications*.

S

SSL (Secure Sockets Layer): Protocolo criptográfico diseñado para brindar seguridad en la comunicación por internet.

SSH (Secure Shell): Protocolo de red criptográfico, utilizado principalmente para la ejecución de comandos de manera remota.

Stateless: Patrón o principio de diseño arquitectónico que manifiesta que no se debe mantener información de estado para favorecer la escalabilidad del sistema.

Stakeholder: cualquier persona o entidad que es afectada o concernida por las actividades o la marcha de una organización.

SVN: Herramienta de control de versiones open source basada en un repositorio.

Scrum: Marco de trabajo para la gestión.

Scrum Master: Rol dentro de la metodología scrum, encargado de apoyar al equipo de desarrollo y al product owner.

Sprint: Ciclo de la metodología scrum, representa una iteración.

Sprint Backlog: Lista de tareas a realizar durante el sprint.

Software Factory: Fábrica de software.

U

User story: formato en el que generalmente se especifican los requerimientos para la utilización de metodologías ágiles.

T

Tester: la persona encargada de llevar a cabo las pruebas.

X

XMPP: Protocolo de comunicaciones basado en XML para la mensajería instantánea.

2. Introducción

2.1. Entorno conceptual de Software Factory

La primera decisión que se tuvo que tomar al comenzar este proyecto fue si se iba a trabajar con el laboratorio de investigación en sistemas de información (LISI) o con el laboratorio de Software Factory. Se decidió optar por el laboratorio de Software Factory, ya que este brinda apoyo en las distintas áreas del proceso de Ingeniería. Todo el equipo estuvo de acuerdo en que esta era la decisión correcta ya que la base del proyecto era la creación de una solución de software.

Las características de Software Factory son:

2.1.1. 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”

2.1.2. Visión

“Ser líderes en la generación de conocimiento sobre la producción de software de calidad. Ser un referente dentro de la Universidad ORT Uruguay, en el medio y la región, en la creación y aplicación de prácticas de ingeniería de software para la producción de productos de calidad”

2.1.3. Objetivos

Los objetivos de Software Factory son:

- Formar en la producción de software regida por un proceso, y en técnicas, herramientas y métodos de ingeniería de software.
- Promover el trabajo en equipo
- Transferencia de tecnologías a la industria

2.2. Objetivos del proyecto

Los objetivos del proyecto se pueden dividir en dos tipos:

Objetivos del producto:

- Desarrollar un producto de software que agregue valor y eficiencia a la comunicación empresarial, cumpliendo con todos los requisitos establecidos por el cliente.
- Implementar los módulos funcionales necesarios para obtener una primera versión de dicho producto.

Objetivos personales:

- Aplicar y desarrollar las distintas habilidades adquiridas a lo largo de la carrera de Ingeniería de Sistemas.
- Adquirir experiencia de trabajo en equipo
- Desarrollar un proceso o metodología de trabajo eficiente, que pueda ser reutilizable en futuros proyectos.
- Aprobar el proyecto de fin de Carrera para poder obtener el título de Ingeniero de Sistemas.

2.3. Descripción del cliente

Moon Ideas es una empresa de desarrollo de software Uruguaya, fundada hace tres años, en los cuales ha tenido un fuerte crecimiento.

La empresa fue creada por dos estudiantes de ORT, en torno a su propio proyecto de carrera, el juego de *fantasy sport gaming* ELSuperDT.com. Este juego, el cual fue adoptado por la empresa Tenfield, tuvo un gran éxito (cuenta con miles de usuarios) y sirvió como plataforma de despegue para la empresa.

Hoy en día, Moon Ideas ha tenido un gran crecimiento y se dedica a desarrollar todo tipo de aplicaciones, para múltiples clientes. Aunque desarrolla productos para un numero variado de sectores, como por ejemplo el sector del *gaming* o de las aplicaciones para Facebook, su actividad principal gira en torno a la creación de

soluciones de *e-commerce,* tanto para la web como para los dispositivos móviles. Cuenta con oficinas en Montevideo, en las cuales trabajan decenas de empleados.

2.4. Descripción del equipo

El equipo está formado por tres estudiantes de la carrera de Ingeniería de Sistemas: Rafael Etcheverry, Diego Lussich y José Luis Obes.

Rafael y José Luis trabajan juntos desde hace un tiempo largo; se dedican principalmente al desarrollo de aplicaciones web y aplicaciones móviles. Diego trabaja en el área de soporte técnico, y conoció a Rafael cuando hicieron juntos un trabajo obligatorio en ORT.

En el comienzo del proyecto, se repartieron las responsabilidades de cada área del proyecto según su experiencia y afinidad con cada una de ellas. A continuación se muestran las áreas de las cuales se encargaron cada uno de los integrantes del equipo:

Rol/Área	Encargado
Gerente	Diego Lussich
Arquitecto	José Luis Obes
Ingeniero Requerimientos	Rafael Etcheverry
SQA	Rafael Etcheverry
SCM	Rafael Etcheverry
Desarrollo PHP	Todo el equipo
Desarrollo iOS	José Luis Obes
Desarrollo Android	Rafael Etcheverry
Testing	Diego Lussich

Figura 2.1 - Responsables de área

Si bien existen encargados para cada área, estos se designaron para que se aseguren de que se llevan a cabo todas las tareas del área, sin ser necesariamente ellos quienes las realizaron. En la mayoría de las áreas, los integrantes del equipo trabajaron de manera conjunta, aportando todos para realizar las tareas necesarias.

El tutor del proyecto fue el Lic. Marcelo Cagnani.

2.5. Motivación

La motivación para realizar este proyecto surge a partir de la oportunidad presentada por el cliente, la cual se explica en forma detallada más adelante en este documento. Una vez que se presentó la oportunidad, los integrantes del equipo se vieron motivados a llevar a cabo este proyecto por las siguientes razones:

- La propuesta planteaba crear una aplicación de mensajería instantánea, similar a otras aplicaciones utilizadas por todos los integrantes del equipo; estas aplicaciones son de las más utilizadas en el planeta, con cientos de millones de usuarios [1]. Debido a la popularidad de dichas aplicaciones, los miembros del equipo se vieron muy motivados para aprender cómo éstas funcionan y para crear una propia, a medida para un cliente.

- Llevar a cabo un desarrollo a medida para un cliente, realizando todas las tareas necesarias, desde el trato con dicho cliente hasta el desarrollo mismo, es algo que ninguno de los integrantes había realizado anteriormente y por lo tanto se vieron entusiasmados con esta oportunidad.

- La posibilidad de mantener el vínculo con el cliente y continuar con el desarrollo de futuras versiones en caso de que se logre crear un producto satisfactorio.

2.6. Descripción del documento y estructura del CD

2.6.1. Descripción y organización del documento

El documento está dividido en trece capítulos, cada uno de los cuales se describe a continuación:

Glosario: En esta sección se presenta un listado de palabras y siglas, junto con su significado, las cuales figuran en varias ocasiones a lo largo del documento, y son claves para la comprensión del mismo.

Introducción: En este capítulo se hace una breve descripción del cliente, el equipo y los objetivos y motivaciones que el equipo tuvo para la realización del proyecto.

Planteamiento del problema: Plantea la situación actual del cliente, sus necesidades y porque era necesario contar con el nuevo producto.

Descripción de la solución: Este capítulo describe la solución planteada para resolver el problema presentado, detallando el producto a construir y sus principales características.

Proceso y ciclo de vida: Este capítulo, explica que ciclo de vida se utilizó para el proyecto y en qué orden cronológico se fueron realizando las actividades de cada área de ingeniería de software.

Ingeniería de requerimientos: Capítulo destinado a explicar las distintas etapas del proceso aplicado para llegar a los requerimientos del sistema.

Arquitectura: En este capítulo se describe la arquitectura diseñada para la construcción del sistema, indicando los atributos de calidad que dicha arquitectura contempla, las principales decisiones tomadas para lograr satisfacer estos atributos y las tecnologías utilizadas para el desarrollo del sistema.

Aseguramiento de la calidad: Capítulo orientado a describir los objetivos de calidad y las actividades se realizaron para lograr cada uno de estos objetivos. Se definen métricas las cuales permiten realizar seguimientos de los objetivos de calidad. Para cerrar el capítulo se explica a qué conclusiones se llegó, basándose en las métricas recolectadas.

Actividades de pruebas: Describe los diferentes tipos de pruebas realizadas, como se llevaron a cabo dichas pruebas y que importancia tuvieron para el proyecto.

Gestión de la configuración: Este capítulo muestra cómo se gestionaron todos los recursos digitales relacionados al proyecto, pasando por código, cálculos, gráficas, imágenes, documentos, etc.

Gestión de proyecto: Describe y justifica que actividades se realizaron para llevar a cabo la gestión del proyecto. La metodología utilizada, métricas utilizadas para controlar el proceso, riesgos detectados, que problemas se encontraron y como se resolvieron.

Conclusiones: Reflexión y principales conclusiones generadas a partir del proyecto.

Bibliografía: Se adjunta la fuente del material bibliográfico consultado.

Anexos: Se anexan documentos complementarios que formaron parte del proyecto.

2.6.2. Estructura del CD

El CD cuenta con un único directorio llamado “Documentacion”, en el cual se encuentra la documentación en versión PDF y versión DOCX. Los archivos contenidos son:

- Entrega_SolucionDeComunicacionEmpresarial.pdf
- Entrega_SolucionDeComunicacionEmpresarial.docx

Cada uno de estos archivos contiene el cuerpo principal del trabajo, junto con los anexos.

3. Planteamiento del problema

3.1. Contexto del problema

Hoy en día existen varias alternativas de comunicación, la mayoría de las cuales son muy eficientes y utilizadas de forma exitosa en el día a día. Sin embargo, en los últimos tiempos la que ha tomado mayor importancia y ha sobrepasado al resto de las vías de comunicación es la mensajería instantánea [2], para la cual ya existen varias alternativas exitosas en el mercado.

El cliente, en un principio, necesitaba contar con un sistema de mensajería instantánea para lograr una buena comunicación interna entre sus empleados, pero no deseaba utilizar ninguna de las opciones que ofrece el mercado por las siguientes razones:

- Buscaba contar con una herramienta de comunicación propia para asegurarse de no tener que pagar licencias ni estar sujeto a términos y condiciones, ni en el presente ni en el futuro.

- Creando una herramienta de mensajería instantánea propia, generaría un sentido de pertenencia en los empleados hacia la empresa. También al tratarse de una empresa de desarrollo de software, a ésta le interesaba mostrar el tipo de aplicaciones que eran capaces de construir, a sus empleados.

En una segunda instancia, el cliente notó que no solo necesitaba una herramienta para la comunicación interna entre sus empleados, sino que también necesitaba que esta herramienta facilitara la comunicación externa con los clientes. En otras palabras, la herramienta debería permitir que los clientes de Moon Ideas se puedan comunicar de manera directa (instantáneamente) con los empleados de la empresa para realizar consultas técnicas o de otro tipo. La idea era que la herramienta permita brindar una atención al cliente constante, promoviendo así el espíritu de transparencia y confiabilidad que define a Moon Ideas.

Una vez que se había definido la idea de la herramienta a desarrollar, según las funcionalidades mencionadas anteriormente (comunicación interna y externa), ocurrió un evento que cambió la dirección que iba a tomar el proyecto. En conversaciones con uno de sus clientes (en las cuales surgió el tema de la nueva herramienta de comunicación a desarrollar), los directivos de Moon Ideas notaron que dicho cliente también estaba interesado en contar con una herramienta de este tipo, la cual le permitiera ponerse en contacto directo con sus propios clientes. Se decidió entonces cambiar el enfoque del desarrollo; ya no sería una herramienta disponible únicamente para el uso interno, sino que Moon Ideas la ofrecería de

manera gratuita y opcional a sus clientes, principalmente a aquellos clientes que contrataran sus servicios para desarrollar sitios de *e-commerce*.

Por último, al ser una herramienta con potencial para ser utilizada por múltiples empresas (en un principio por todos los clientes de Moon Ideas), se decidió que ésta debería brindar funcionalidad para que las distintas empresas que la utilicen sean capaces de comunicarse entre ellas. De esta manera, distintas empresas que utilicen la herramienta serían capaces de intercambiar sus servicios contactándose de forma inmediata.

En resumen, Moon Ideas necesitaba desarrollar una herramienta que le brinde a cualquier empresa que la utilice, las siguientes tres formas de comunicación instantánea:

- Interna entre sus propios empleados
- Externa con sus propios clientes, para poder brindar soporte
- Externa con otras empresas para poder intercambiar servicios

Es entonces, a partir de estas necesidades del cliente que surge la idea del producto a crear.

3.2. Tipos de usuario y necesidades

Cualquier empresa es un usuario potencial de este producto, ya que la necesidad de tener una comunicación eficiente no es específica a algún sector industrial en particular, sino que es de interés común para todos los sectores. Se consideró a una empresa como cualquier organización que persigue un fin comercial. Cuando se habla de empresas, se hace referencia a todo tipo de empresa, sin importar su rubro o tamaño.

Sin embargo, las empresas que tienen mayor factibilidad para ser potenciales usuarios son aquella que tienen las siguientes necesidades:

- Empresas que no cuentan con ninguna solución de comunicación.
- Empresas que necesitan mejorar una o varias de las formas de comunicación expresadas en la sección anterior.

- Empresas que tengan dificultades para encontrar otras entidades a las cuales ofrecerles sus servicios.

- Empresas que tengan dificultades en encontrar otras empresas a las cuales solicitarles servicios específicos.

Si bien toda empresa es una potencial fuente de usuarios, tras un análisis, se detectó que es más factible que las pequeñas y medianas empresas le encuentren utilidad a nuestro sistema, que las empresas de mayor tamaño. Esto se debe principalmente a la diferencia de poder adquisitivo, ya que las grandes empresas probablemente cuenten con otras herramientas elaboradas de comunicación, mientras que otras empresas de menor porte pueden aprovechar el hecho de que el producto desarrollado para el proyecto es en su mayoría gratuito (*freemium*).

A pesar de lo expresado anteriormente en cuanto al tipo de empresa que se evaluó como más propensa a utilizar la herramienta, se confió en que el producto brinda una serie de funcionalidades que ninguna otra herramienta ofrece en conjunto, en una única aplicación, y por lo tanto toda empresa es un potencial usuario.

Hasta el momento, para referirse al usuario se habló siempre de empresas, pero en realidad los verdaderos usuarios no son las empresas, sino que son las personas asociadas a la empresa, es decir los empleados de la empresa misma, sus clientes, y los empleados de otras empresas. Para ver como cada uno de ellos interactúa con el sistema, referirse a la sección 4.1: usuarios del sistema identificados.

3.3. Restricciones

Existen ciertas restricciones que encaminaron al equipo a tomar ciertas decisiones a la hora de diseñar la solución. Todas estas restricciones fueron impuestas por el cliente. A continuación se listan las más relevantes:

- No se pueden utilizar soluciones existentes de comunicación que estén bajo algún tipo de licencia comercial, ya que la idea es comercializar el producto en el futuro y esto lógicamente podría causar problemas.

- La solución deberá ser una aplicación móvil gratuita (en primera instancia, luego ciertas características deberán ser pagas), la cual podrá ser descargada tanto desde la App Store para dispositivos móviles que utilicen la plataforma iOS como desde Google Play para dispositivos móviles que utilicen la plataforma Android. Una vez descargada, solo usuarios autorizados mediante el módulo de administración podrán acceder a su funcionalidad.

- El código del servidor deberá ser programado en un lenguaje conocido por el cliente, preferentemente PHP, ya que es posible que un equipo interno del cliente haga modificaciones en el futuro.

- La solución debería ser un desarrollo a medida, es decir las decisiones finales acerca del contenido de la solución siempre serian tomadas por el cliente.

En cuanto al último punto del listado anterior, hay destacar que por más que la decisión final fue siempre del cliente, este le dio al equipo total libertad para presentar ideas y sugerencias propias. En un principio el cliente presentó un esqueleto de las funcionalidades que él pretendía para la aplicación. El equipo tomó este esqueleto, aportó sus propias ideas y mejoras, y se lo volvió a presentar al cliente. De ahí en adelante las decisiones de implementación de las distintas funcionalidades se hicieron en conjunto, aunque siempre con la palabra del cliente como palabra final.

4. Descripción de la solución

En este capítulo se explica en detalle cual es la solución que se diseñó para enfrentar la problemática planteada en el capítulo anterior. Se describen cuáles son los usuarios que interactúan con el sistema desde un punto de vista funcional y se detallan las funcionalidades. Pero primero, para que el lector tenga una idea general de cómo se utiliza el sistema, se presenta un escenario de uso del mismo.

4.1. Escenario de uso

La empresa de venta de zapatos Mamoon Shoes decide adquirir el producto, ya que necesita una herramienta para acabar con sus problemas de comunicación. Una vez adquirido el producto, ingresa al sistema de administración y crea nuevos usuarios para sus empleados. Éstos se bajan la aplicación móvil e ingresan con sus datos de usuario.

Inmediatamente, la empresa resuelve su primer problema de comunicación: mantener en contacto a todos los vendedores (empleados) que están esparcidos a lo largo de las distintas tiendas del país. Esto queda resuelto ya que ahora pueden utilizar los servicios de mensajería instantánea de la aplicación móvil, ya sea a nivel individual o grupal.

También necesitan una forma de atender las consultas de los clientes, ya que estos están preguntando constantemente por la disponibilidad de ciertos talles y modelos de zapatos, y el único operador telefónico disponible (por razones económicas no es posible contar con un *call center*) no da abasto. La herramienta soluciona este problema, ya que al adquirirla se le instala un módulo de comunicación a la empresa en su sitio web (si así lo desean). Los clientes de Mamoon Shoes pueden entonces entrar a su sitio web y realizar las consultas allí mismo. Estas consultas irán directamente a los dispositivos móviles de los empleados de Mamoon Shoes, y el que decida atenderla podrá mantener una conversación con el cliente y contestar todas sus dudas.

El otro gran problema de Mamoon Shoes es que no cuenta con los medios necesarios para realizar envíos por sí mismo, y no conoce empresas que se dediquen a esa tarea. Ahora con el nuevo sistema, rápidamente puede encontrar una compañía de envíos, ya que la aplicación móvil permite que Mamoon Shoes se comunique con otras empresas asociadas al sistema, varias de las cuales figuraban bajo el rubro de envíos. Además tuvo la bonificación de que varias tiendas de ropa asociadas al sistema, que estaban interesadas en vender zapatos de otras marcas, se contactaron con Mamoon Shoes para saber más sobre sus productos y ver si les interesaba llegar a un acuerdo comercial.

El ejemplo anterior muestra las distintas formas en que una empresa puede utilizar el sistema. En las siguientes secciones se explica cómo está formado el sistema desde un punto de vista funcional.

4.2. Usuarios del sistema identificados

Se detectaron los siguientes tipos de usuarios:

Súper Administradores: Los súper administradores son usuarios que pueden ser creados únicamente por nuestro cliente Moon Ideas. Estos tienen como única función el registro y configuración inicial de las nuevas empresas en el sistema.

Empresas: Son aquellas empresas que cuentan con el sistema y están registradas en el mismo. Pueden ser empresas que contratan los servicios de Moon Ideas para un desarrollo web, y desean incluir el sistema de comunicación, o pueden ser empresas que desean adquirir únicamente el sistema de comunicación para integrarlo con sus propios sistemas. Una vez que la empresa es registrada, se crea una cuenta de administración, la cual se utiliza para configurar las opciones de dicha empresa y agregar nuevas cuentas de usuario.

Administradores: Son los usuarios de una empresa que tiene privilegios especiales para acceder al módulo de administración, modificar las opciones de configuración de la empresa y agregar nuevos usuarios. Los administradores son empleados de una empresa cliente que están encargados de administrar el sistema de comunicación. El usuario de tipo administrador es el que tiene los privilegios de más alto nivel, es decir tiene permiso para acceder a todas las funcionalidades que se le brindan al resto de los usuarios.

Usuarios básicos: Son los usuarios que cuentan con los privilegios básicos de la aplicación. Utilizan el sistema para comunicarse con otros usuarios pertenecientes a su misma empresa. Este tipo de usuario representa al empleado común de la empresa que solo utilizaría el sistema para la comunicación interna con sus colegas.

Operadores: Son usuarios especiales de la empresa. Además de contar con los privilegios del usuario común, estos serán responsables de contestar los pedidos de comunicación generados por los usuarios web. En otras palabras, el usuario de tipo operador, utiliza la aplicación tanto para la comunicación interna (de la misma forma que el usuario común), como externa (atiende solicitudes de consulta de los usuarios web).

Referente: El usuario referente es aquél con el cual los usuarios de otras empresas se comunican para consultar acerca de sus servicios. Es decir si un usuario de un

empresa ‘A’ desea comunicarse con la empresa ‘B’, se comunicará con el usuario de tipo referente de la empresa ‘B’. Este tipo de usuario también tiene acceso a las funcionalidades de los operadores y los usuarios básicos.

Usuarios web: Son usuarios anónimos (clientes o potenciales clientes de una empresa registrada) que desean comunicarse con una empresa para realizar cualquier tipo de consulta. Estos usuarios interactúan con el sistema a través de los sitios web de las empresas registradas. Cuando generan consultas en la web, estás son atendidas por los usuarios de tipo operador, quienes las reciben en sus dispositivos móviles. Son usuarios especiales, ya que no cuentan con permisos para utilizar las aplicaciones móviles, sino que interactúan con el sistema de manera anónima a través de la web, sin necesidad de registrarse.

4.3. Descripción funcional

Como mencionamos anteriormente, la solución es una herramienta de comunicación basada en mensajería instantánea para dispositivos móviles. Sin embargo ciertas acciones se realizan en la web a través de un *web browser*, por lo que la solución total es un híbrido entre aplicación móvil y aplicación web. Para facilitar la comprensión de las distintas capacidades que ofrece el sistema, éstas fueron agrupadas en módulos. Cada módulo tiene un conjunto de funcionalidades asociadas que brindan un servicio específico.

La siguiente figura muestra un panorama general de los módulos que forman la totalidad del sistema:

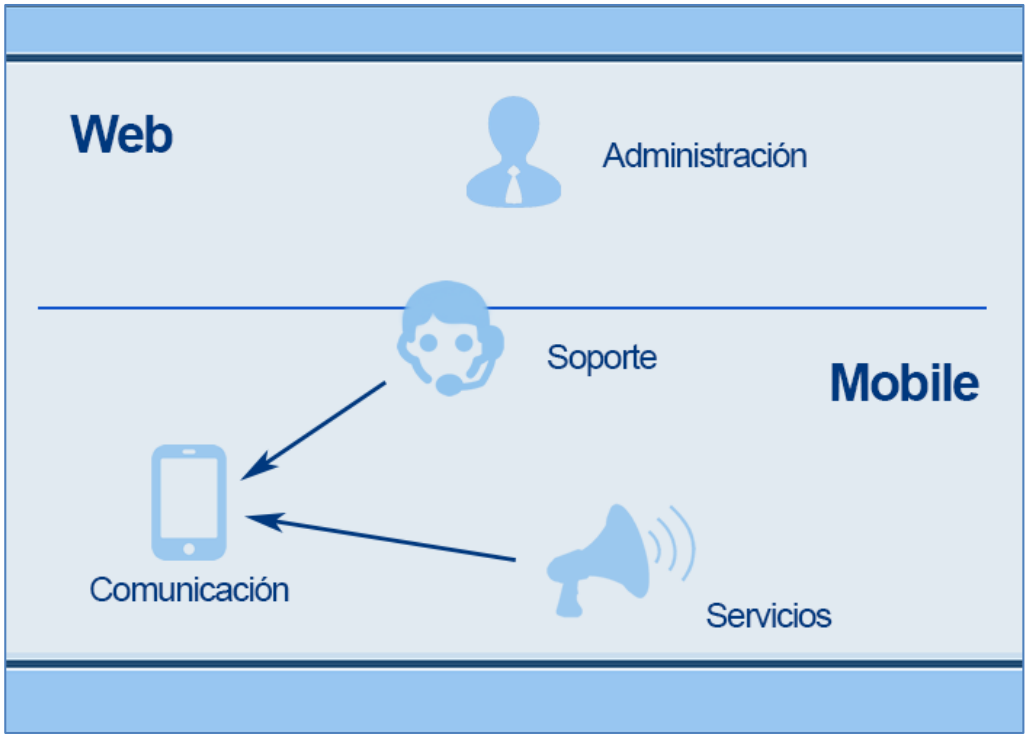


Figura 4.1 – Panorama general de los módulos del sistema

A continuación se detalla cada uno de estos módulos, indicando que funcionalidad brinda, como se utiliza y que tipos de usuarios interactúan con él:

4.3.1. Administración

Figura 4.2 - Icono representativo del módulo de administración

El módulo de administración es utilizado por los usuarios para administrar (crear, modificar o eliminar) las cuentas de otros usuarios pertenecientes a su misma empresa y para editar las opciones configurables de dicha empresa. Hay que destacar que este es el único módulo de la solución que está completamente en el ambiente web, es decir que solo se puede ingresar a él mediante un *web browser.*

Para ingresar al módulo de administración, un usuario debe ingresar al sitio de administración y presentar sus credenciales. Es importante saber que únicamente usuarios del tipo Administrador tienen los privilegios necesarios para acceder a este

módulo. La figura 2 muestra la pantalla de acceso al módulo de administración, donde el Administrador debe proporcionar un usuario y contraseña.

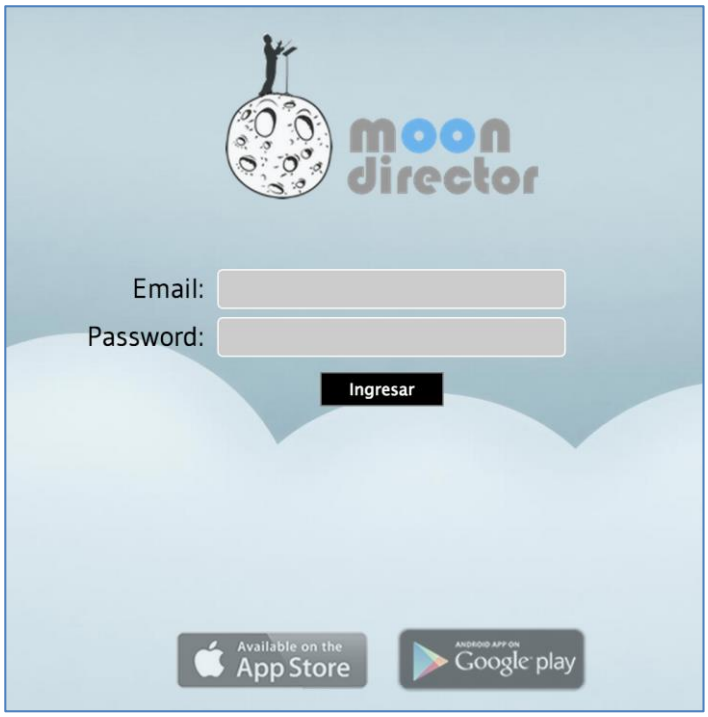


Figura 4.3 – Acceso a sitio de administración

Una vez que el usuario presenta las credenciales necesarias, se le concede el acceso al módulo de administración, en el cual puede realizar las acciones pertinentes en relación a su empresa. La siguiente figura ilustra el módulo de administración:

Figura 4.4 – Módulo de administración, listado de usuarios

El módulo de administración es el único modulo que no agrega funcionalidad asociada al cometido principal de la solución, es decir, no brinda capacidades de comunicación.

4.3.2. Comunicación

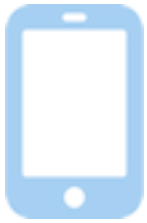


Figura 4.5 – Icono representativo del módulo de comunicación

El módulo de comunicación es el módulo central de la aplicación, es decir brinda la funcionalidad básica a la cual todos los tipos de usuarios tienen acceso.

Este módulo permite que los usuarios de una empresa se comuniquen con otros usuarios de su misma empresa, ya sea en conversaciones de uno a uno o en conversaciones en grupo.

Para iniciar una conversación de uno a uno, y para promover un ambiente no invasivo, se decidió que el usuario debe realizar una solicitud de contacto inicial. Una

vez que la solicitud de contacto es aceptada, se establece la relación de contacto entre los participantes y de manera automática se genera una conversación entre ellos.

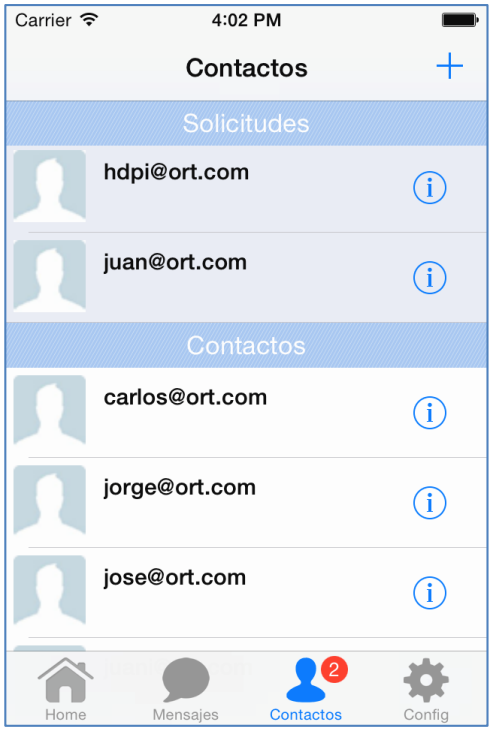


Figura 4.5 – Listado de contactos y solicitudes de contacto

Cualquier usuario puede crear grupos. El usuario que crea el grupo lo debe configurar en el momento de la creación y automáticamente se le concederán permisos de administración sobre él. Al quedar creado el grupo, todos los usuarios que forman parte del grupo serán notificados acerca de su creación, y podrán comenzar a participar en las conversaciones.

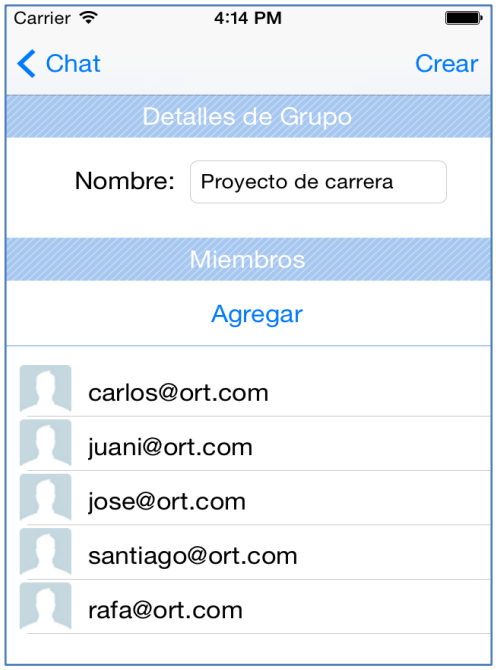


Figura 4.7 – Creación de grupo

Tanto las conversaciones de uno a uno como las conversaciones de grupo se le muestran al usuario en un mismo listado. Este listado esta ordenado según la fecha en la que la conversación tuvo su última actividad y cada ítem del listado le notifica al usuario en caso de que haya nueva actividad, como se pude apreciar en la siguiente figura:

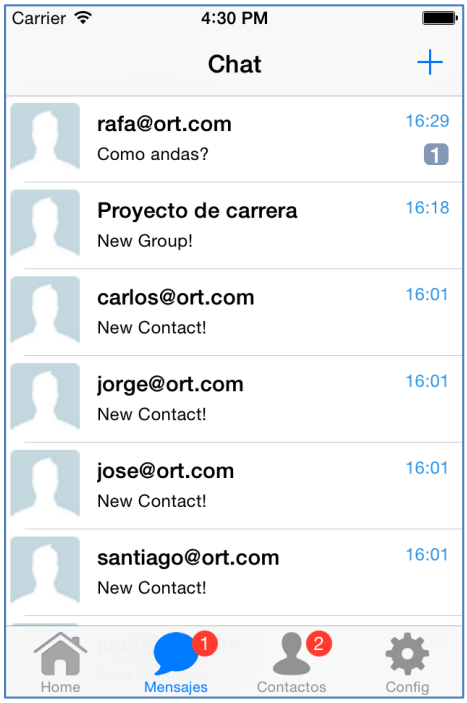


Figura 4.8 – Listado de chats/conversaciones

Cada uno de los ítems de este listado puede ser seleccionado para acceder a su conversación correspondiente. Una vez seleccionado se muestra la pantalla de conversación en la cual el usuario puede ver el detalle completo de la conversación con el contacto/grupo y puede enviar sus propios mensajes. La pantalla de conversación es utilizada en varios módulos del sistema, variando levemente su presentación según el tipo de conversación que se está llevando a cabo. La siguiente figura representa una pantalla de conversación de uno a uno, pero sirve como referencia hacia cada uno de los tipos de pantalla de conversación a los cuales se hace referencia en esta sección:

Figura 4.9 – Pantalla de conversación

El módulo de comunicación se accede únicamente desde las aplicaciones móviles. Es el único módulo al cuál tienen acceso los usuarios básicos.

4.3.3. Soporte

Figura 4.10 – Icono representativo del módulo de soporte

El módulo de soporte conecta a los usuarios de una empresa con sus propios clientes. Únicamente los usuarios de tipo operador, referente y administrador tienen los permisos necesarios para realizar este tipo de comunicación. Ellos atienden las solicitudes de consulta generadas desde los sitios web por parte de los usuarios web, y entablan conversaciones con estos para resolver sus consultas.

Para entender mejor el funcionamiento del módulo de soporte, se presentará un ejemplo:

Se utiliza a nuestro cliente Moon Ideas como una empresa que cuenta con el sistema. Un usuario anónimo (usuario web) ingresa al sitio web de Moon Ideas y decide realizar una consulta mediante la interfaz de consultas web que nuestro sistema provee (esta interfaz de consulta es un *plugin* capaz de comunicarse con los dispositivos móviles de los operadores de la empresa, el cual se instala en los sitios web de las empresas que adquieren el sistema). Al enviar la solicitud de consulta, todos los usuarios de la empresa (con los permisos necesarios) son notificados, y el usuario web espera a que su solicitud sea atendida por uno de ellos.

Figuras 4.11 y 4.12 – Plugin web, solicitud de consulta

Cuando un operador recibe una solicitud de consulta en su dispositivo móvil (en este caso un operador de la empresa Moon Ideas), ésta se presentará en el módulo de soporte bajo la pantalla de listado de solicitudes. El operador puede decidir tomar la solicitud o ignorarla. Si decide ignorarla, entonces será tomada por otro operador de la empresa. Si decide tomarla, se iniciará una conversación con el usuario web. Si la solicitud es tomada por otro operador de la empresa, el resto de los operadores serán notificados.

La pantalla de solicitudes muestra todas las solicitudes generadas, agrupadas según su tipo. Los tipos posibles de solicitud son: nuevas, atendidas por mi y atendidas por otro operador. En el caso de las solicitudes nuevas (aquellas que aún no han sido atendidas por ningún operador), se le indica a los operadores si la solicitud sigue a la espera de ser atendida, o ha sido cancelada por el usuario web (una solicitud se considera cancelada si el usuario web abandona el sitio que contiene el *plugin*, o si pasa un tiempo determinado sin que nadie atienda la solicitud). La siguiente figura nos muestra la pantalla de listado de solicitudes:

Figura 4.13 – Pantalla de listado de solicitudes

Al aceptar una solicitud se genera una conversación de uno a uno con el usuario web (ver figura 8 de esta sección), y se le notifica al resto de los operadores de la empresa que la solicitud ya fue atendida.

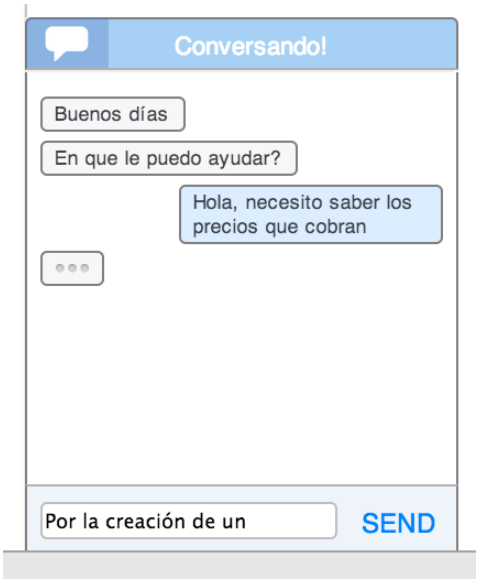


Figura 4.14 – Pantalla de conversación, *plugin* web

De esta manera, los empleados (operadores) de las empresas se comunican con los clientes de la empresa (usuarios web) para brindarles soporte y contestar todas sus consultas.

El módulo de soporte es el único que es un híbrido entre la plataforma *mobile* y la plataforma web, ya que tiene una componente web (*plugin*) por donde los usuarios web se comunican con los dispositivos móviles de los operadores.

4.3.4. Servicios

Figura 4.15 – Icono representativo del módulo de servicios

Por último tenemos al módulo de servicios. Este módulo permite que usuarios de distintas empresas que cuentan con el sistema se comuniquen entre sí.

El módulo de servicios provee un listado de todas las empresas que cuentan con el sistema, junto con cientos de empresas no registradas, las cuales son ingresadas al sistema como empresas invitadas. Todas estas empresas son catalogadas y agrupadas según su rubro comercial (para permitir que el usuario busque la empresa según el tipo de servicio que necesita), brindando así un servicio centralizado de búsqueda que busca emular en cierto sentido a las páginas amarillas.
Cuando una empresa necesita adquirir algún servicio específico, ingresa al listado y busca el rubro correspondiente al servicio que desea adquirir. Una vez que ubique el rubro deseado, podrá ver ahí todas las empresas que ofrezcan servicios relacionados al rubro. Al seleccionar una empresa se pueden presentar dos casos. En el primero, caso en el cual la empresa seleccionada no está registrada en el sistema, se le muestra al usuario información básica de la empresa. En el segundo caso, caso en el cual la empresa está registrada en el sistema, además de obtener la información de contacto, el usuario puede iniciar inmediatamente una conversación con el usuario referente (ver sección 4.1, tipos de usuarios identificados) de la empresa seleccionada.

Para tener un mejor entendimiento de cómo funciona el módulo se presenta un ejemplo:

La empresa Moon Ideas necesita los servicios de un diseñador gráfico para uno de sus proyectos y no conoce una empresa que brinde tales servicios. Por lo tanto uno

de sus empleados utiliza el módulo de servicios para buscar empresas relacionadas a dicho rubro.

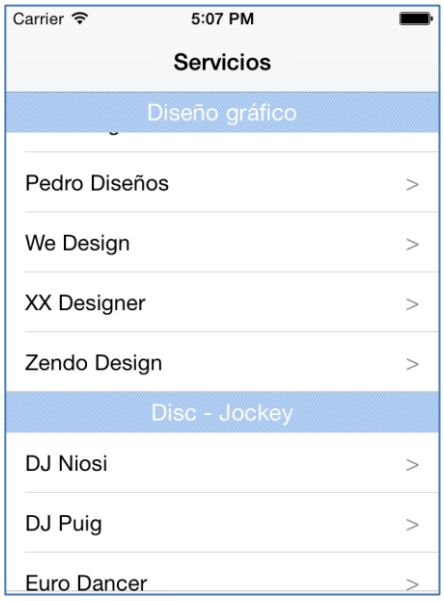


Figura 4.16 – Pantalla de listado de servicios

El empleado de Moon Ideas selecciona la empresa XX Designer, y al estar ésta registrada en el sistema, se le permite al empleado contactar directamente al usuario referente de XX Designer mediante un chat de uno a uno. Una vez que el empleado de Moon Ideas decide iniciar la conversación, el usuario referente de XX Designer es notificado, y se inicia la conversación.

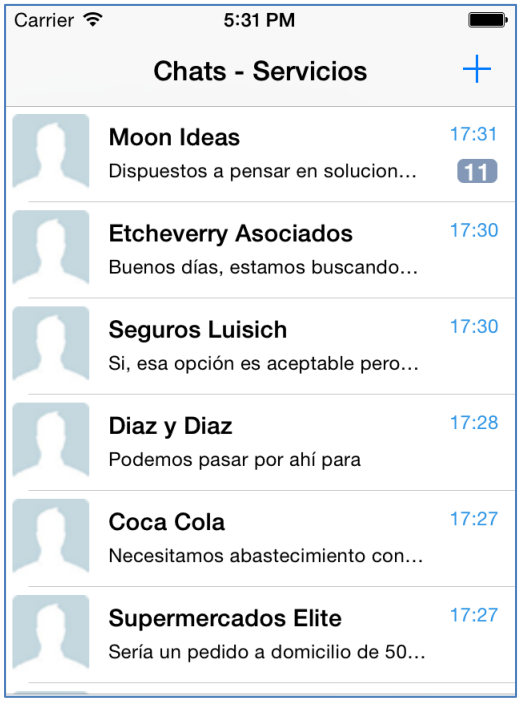


Figura 4.17 – Lista de conversaciones de servicio del usuario referente

De esta manera, el módulo de servicios permite que empleados de distintas empresas se comuniquen entre sí, abriendo la posibilidad de que distintas empresas ofrezcan y soliciten servicios.

El módulo de servicios puede ser accedido por todos los tipos de usuarios, a excepción del usuario básico. Este módulo es accedido únicamente desde los dispositivos móviles, y es el único que involucra interacción entre usuarios de distintas empresas.

4.4. Otras funcionalidades consideradas

Si bien las funcionalidades que se decidieron incluir dentro del alcance de este proyecto fueron las que se explicaron en la sección anterior, se manejaron varias opciones más, las cuales quedaron como posibles opciones para desarrollar en el futuro. No todas están involucradas con el proceso de comunicación de una empresa, sino que se contempló abarcar otras tareas. A continuación se listan algunas de estas funcionalidades:

- **Proyecto:** Se manejó la posibilidad de crear un módulo de control de proyectos, en el cual los usuarios pudiesen agregar proyectos, asignar y manejar los recursos del proyecto, programar un cronograma de actividades asociado al mismo y monitorear su avance, entre otras funciones.

- **Conectar:** Este módulo le permitiría a los usuarios agendar reuniones con otros usuarios, permitiendo crear un espacio compartido entre los miembros de la reunión para compartir recursos y opiniones. Las reuniones serían programadas para que notifiquen de manera automática a los miembros acerca del momento de su realización.

- **Control:** El módulo de control brindaría funcionalidad para que los usuarios lleven a cabo tareas de gestión financiera, es decir puedan llevar un registro organizado de cuentas, ya sea a nivel personal o para una escala mayor, como por ejemplo el registro de movimientos monetarios de una empresa.

- **Servicios, v2.0:** También se manejó la posibilidad de que servicios no funcione como un módulo únicamente de exposición de servicios y comunicación, si no que sirviera para comercializar los productos mediante el *e-commerce*.

Todas estas ideas y otras fueron contempladas junto con el cliente, pero se decidió optar por no implementarlas como parte del alcance de este proyecto, aunque se estableció que se volverían a tener en cuenta en una instancia futura.

5. Marco metodológico

5.1. Introducción

En este capítulo se describe el ciclo de vida y el proceso general de ingeniería de software utilizado en el proyecto.

5.2. Ciclo de Vida

El ciclo de vida elegido fue el incremental iterativo, en el cual se especifican la mayoría de los requerimientos inicialmente y luego se va construyendo el producto en base a iteraciones. A lo largo del proyecto, los requerimientos se pueden ir modificando o se pueden incorporar nuevos.

Este ciclo de vida, se adecua perfectamente a la metodología utilizada durante el proyecto. El hecho de ir produciendo incrementos en la solución, permite obtener mayor *feedback* por parte del cliente, lo cual es favorable para lograr un producto aceptado. A su vez, permite utilizar lo aprendido en la iteración anterior, para no cometer los mismos errores e ir mejorando en las iteraciones subsiguientes.

Antes de presentar el proyecto ante el comité de aceptación de proyectos, se realizó un relevamiento de funcionalidades generales con el cliente. De esa manera, se pudieron establecer los límites del alcance del proyecto. Las funcionalidades relevadas, quedaron registradas en el documento de requerimientos del sistema (ver Anexo 2 : Requerimientos del Sistema – Primera Etapa).

Luego de obtener la aprobación, se realizaron tareas de investigación y pruebas de concepto. Estas tareas, se llevaron a cabo para minimizar el impacto en el desarrollo del producto, ya que ninguno de los integrantes del equipo poseía mayores conocimientos en ciertas tecnologías solicitadas por el cliente.

A partir de ese momento, se comenzó con la planificación del proyecto. Como resultado, se obtuvieron los planes a aplicar en las diferentes áreas que se debían gestionar. Además, con los conocimientos adquiridos en la investigación, se diseñó la primera versión de la arquitectura conjuntamente con el cliente, la cual fue evolucionando a lo largo del proyecto.

Luego, se detalló una porción de las funcionalidades que habían sido relevadas al inicio para comenzar la construcción del producto. Esta instancia, luego se repitió en una de las iteraciones posteriores.

En el diagrama que se muestra a continuación, las actividades de Diseño de Arquitectura no aparecen como parte de las iteraciones. Sin embargo, se decidió mostrarlas en color azul para diferenciarlas del resto de las actividades del proceso. Éstas, se realizaron más de una vez a lo largo del ciclo de vida, pero no se repitieron con tanta frecuencia como las demás tareas de la iteración.

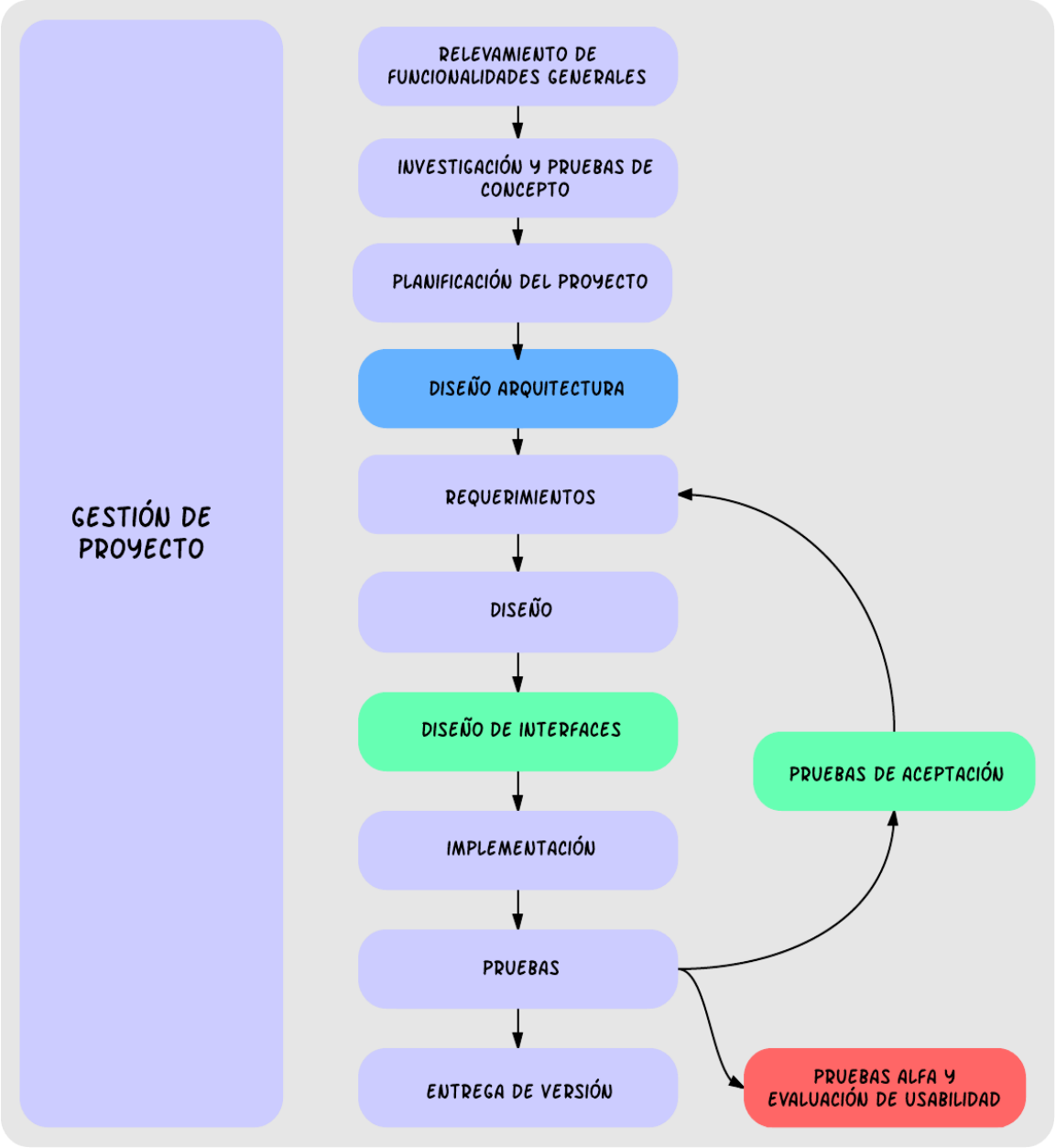


Figura 5.1 - Ciclo de vida del proyecto.

Las iteraciones, tenían una duración de 14 días e incluían las siguientes actividades:

- Requerimientos

- Diseño
- Diseño de interfaces
- Implementación
- Pruebas
- Pruebas de aceptación

Las actividades de Diseño de Interfaces y Pruebas de Aceptación, aparecen en color verde, ya que no formaron parte de todas las iteraciones. Mientras que el Diseño de Interfaces coincidía con el inicio de los diferentes módulos, las Pruebas de Aceptación con la finalización de los mismos.

Por último, antes de la entrega final al cliente, se llevó a cabo la liberación de una versión alfa del producto. Ésta, se le entregó a un grupo de usuarios finales para que realizaran pruebas del sistema en un ambiente controlado. Durante la realización de las mismas, se le solicitó a los usuarios que registraran fallas encontradas, y al final se les realizó una evaluación de satisfacción en términos de usabilidad.

Durante todo el ciclo de vida se realizaron actividades de gestión que se explicarán luego en el capítulo de gestión.

5.3. Selección de la Metodología de Trabajo

Durante la planificación del proyecto, para seleccionar la metodología de trabajo se tuvieron en cuenta tres aspectos centrales: las características del cliente, del equipo y el tipo de interacción que debía haber entre ambos para lograr un producto de calidad.

Al final de la etapa de planificación, se concluyó que se debía optar por una forma de trabajo ágil y flexible, motivo por el cual se definió una metodología propia, basada en las principales características de *scrum* [3].

En el capítulo de Gerencia, se explicará porque razones se decidió definir una metodología y que características le aportó al proyecto.

5.4. Metodología de trabajo

Como se menciona en la sección ciclo de vida, antes de comenzar las iteraciones hubo una segunda etapa de especificación de requerimientos. Éstos, se obtuvieron a partir del documento de requerimientos obtenido en la reunión con el cliente antes de comenzar el proyecto.

Se creó un *product backlog,* que es la pila de requerimientos necesarios para construir el producto, mapeando todos los requerimientos del documento de requerimientos del sistema. Éstos, se especificaron mediante *user stories,* que es la forma típica de especificar requerimientos dentro de las metodologías ágiles. Esta actividad, se repitió en el tercer *sprint* para generar la segunda y última versión del *product backlog.*

Luego, el equipo contaba con suficientes insumos como para comenzar a construir el producto. Más allá de que hubo dos instancias de transformación de funcionalidades a un *product backlog*, en cada iteración las *user stories* seleccionadas se llevaban a un mayor nivel de detalle con la ayuda del *product owner*.

Proceso de Desarrollo

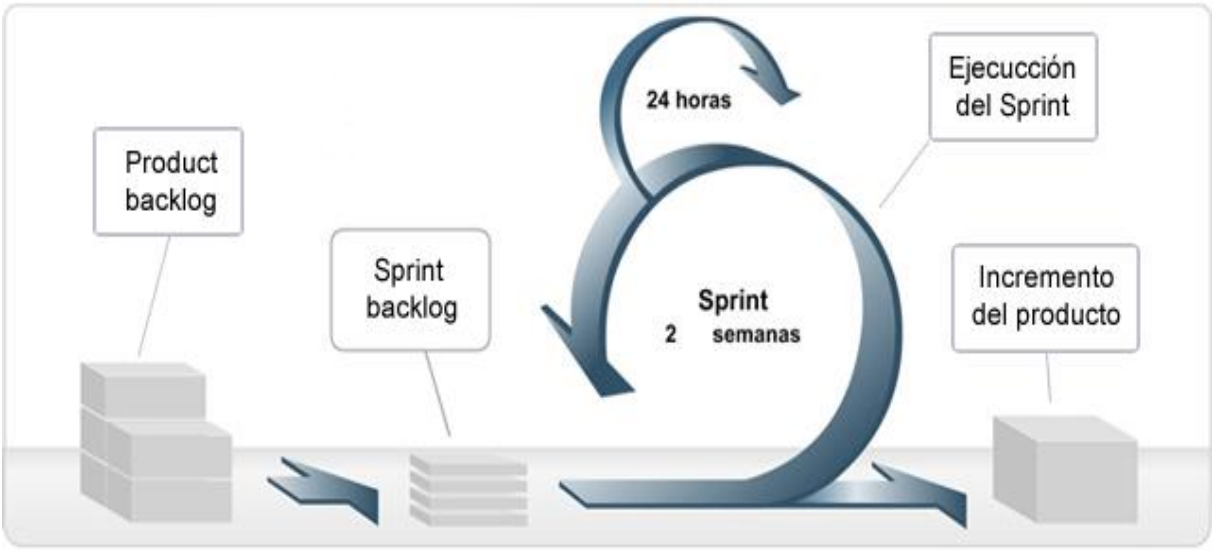


Figura 5.2 - Metodología de trabajo orientada al producto.

El proceso de construcción del producto, se realizó mediante la ejecución de *sprints.* Cada *sprint*, coincidió con las iteraciones mencionadas en la sección 5.2: Ciclo de Vida.

En la metodología seleccionada, se llevaron a cabo seis ceremonias a lo largo de cada *sprint*:

- Planificación del *sprint*
- Ejecución y seguimiento del *sprint*
- Reunión diaria
- Reunión de revisión
- Reunión retrospectiva
- Reunión semanal

Cada una de estas ceremonias, serán explicadas en forma detallada en el capítulo de Gerencia. El aspecto que se debe resaltar en este capítulo es la ejecución y seguimiento del *sprint,* dado que durante ese período se llevan a cabo las actividades del proceso de construcción de software las cuales forman parte de la iteración del Ciclo de Vida*.*

Durante la ejecución de cada *sprint*, se llevaron a cabo actividades de Requerimientos, Diseño, Diseño de Interfaces, Implementación, Pruebas y Pruebas de Aceptación. Todas ellas, se explicarán detalladamente a lo largo de este documento, en los capítulos correspondientes a cada una de las áreas del proceso de Ingeniería de *Software*.

6. Ingeniería de Requerimientos

6.1. Introducción

En el siguiente capítulo se detallará el proceso de Ingeniería de Requerimientos del proyecto. Se presentarán todas la actividades realizadas para llegar a los requerimientos funcionales y no funcionales del sistema.

6.2. Metodología

6.2.1. Relevamiento

Antes de comenzar el proyecto, cuando el equipo debía tomar una decisión acerca de que proyecto final se iba a elegir, todos los integrantes estuvieron seguros de que querían llevar adelante la propuesta “Solución de Comunicación Empresarial”. Para asegurar la aprobación de la misma ante el comité, el equipo decidió realizar un documento de requerimientos (ver Anexo 2: Requerimientos del sistema – Primer Etapa) con una especificación sólida, para que el proyecto fuese tomado con seriedad. Así entonces comenzó la etapa de relevamiento.

El equipo organizó una reunión con el cliente. En esta reunión se discutieron todas las funcionalidades del sistema. Habían dos tipos de funcionalidades, aquellas que seguro iban a ser considerados dentro del alcance del proyecto y otras que la empresa Moon Ideas desearía desarrollar en un futuro. Lo primero que se estableció en esta reunión fueron los módulos del sistema. Luego, para cada módulo se tomó nota de cada una de sus funcionalidades, definiendo su nombre, una breve descripción, módulo al que pertenecían y su prioridad.

El relevamiento continuó a lo largo del proyecto. Es preferible comenzar a describir la etapa de especificación para explicar como continuó el mismo ya que se encuentran fuertemente vinculados.

6.2.2 Especificación

La especificación de requerimientos estuvo constituida por tres grandes etapas y se complementó con instancias de profundización a lo largo del proyecto. La primera etapa fue previa al comienzo del proyecto, y las otras dos, muy similares entre sí, luego del comienzo del mismo.

Luego de la reunión con el cliente se realizó la primer etapa de especificación. Lo que se hizo aquí fue tomar el documento obtenido en la reunión y definir formalmente una por una las funcionalidades, basadas en los datos recolectados correspondientes. Este proceso permitió llegar a un documento de especificación de requerimientos, comprensible para el equipo, el cliente y el comité**.**

Un vez que comenzó el proyecto se optó por trabajar con una adaptación de metodología ágil (*scrum*). Esto forzó a tener que generar un *product backlog* orientado a aquellos requerimientos que estaban contemplados dentro del alcance del proyecto.

En la segunda etapa de especificación se construyó un *product backlog* de veintidós *user stories*, el cual abarcaba dos módulos y parte de un tercero. Se consideró que era preferible especificar solamente estos módulos para ganar experiencia en la estimación y cumplir con una primera meta de desarrollo. Esta etapa no requirió mucho esfuerzo ya que la construcción del *product backlog* se basó en el mapeo de los requerimientos contenidos en el documento de requerimientos del sistema. Para cada requerimiento seleccionado se creó una *user story* con el formato adecuado (ver Anexo 4 : Estructura y formato de *scrum*). Cuando la construcción del *product backlog* se dio por finalizada, el equipo se reunió con el cliente para definir los bosquejos de las *users stories* del *product backlog*. Los mismos fueron hechos en un pizarrón a mano por el cliente.

Junto a la segunda etapa de especificación, comenzaron las instancias de profundización de las *user stories,* aquí surgían nuevas *user stories* o se pulían aquellas que era necesario. La profundización se realizó en las reuniones semanales de los *sprints*. En éstas, se discutían las *user stories* en cuestión, se documentaban y se creaban o rehacían bosquejos para dejar en claro la funcionalidad a realizar. Como estas reuniones eran en plena ejecución de un *sprint,* el equipo desarrollaba las *user stories* con las ideas bien definidas. Estas instancias continuaron hasta el último sprint. Para tener un mejor entendimiento de estas estas reuniones, vea su explicación detallada en el capítulo de Gerencia.

La tercera etapa fue análoga a la segunda. La única diferencia fue que se definieron 16 *user stories* en vez de 20. Con este incremento se completó la parte restante del tercer y cuarto módulo.

6.2.3. Priorización de *user stories*

En la reunión de relevamiento se definió la prioridad de todos los requerimientos del sistema. En la instancia de mapeo se asignó la prioridad que tenía el requerimiento al *user story.* La escala de prioridad estaba dada por los siguientes niveles:

- Nivel 1: Requerimientos indispensables para el sistema.
- Nivel 2: Requerimientos que no son indispensables para el sistema pero aumentarían el atractivo para los usuarios.
- Nivel 3: Desde la perspectiva del usuario, los requerimientos con menor atractivo.

A la hora de asignar *user stories* a un *sprint* se consideró la priorización hecha por el cliente en la reunión de relevamiento. Existieron casos en los cuales, por un tema de cantidad de *story points* por *sprint,* no era posible introducir una *user story* de mayor prioridad frente a otra de menos. Aquí se decidía junto al cliente, que *user story* era apropiado o conveniente desarrollar en ese *sprint*, según el tamaño y prioridad de la misma.

6.2.4. Validación

Una vez que el equipo finalizó el documento de requerimientos, el mismo se lo entregó al cliente para que éste lo validara. Afortunadamente el cliente aprobó el documento y éste fue entregado al comité, el cual también dio su aprobación dando un comienzo oficial al proyecto.

Como se construyó un *product backlog* especificado con otro nivel de detalle respecto al documento de requerimientos, se le entregó dicho documento al cliente. Esto se llevó a cabo luego de la segunda y tercer etapa de relevamiento con el fin de obtener su aprobación. El cliente valido este documento, lo cual era bastante predecible ya que había aprobado el documento de requerimientos.

6.2.5. Verificación

Las verificaciones fueron realizadas mediante pruebas de aceptación y pruebas de usabilidad. Las pruebas de aceptación, se llevaron a cabo luego de la construcción de cada módulo con el cliente (ver Anexo 8: Pruebas de Aceptación). Las pruebas de usabilidad fueron realizadas por usuarios finales. Dichas prácticas se detallan en el capítulo de Pruebas.

6.3. Justificación de la estrategia de relevamiento

Partiendo de la especificación inicial de requerimientos (ver sección 6.2.1) y tomando en cuenta la metodología elegida, se procedió a confeccionar un *product backlog* que incluya los requerimientos mencionados anteriormente junto con las nuevas propuestas del cliente. Matías Burgos fue la persona dentro de Moon Ideas que se encargó de transformar estas propuestas en requerimientos (debido a su conocimiento del negocio), adoptando el rol de *product owner.* El mismo se aseguró de apegarse a los siguientes principios:

- "Ser responsable de definir la visión del producto desde el punto de vista de negocio definiendo las características principales que harán ver la luz al producto [4]."
- "Ser el responsable de compartir la visión con todo el equipo tanto hacia el negocio como hacia el equipo de desarrollo [4]."
- "Ser el responsable de la definición del producto mínimo viable y el plan de releases del producto [4]."
- "Colaborar para definir el *product backlog* [4]."
- Evaluar los criterios de aceptación luego de desarrollo de cada módulo.
- "Aceptar o rechazarlos resultados del *sprint* [4]."
- "Estar disponible para atender las consultas y preguntas del equipo acerca de las reglas de negocio para implementar cada historia [4]."
- "Participar en la reunión de *sprint planning*, para la definición estratégica del *sprint* y estar disponible a lo largo de toda la reunión [4]."
- "Comunicar a los *stakeholders* el plan de lanzamiento del producto y coordinar con otros actores los planes de ventas, marketing de producto, formación, atención a clientes, etc. [4]."
- "Colaborar con el *scrum master* para el buen fin del desarrollo del producto, ayudándole y apoyándole en sus funciones [4]."
- "Procurar los recursos necesarios al equipo para que éste pueda realizar su trabajo de manera adecuada [4]."
- "Ser responsable de la rentabilidad del producto [5]."
- "Priorizar las características según el valor de Mercado [5]."

- “Ajustar las características y prioridades por iteración, según sea necesario [5].”

6.4. Requerimientos del sistema

A continuación se presentan los requerimientos funcionales y no funcionales del sistema.

6.4.1. Requerimientos Funcionales

El capítulo 4: Descripción de la solución, introduce las funcionalidades del sistema. Las mismas pasaron por el proceso explicado en las secciones anteriores, llegando en última instancia al *product backlog*. El detalle del mismo se encuentra en el Anexo 7: Product backlog. Aquí se ven todas las *user stories* escritas para el desarrollo de las funcionalidades del sistema.

A modo de resumen se listan los nombres de las funcionalidades de cada módulo desarrollado.

6.4.1.1 Módulo de Administración

1. Registro de una empresa.
2. Editar empresa.
3. Listar empresas.
4. Ver estadísticas del sistema.
5. Ver estadísticas de empresa.
6. Ver estadísticas de usuario.
7. ABM de administradores.
8. Iniciar sesión usuario básico.
9. Iniciar sesión súper administrador.
10. Cerrar sesión.
11. Registro de un usuario básico a una empresa.

12. Asignar rol a usuario básico.
13. Cambio de contraseña.

6.4.1.2 Módulo de Comunicación

1. Inicio de sesión.
2. Cierre de sesión.
3. Lista de secciones de la aplicación.
4. Enviar solicitud de contacto.
5. Aceptar solicitud de contacto.
6. Listar registrados.
7. Enviar mensaje.
8. Recibir mensaje.
9. Listar contactos.
10. Crear Grupo.
11. Agregar usuario básico al grupo.
12. Eliminar usuario básico del grupo.
13. Abandonar Grupo.
14. Eliminar grupo.

6.4.1.3 Módulo de Soporte

1. Comunicarse de la web al dispositivo.
2. Enviar mensaje del dispositivo a la web.
3. Listar conversación con usuarios web.
4. Listar solicitudes de web usuarios web.
5. Aceptar solicitud web usuarios web.

6. Eliminar solicitudes aceptadas.

6.4.1.4 Módulo de Servicios

1. Listado de empresas por rubro.
2. Listado de empresas por nombre.
3. Búsqueda (rubro o nombre).
4. Ver datos empresa.
5. Listar contactos de empresa

6.4.2. Requerimientos No Funcionales

El cliente también pidió que el sistema cumpla con ciertas características. Las mismas se describen a continuación como requerimientos no funcionales. La forma en que se cubrió cada uno de ellos se encuentra en el capítulo de arquitectura.

- **Seguridad**

Autenticación: El sistema deberá identificar al usuario que intente entrar al sistema a través de su email y contraseña por cualquiera de las vías posibles(web o dispositivo móvil).

Detección de intrusos: El sistema bloqueará a aquellos usuarios que luego de un determinado número de intentos fallidos no logren ser autenticados. Acto seguido el sistema guarda esta información para llevar registro de este tipo de actividad.

Privacidad: El sistema mantendrá privada toda la información relacionada a las empresas y sus usuarios.

Auditabilidad: El sistema deberá registrar todas la acciones realizados por los usuarios del sistema.

- **Usabilidad**

Interfaz: La aplicación móvil contará con una interfaz intuitiva y del estilo de las

aplicaciones ya existentes en el mercado para que el usuario se sienta a gusto con la misma. El sitio web también será intuitivo.

- **Mantenibilidad**

El sistema debe ser comprensible y modularizado para que futuros programadores puedan hacer cambios de forma eficaz y eficiente.

- **Performance**

Es necesario que las siguientes acciones sean realizadas en un determinado lapso de tiempo.

  - En caso de que un usuario básico tenga un historial (mensajes, reuniones, grupos, etc.) e inicie sesión en cualquier dispositivo, el mismo recuperará su historial entre 3 y 8 segundos.
  - El dispositivo móvil contará con un tiempo de respuesta con el servidor entre 2 y 5 segundos para cualquier funcionalidad.
  - El envío de mensajes no deberá ser tener una duración mayor a 5 segundos.

- **Disponibilidad**

La disponibilidad del sistema debe ser 24X7. Estará sujeta a la disponibilidad de los servidores ya contratados.

- **Escalabilidad**

Las capacidades del sistema deberán poder adaptarse rápidamente ante cambios que surjan.

7. Arquitectura

7.1. Introducción

Este capítulo tiene como objetivo presentar y detallar las decisiones arquitectónicas tomadas durante el desarrollo de la solución. Se muestran aquí las decisiones que se tomaron para satisfacer los atributos de calidad que se consideraron necesarios y como estas fueron afectando a la arquitectura del sistema.

Se describen los patrones utilizados, las distintas componentes que conforman el sistema y las tecnologías seleccionadas para la construcción de dichas componentes.

7.2. Atributos de calidad considerados

Debido a que el diseño arquitectónico es determinante para cumplir con los atributos de calidad que consideramos necesarios, se decidió realizar un diseño para cumplir con un grupo específico de ellos, los cuales consideramos eran fundamentales. Los atributos a cumplir se definieron en base al tipo de aplicación que se buscaba desarrollar, tomando en cuenta aplicaciones similares ya existentes y las exigencias que los usuarios tienen con las mismas.

Se identificaron los siguientes atributos :

- Escalabilidad
- Seguridad
- Mantenibilidad
- Disponibilidad
- Usabilidad
- Performance

7.2.1. Atributos de calidad etapa 1

Hay que destacar que la arquitectura del sistema tuvo dos etapas marcadas, las cuales se explican en distintas secciones durante este capítulo. En la primera etapa se hizo foco en los siguientes atributos:

Escalabilidad:

Debido a la naturaleza de la aplicación es probable que el número de usuarios pueda crecer rápidamente, y que la cantidad de datos enviados a los servidores crezca de manera conjunta. Por ello, nos pareció imprescindible que el sistema facilite la escalabilidad, tanto de manera vertical como horizontal.

Para lograr dicha escalabilidad se hizo hincapié en conseguir un buen servicio de *hosting* y seguir el patrón Stateless.

En cuanto al *hosting*, se analizaron varias opciones, y finalmente se decidió seguir los consejos de nuestro cliente (el cual tiene amplia experiencia en el tema) y contratar los servicios de la empresa Total Server Solutions (ver Anexo 10: Hosting). Dicha empresa nos provee el poder de procesamiento y el almacenamiento que sea necesario, permitiendo que se agreguen servidores a gusto para poder soportar grandes números de conexiones simultáneas y grandes volúmenes de datos. Además cuenta con un servicio de soporte que se encarga de realizar las tareas de escalabilidad y dar asesoramiento acerca de cómo hacerlo de la manera más efectiva.

Para poder aprovechar los servicios ofrecidos por nuestro proveedor de *hosting* fue necesario utilizar el patrón *Stateless*. Dicho patrón dictamina que no se debe guardar información de estado en los servidores. De esta manera, los usuarios pueden realizar múltiples llamadas al sistema y la carga será distribuida entre los distintos servidores de la manera más eficiente, sin ningún problema.

Seguridad:

En cuanto a la seguridad, se consideró necesario que el sistema no sea abierto al público general, sino que solo pueda ser utilizado por usuarios autenticados. Además, se consideró que los datos de los usuarios son datos sensibles y por lo tanto no deben poder ser accedidos por otros usuarios ni por agentes ajenos al sistema. Por último, al tratarse de un sistema de mensajería, en el cual los mensajes viajan por la red y por lo tanto quedan expuestos a ser interceptados, se decidió utilizar algún mecanismo para que no se pueda leer el contenido de estos mensajes.

Para la autenticación se creó un sistema de *login* de usuarios, tanto en el módulo web de administración como en los dispositivos móviles. En este último caso, el *login* se realiza una única vez, cuando se utiliza la aplicación por primera vez. Además, se crearon distintos roles de usuario, para diferenciar a aquellos que tienen permisos de administración de aquellos que no los tienen.

Para mantener los datos sensibles seguros, estos se guardan únicamente en el servidor, y nunca viajan por la red. Los dispositivos móviles no hacen uso de estos datos ya que son inseguros y susceptibles a ataques.

Para la encriptación de los mensajes se utilizó el protocolo SSL (Secure Sockets Layer). Este provee criptografía asimétrica, la cual permite encriptar los mensajes mediante una clave pública para que solo puedan ser desencriptados por su llave privada correspondiente. Éste mecanismo no solo asegura que los mensajes sean ilegibles por intrusos, sino que permite verificar que el mensaje no haya sido alterado y permite verificar la identidad de quien lo envía.

Mantenibilidad:

Debido a la naturaleza ágil del proyecto y sus requerimientos cambiantes, se tuvo en cuenta la mantenibilidad a la hora de diseñar la arquitectura.

Para lograr la característica de mantenibilidad, se diseñó una arquitectura de capas, la cual permite separar las distintas responsabilidades del sistema en módulos. Esta separación permite que cada módulo sea modificado sin afectar al resto y además brinda reusabilidad.

A su vez, se utilizó el patrón MVC, el cual permite que la interfaz gráfica sea independiente de la lógica de negocio y de los datos, facilitando la modificación de ésta para los distintos dispositivos con los cuales interactúa la aplicación.

Disponibilidad:

En cuanto a la disponibilidad, al tratarse de un sistema de mensajería instantánea, el cual es utilizado en todo momento, se consideró fundamental contar con un servicio que asegure la máxima disponibilidad posible. Nuestro proveedor de *hosting*, Total Server Solutions es quien nos brinda esta característica, asegurando una disponibilidad del 99,9%. Además, como nuestros servidores no tienen estado, las solicitudes que se envían a nuestro sistema pueden ser atendidas de igual forma por cualquiera de nuestros servidores. Esto permite que en caso de falla, las solicitudes se pueda reenviar a los servidores de respaldo sin problemas.

7.2.2. Atributos de calidad etapa 2

En la segunda etapa de la arquitectura, con la inclusión del protocolo de mensajería instantánea XMPP (ver sección 6.3: Descripción y fundamentación de arquitectura) y con una mayor dedicación de los tiempos de construcción al desarrollo de la interfaz de usuario, se favorecieron la performance y la usabilidad:

Performance:

Dado que la mensajería instantánea es la funcionalidad central del servicio, realmente era necesario que los mensajes sean enviados y recibidos de manera instantánea o en el menor tiempo posible que las tecnologías existentes permitieran.

En la primer etapa de la arquitectura, la comunicación entre los dispositivos móviles y el servidor, tanto para el intercambio de datos de lógica de negocio como para el envió de los mensajes, se realizaba mediante el protocolo HTTP a través de una API REST. Esto generaba una cargaba excesiva en el servidor, ya que los servidores HTTP no cuentan con mecanismos para alertar a los dispositivos cliente acerca de la llegada de nueva información, y por lo tanto se debía realizar un *polling* periódico. Además, aunque tras realizar pruebas, los tiempos de respuesta eran aceptables, estaba claro que no eran óptimos y que se debía buscar una mecanismo alternativo.

Se comenzaron a investigar soluciones similares existentes en el mercado, y se llegó a la conclusión de que para conseguir un resultado óptimo, se debería contar con un servidor que utilice el protocolo XMPP. Este protocolo, diseñado específicamente para la mensajería instantánea [6] permitió obtener unos tiempos de envío y recepción de mensajes mucho menores, descomprimir la carga del servidor HTTP (el servidor XMPP es un servidor dedicado únicamente al envío de mensajes) y brindó un mecanismo para que los dispositivos reciban la nueva información sin necesidad de realizar *polling*.

Usabilidad:

Durante todo el desarrollo se consideró que la usabilidad era clave para lograr un producto satisfactorio. Para lograr una usabilidad adecuada, se procuró desarrollar una interfaz intuitiva y sencilla, basada en las heurísticas de Nielsen [13]. Los principales criterios que se tuvieron en cuenta fueron:

- Informar al usuario cuando hay nueva actividad, de tal forma que el éste comprenda la información de manera intuitiva y sin alterar al resto de la interfaz de usuario.

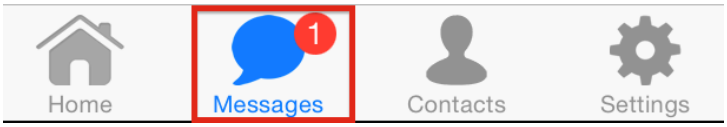


Figura 7.1 – Información de actividad, "Nuevo mensaje recibido".

- Mostrar información de estado cuando la aplicación está realizando actividades de procesamiento, sin bloquear ni interferir con la interfaz de usuario.

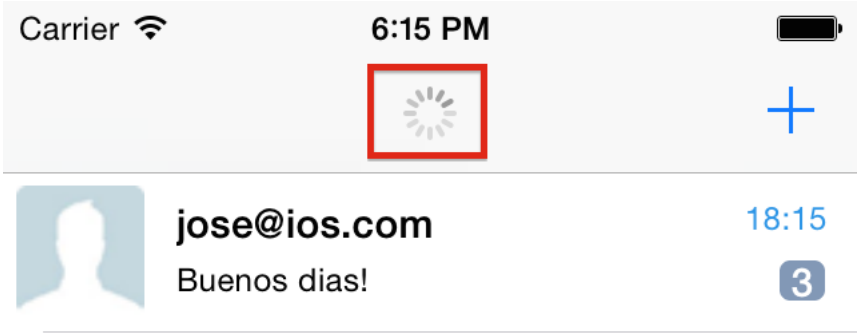


Figura 7.2 - Informe de estado, "*Loading*".

- Guiar a los usuarios nuevos, que usan la aplicación por primera vez, para que aprendan a usarla por si mismos rápidamente.

Figura 7.3 - Guía de nuevos usuarios, "Agregar contactos".

- Mostrar mensajes de error cuando sea necesario, pero que estos aparezcan en lenguaje natural, que pueda ser fácilmente entendido por el usuario.

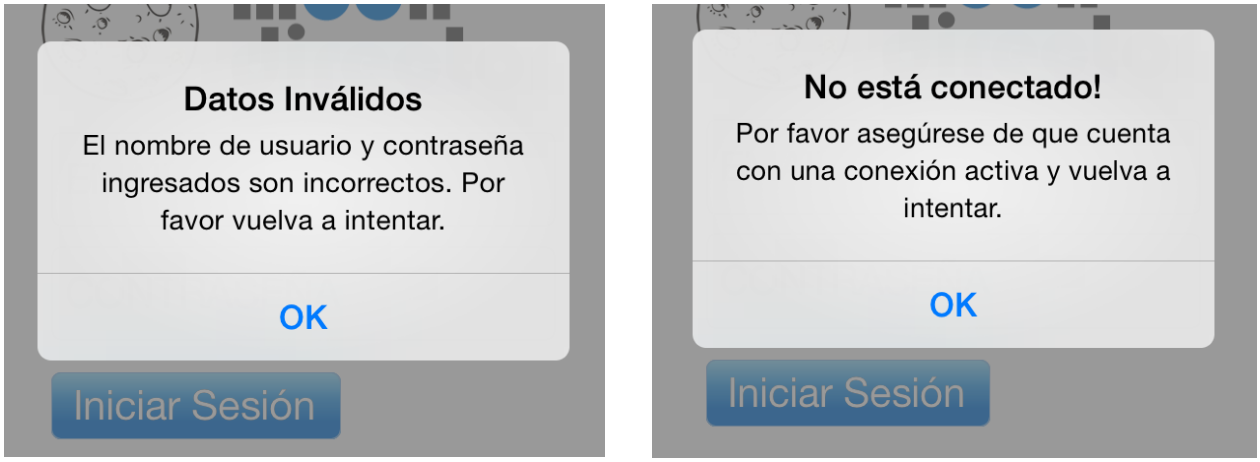


Figura 7.4 – Mensajes de error

7.3. Descripción y fundamentación de la arquitectura

Como se mencionó anteriormente (ver sección 6.2: atributos de calidad) la arquitectura tuvo dos etapas marcadas. En la primera etapa se utilizó un servidor HTTP junto con una API REST como mecanismo de envío de mensajes. En la segunda, se realizó una investigación profunda acerca de las tecnologías utilizadas para la mensajería instantánea, con el fin de solucionar algunos problemas originados a partir del uso del primer mecanismo, y se decidió utilizar un servidor dedicado para la mensajería que utilice el protocolo XMPP. Esto permitió que el servidor HTTP quede libre para enfocarse en otras tareas (para ver el análisis completo de la inclusión del servidor XMPP vea el Anexo 12: Detalle de protocolos utilizados).

La descripción y los fundamentos de arquitectura que se presentan a continuación hacen referencia a la segunda etapa, la cual fue la solución final del sistema.

7.3.1. Visión global

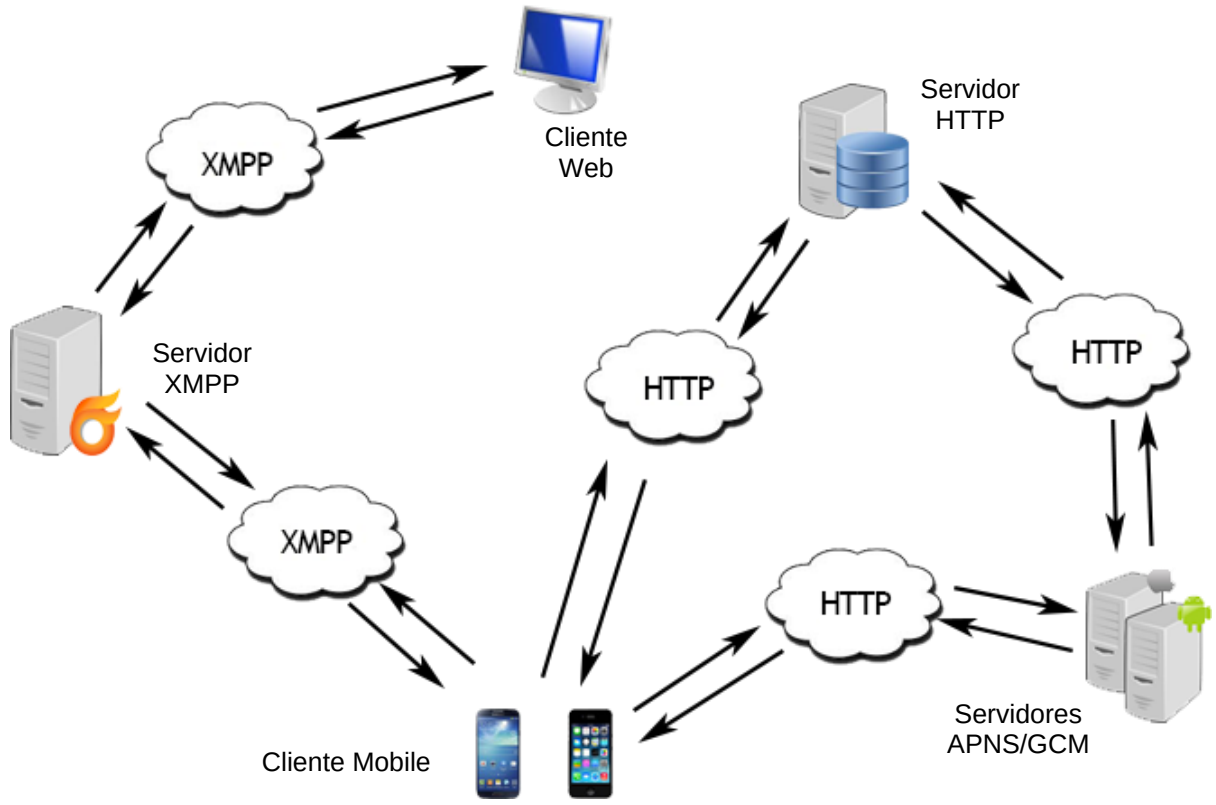


Figura 7.5 – Visión global de arquitectura

El sistema es utilizado principalmente en el entorno *mobile*. Los usuarios se comunican los unos con los otros utilizando la aplicación para dispositivos móviles. Para ello utilizan el servidor XMPP para el envío de mensajes instantáneos, y el servidor HTTP para realizar otras tareas de lógica de negocio, como pueden ser el envío de notificaciones o las tareas de manipulación de contactos. A la misma vez, los usuarios pueden utilizar clientes web para comunicarse con aquellos usuarios que cuentan con la aplicación móvil.

Un caso de uso base que muestra cómo funciona esta arquitectura es el envío de un mensaje de texto. Al enviar el mensaje suceden dos acciones:

Por un lado, el contenido del mensaje es enviado al servidor XMPP para que éste lo redirija a el destinatario. Si el destinatario está activo, es decir tiene la aplicación móvil abierta, el servidor XMPP le envía el mensaje de manera instantánea. Si no está activo, el servidor XMPP guarda el mensaje y queda a la espera de que el destinatario se active, para enviarlo.

De manera simultánea a la acción anterior, el dispositivo móvil se comunica con el servidor HTTP para indicarle que se ha enviado un mensaje. Éste a su vez se encarga de notificar al destinatario (en caso de que éste no esté activo) de que se le ha enviado un nuevo mensaje, a través de los servicios de APNS y GCM. Una vez que el destinatario es notificado y éste accede a la aplicación, el servidor XMPP le envía el mensaje que había almacenado.

7.3.2. Estilos y patrones utilizados

7.3.2.1. Capas Lógicas (*Layers*)

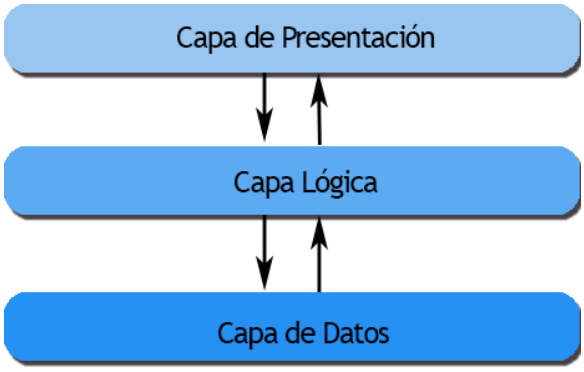


Figura 7.6 – Arquitectura de tres capas

Desde el punto de vista lógico, la arquitectura se diseñó en tres capas. Ésta división en capas permite separar las responsabilidades, encapsulándolas en unidades independientes. Esto favorece la mantenibilidad y la reusabilidad, ya que cada capa puede ser utilizada de manera individual y su modificación no causa un impacto directo en las otras capas.

Se implementaron las siguientes tres capas:

- Capa de presentación: Esta capa presenta la interfaz gráfica mediante la cual el usuario interactúa con el sistema.
- Capa Lógica o de negocio: Maneja la lógica del negocio. Actúa como enlace entre las otras dos capaz, es decir obtiene los datos de la capa de datos, los procesa y se los envía a la capa de presentación.
- Capa de datos: Encapsula la funcionalidad de comunicación con la base de datos y provee métodos para que otras capaz accedan a los datos.

7.3.2.2. MVC

Para conseguir la arquitectura de tres capas mencionada anteriormente, se utilizó el patrón MVC. Cuando se realiza un pedido al servidor, éste es atendido por un controlador, el cual le pide los datos a un modelo, los procesa y se los envía a la vista correspondiente. Las vistas corresponden tanto a los clientes web como a los clientes móviles.

La combinación de MVC y *Layers*, logra abstraer las distintas responsabilidades en módulos que pueden ser remplazados sin afectar a los otros. Por ejemplo nuevas vistas pueden ser creadas y remplazar a las viejas vistas sin afectar a los datos ni a los controladores, y de la misma forma el mecanismo de almacenamiento de datos se podría cambiar sin afectar a las vistas ni a los controladores.

7.3.2.3. Cliente – Servidor

El sistema presenta una arquitectura de tipo cliente – servidor, en la cual el cliente es aquel que contiene la lógica de presentación, y el servidor es aquel que atiende las peticiones de los clientes y mantiene la lógica de negocio y los datos.

Hay que destacar que la solución cuenta con dos tipos de cliente y dos tipos de servidor. El primer tipo de servidor es el servidor HTTP. Este provee una API REST, con la cual los clientes se pueden comunicar, realizar peticiones y recibir respuestas en formato JSON. El otro tipo de servidor es el XMPP. XMPP es un protocolo especializado para el envío de mensajes, el cual utiliza un formato XML especial llamado stanza. Los clientes intercambian stanzas entre ellos mediante este servidor. En cuanto a los clientes, por un lado están los clientes *mobile*, utilizados por los usuarios registrados del sistema para comunicarse mediante dispositivos móviles, y por el otro están los clientes web, utilizados por usuarios anónimos para realizar consultas mediante un browser.

7.4. Vistas Arquitectónicas

En esta sección se muestran algunas de las vistas arquitectónicas más relevantes, basándose en el modelo de 4+1 de Philippe Krutchen [7]. Estas vistas muestran como es la estructura de la arquitectura del sistema, desde los puntos de vista de los distintos interesados.

7.4.1. Vista de despliegue

La siguiente figura muestra las componentes que forman parte del sistema y como estas se conectan e interactúan entre ellas.

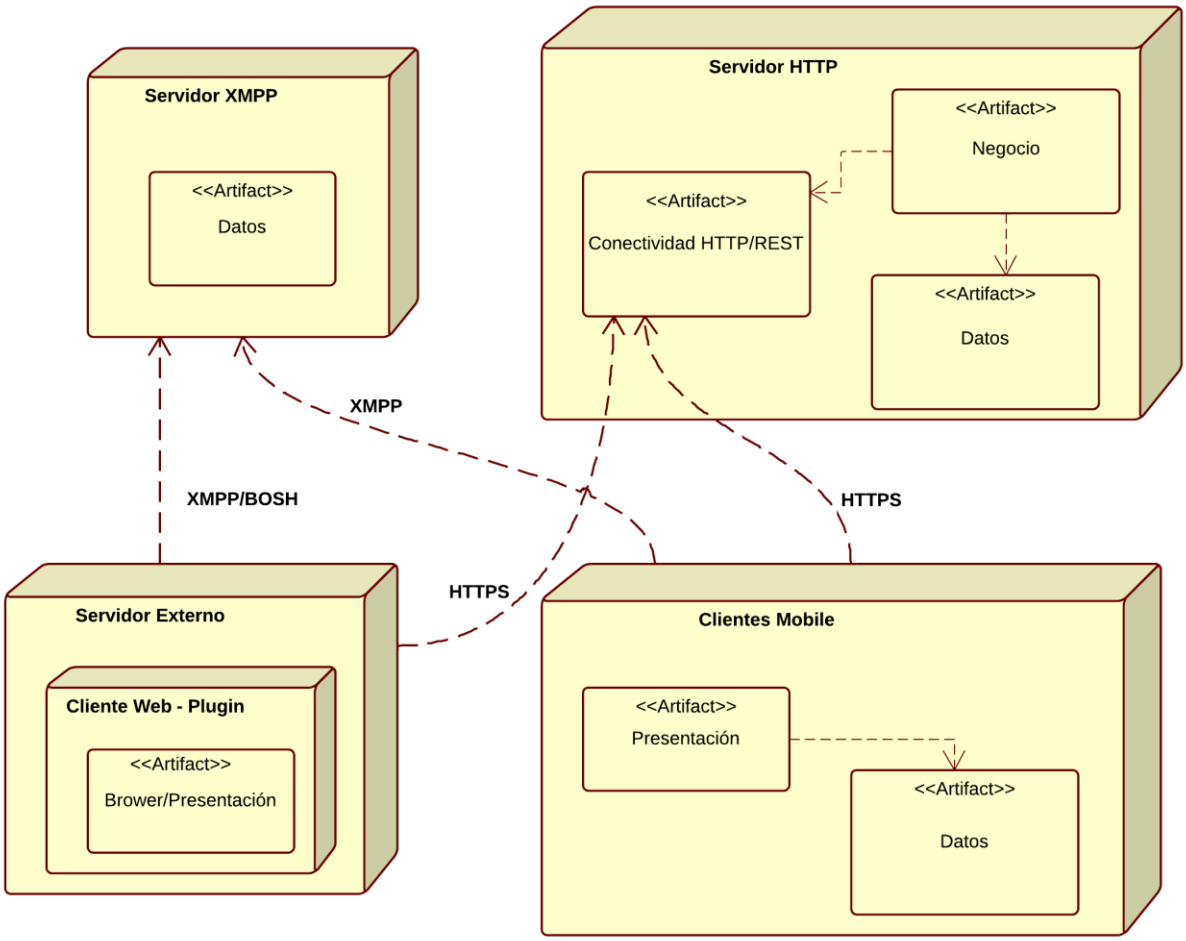


Figura 7.7 – Diagrama de despliegue

El diagrama nos muestra las distintas componentes del sistema:

Servidor HTTP: Es el encargado de llevar a cabo la lógica de negocio, es decir, toda la funcionalidad se lleva a cabo aquí, a excepción del envío de mensajes instantáneos. También contiene la base de datos del sistema y los medios para acceder a esta. Provee una API REST para que los clientes puedan realizar solicitudes mediante HTTP/JSON.

Servidor XMPP: Es el encargado de llevar a cabo todas las tareas de mensajería entre los clientes, tanto *mobile* como web.

Clientes Mobile: Aplicación móvil que funciona como principal punto de interacción de los usuarios con el sistema. Posee su propia base de datos, la cual es actualizada constantemente por los servidores para mantener al día los datos que se le presentan al usuario. Realiza solicitudes al servidor HTTP a través de la API REST que éste provee y se comunica con el servidor XMPP mediante el protocolo XMPP.

Clientes Web: *Plugin* que reside en servidores externos (pertenecientes a las empresas que contratan el servicio), para que usuarios de tipo *web* puedan comunicarse con los clientes *mobile.* Realiza solicitudes al servidor HTTP a través de la API REST que éste provee y se comunica con el servidor XMPP mediante XMPP/BOSH (ver Anexo 12: Detalle de protocolos utilizados).

7.4.2. Vista Lógica

A continuación se presentan los paquetes que constituyen a las distintas componentes del sistema:

7.4.2.1. Vista Lógica Servidor XMPP

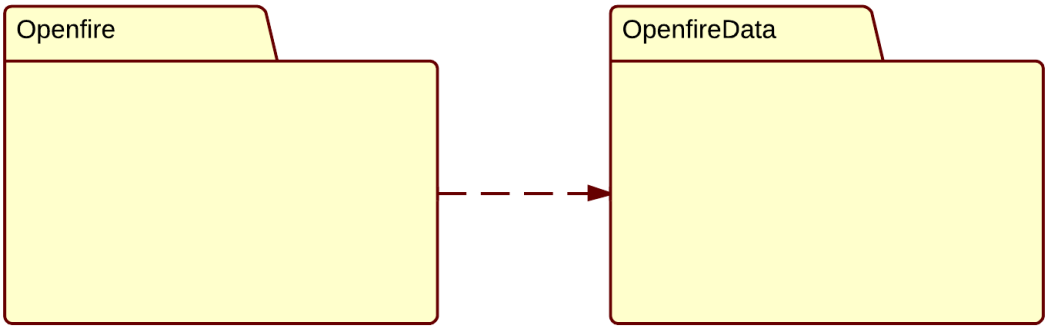


Figura 7.8 – Vista lógica servidor de mensajería

El servidor XMPP fue creado utilizando la tecnología Openfire (ver sección 6.5). Se creó una base de datos dedicada para este servidor, y Openfire se encarga de la lógica necesaria para la manipulación de mensajes, grupos de mensajería y demás actividades relacionadas.

7.4.2.2. Vista Lógica Servidor HTTP

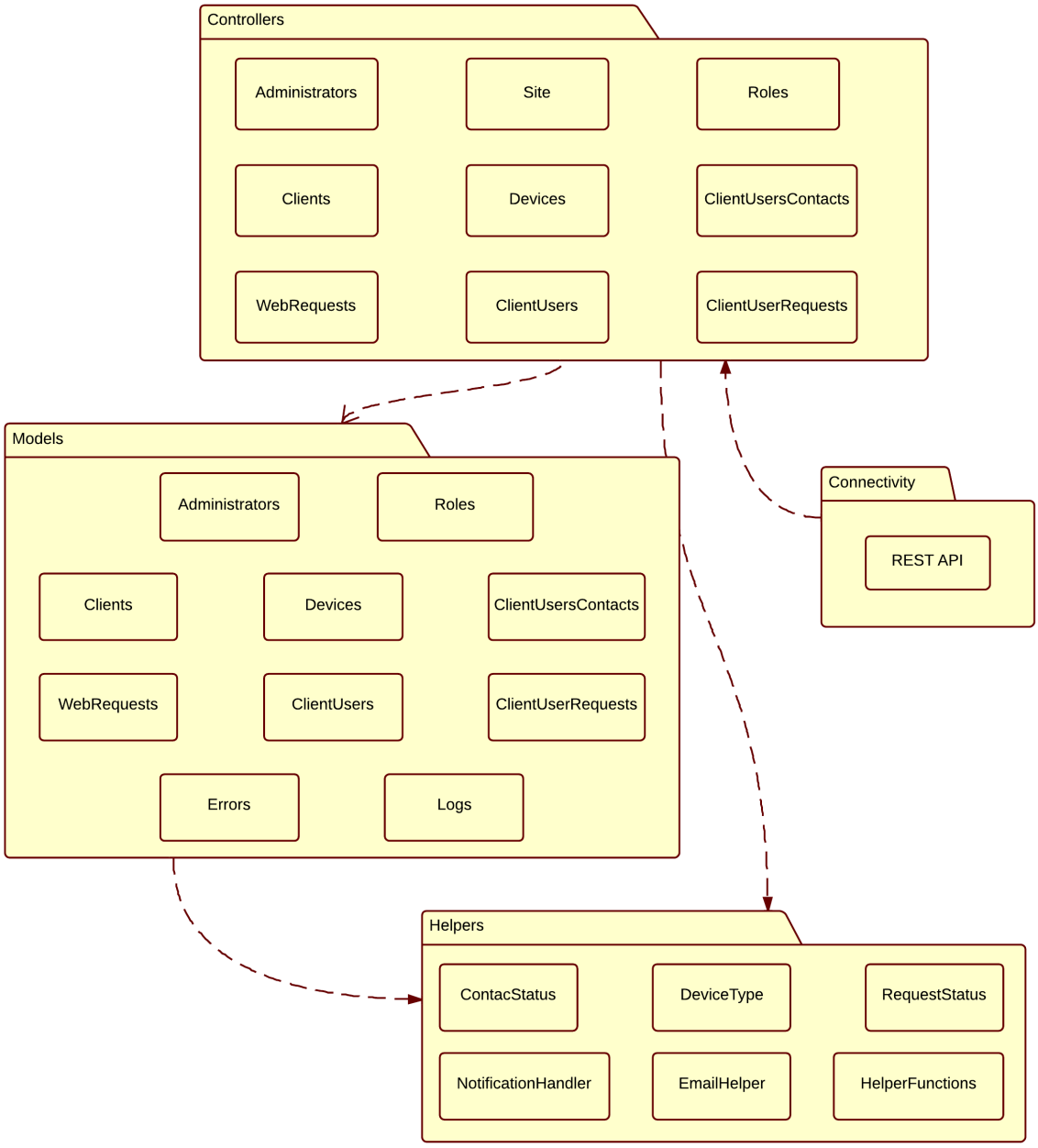


Figura 7.9 – Vista lógica servidor web

El servidor web presenta cuatro grandes paquetes; el paquete de controladores, el de modelos, el de *helpers* o ayudantes y el de conectividad. Cada paquete agrupa elementos con funcionalidades relacionadas.

Connectivity: Contiene las clases encargadas de exponer los servicios del servidor a las aplicaciones cliente. Está constituida principalmente por una API REST que obtiene los datos relevantes de los controladores y los presenta en formato JSON.

Controllers: Contiene los controladores, es decir todas aquellas clases asociadas a la lógica de negocio:

- **ClientUsers:** Representan a los usuarios del sistema.
- **Administrators:** Tipo de usuario especial con permisos para utilizar el módulo de administración.
- **Clients:** Representan a las empresas que están registradas en el sistema.
- **Roles:** Tipos de rol, junto con sus privilegios asociados, que un usuario puede tener.
- **Devices:** Representan a los dispositivos móviles que utilizan la aplicación (clientes *mobile*).
- **ClientUserContacts:** Representan los contactos de un usuario.
- **ClientUserRequests:** Representan las solicitudes de contacto de un usuario.
- **WebRequests:** Son las solicitudes que generan los usuarios en los clientes *web*. Se encargan de poner a éstos en contacto con el cliente *mobile* correspondiente.
- **Site:** Este controlador está encargado del sitio de administración, en el cual los administradores pueden configurar a su empresa y sus usuarios. Además se encarga del sitio de presentación, un sitio estático en el cual se presenta información acerca del producto.

Models: Contiene los modelos. Son aquellos encargados de acceder y manipular los datos de cada entidad. Cada controlador tiene un modelo asociado.

Helpers: Encapsula las clases que brindan funcionalidad adicional al sistema central, las cuales son utilizadas de forma repetitiva en el sistema.

- **ContactStatus**: Representa los distintos estados que pueden tener los contactos de un cliente.
- **DeviceType**: Representa los distintos tipos de plataformas móviles que interactúan con el sistema.

- **RequestStatus**: Representa los distintos estados que puede tener una solicitud web.
- **NotificationHandler**: Encapsula la funcionalidad necesaria para el envío de notificaciones (*Push Notification*) a las distintas plataformas móviles.
- **EmailHelper**: Encapsula la funcionalidad necesaria para el envío de emails.
- **HelperFunctions**: Brinda una serie de funciones útiles que se utilizan múltiples veces en la solución.

7.4.2.3. Vista Lógica Cliente Mobile

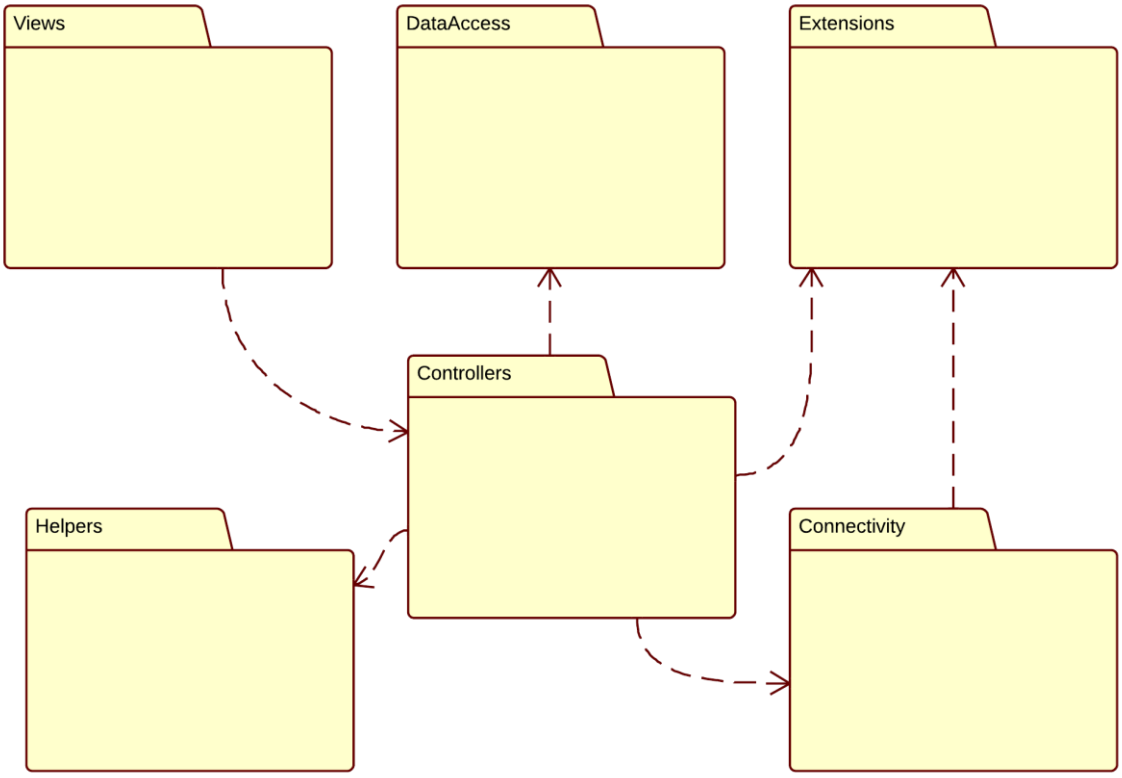


Figura 7.10 – Vista lógica cliente mobile

Los clientes móviles presentan las siguientes componentes:

- **Views**: Contiene las vistas de la aplicación móvil. Es el punto de contacto directo entre el usuario y el sistema.

- **Controllers**: Son controladores locales de la aplicación móvil, que se encargan de interactuar con el resto de las componentes del entorno para presentar la información necesaria a las vistas.

- **DataAccess**: Manipula los datos almacenados en la base de datos local, los cuales son necesarios para mantener interfaces de usuario consistentes.

- **Extensions**: Contiene los *frameworks* y extensiones necesarias para el funcionamiento del sistema, fundamentalmente los *frameworks* de XMPP.

- **Connectivity**: Encapsula la funcionalidad de comunicación con los distintos servidores.

- **Helpers**: Conjunto de clases de apoyo que brindan funcionalidad reutilizable para toda la aplicación.

Tanto los clientes móviles de la plataforma iOS como los de la plataforma Android utilizan la misma organización lógica.

7.4.2.4. Vista Lógica Cliente Web

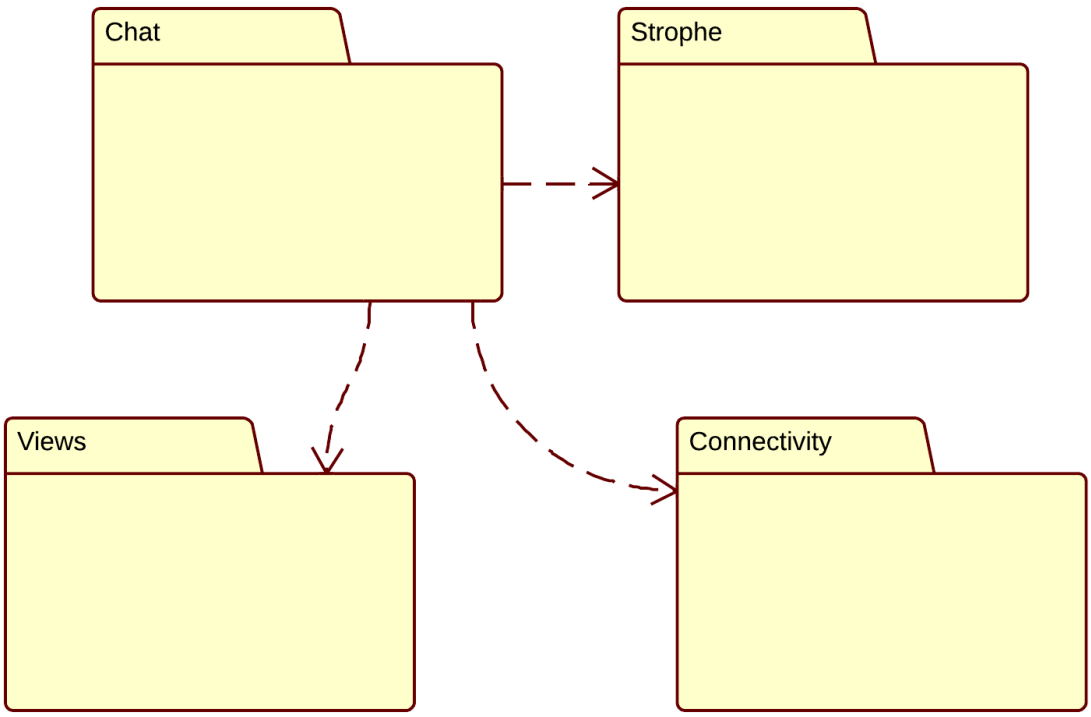


Figura 7.10 – Vista lógica cliente web

Los clientes web presentan las siguientes componentes:

- **Chat**: Componente central. Contiene la lógica central para el funcionamiento del cliente web. Utiliza la funcionalidad brindada por el resto de las componentes.
- **Views**: Contiene las vistas con las cuales interactúa el usuario web.
- **Strophe**: Framework XMPP para javascript. Permite comunicarse con el servidor XMPP desde un *web browser.*
- **Connectivity**: Contiene la funcionalidad necesaria para conectarse con el servidor HTTP.

7.4.3. Interacción (Envío de mensaje)

Como ya explicamos anteriormente (ver sección 6.3.1) el envío de un mensaje desde un cliente está compuesto por dos acciones; el envío del mensaje mismo y la notificación. A continuación se muestran los diagramas de secuencia para ambos casos:

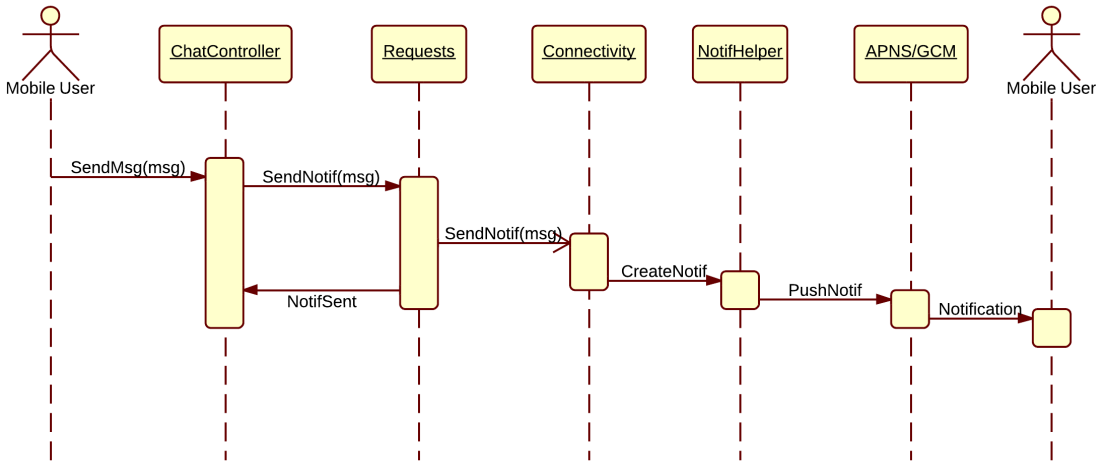


Figura 7.11 – Diagrama de secuencia (envió de notificación)

En el diagrama se ven cinco objetos; ChatController y Requests residen en la aplicación móvil y son los encargados de realizar la solicitud de notificación al servidor HTTP. Connectivity y NotifHelper se encuentran en el servidor HTTP y se encargan de recibir el pedido creado en la aplicación móvil y enviarlo a los servidores de Apple o Google, dependiendo del caso. APNS y GCM son los

servicios de notificación de Apple y Google respectivamente, y se encargan de notificar al dispositivo móvil del destinatario.

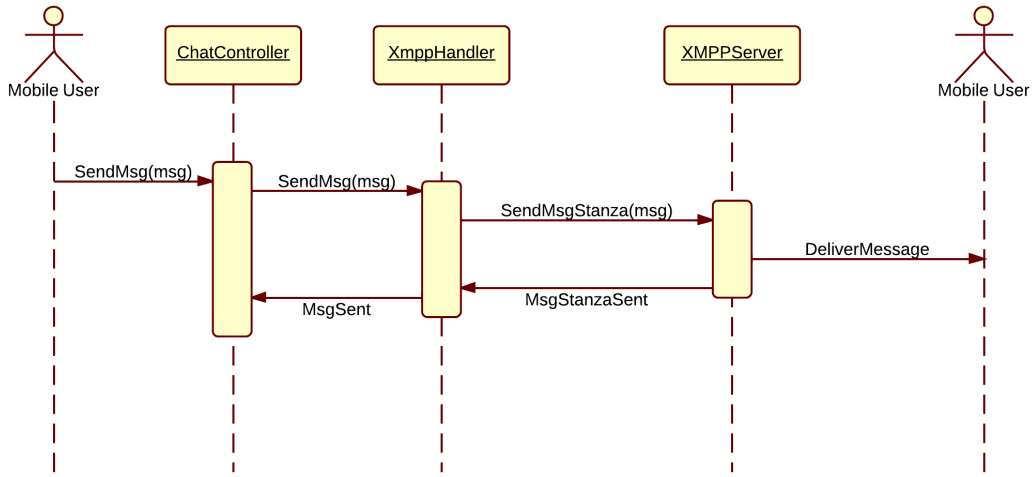


Figura 7.12 – Diagrama de secuencia (envió de mensaje por XMPP)

El mensaje mismo se envía mediante XMPP. Cuando el usuario solicita un envió de mensaje, el objeto ChatController procesa dicha solicitud y se lo comunica al objeto XMPPHandler. XMPPHandler se encarga de encapsular el mensaje, generando la *stanza* (XML en el formato correcto requerido por el protocolo XMPP) y la envía al servidor. El servidor XMPP al recibir el mensaje, lo redirige al destinatario en caso de que este esté conectado, o lo almacena y se queda a la espera de que el destinatario se conecte para enviarlo.

7.5. Evaluación y selección de tecnologías

A continuación se muestra un panorama general de las tecnologías utilizadas para la construcción del sistema:

Figura 7.13 – Panorama general de tecnologías utilizadas

En las siguientes secciones se describen las distintas tecnologías seleccionadas y las razones para su selección. Para ver las distintas tecnologías consideradas ver el Anexo 13: Selección de tecnologías.

7.5.1. Tecnologías servidor HTTP

Para el desarrollo del servidor HTTP se utilizó el lenguaje PHP, principalmente por la experiencia de dos de los miembros del equipo y de nuestro cliente con dicho lenguaje. Además se consideraron importantes las características de PHP que lo hacen un lenguaje ideal para la programación web [8] y la gran comunidad que da soporte a este lenguaje.

Para complementar al lenguaje elegido se utilizó el *framework* Yii MVC, el cual facilita enormemente la creación de *backends* a partir de un modelo de datos [9] .

Como sistema gestor de las bases de datos se utilizó MySql por la experiencia de uso de los miembros del equipo con dicha herramienta.

Para la tecnología de servidor, se seleccionó Apache, por su fácil integración con las tecnologías mencionadas anteriormente.

Para la comunicación con los clientes (tanto móviles como web) se utilizó el protocolo REST, intercambiando los mensajes en formato JSON.

Para la presentación del *backend* de administración se utilizó Javascript mediante la biblioteca JQuery junto con CSS, ya que son las tecnologías estándar para la presentación en la web. Se prefirió utilizar JQuery ante otras opciones como Flash debido al soporte multiplataforma con el que éste cuenta (Flash no es compatible con los dispositivos móviles de Apple).

7.5.2. Tecnologías para clientes móviles

7.5.2.1. Plataforma iOS

Al tener una restricción impuesta por el cliente de que las aplicaciones móviles deberían ser nativas a las plataformas, y iOS solo admite un lenguaje, entonces se utilizó Objective-C como lenguaje de programación. Esto incluye lógica interna de la aplicación, almacenamiento y manejo de datos mediante su framework Core Data y presentación mediante el framework Cocoa Touch.

Para la comunicación con el servidor XMPP, se utilizó la biblioteca XMPPFramework, ya que dentro de las bibliotecas existentes, aunque varias parecían capaces de lograr nuestro cometido, ésta es la que contaba con la mejor documentación, los ejemplos más claros y era la más recomendada.

7.5.2.2. Plataforma Android

La misma restricción aplicada para los clientes iOS se aplicaba a los clientes Android. Aunque en este caso es posible programar aplicaciones en otros lenguajes, Google adoptó al lenguaje Java como lenguaje estándar [10], por lo que la elección del lenguaje fue trivial.

Para la comunicación con el servidor XMPP, se utilizó la biblioteca aSmack por tener la mejor documentación disponible y la mayor base de usuarios.

7.5.3. Tecnologías para clientes web

Los clientes web fueron construidos completamente con Javascript y JQuery. Para su presentación se utilizó CSS, mientras que la comunicación con el servidor HTTP se realizó mediante AJAX.

Para la comunicación con el servidor XMPP se utilizó la biblioteca Strophe.js ya que esta es la que cuenta con el mayor número de usuarios y con la documentación más extensa. Hay que destacar que como los *web browsers* no pueden utilizar el

protocolo XMPP por defecto, se usó el protocolo BOSH, ya que éste es el protocolo estándar utilizado para lograr conexiones XMPP sobre HTTP [11].

7.5.4. Tecnologías para servidor XMPP

Dentro de las tecnologías existentes para crear servidores XMPP, se seleccionó Openfire. Aunque existen otras opciones que permiten alcanzar los objetivos deseados (principalmente el envío de mensajes, el intercambio de presencia y la creación de *chat rooms*), hubieron ciertos factores que fueron determinantes a la hora de seleccionar esta tecnología. Estos factores fueron, principalmente, la facilidad de instalación, la disponibilidad de una interfaz gráfica para la administración del servidor, la cual brinda una forma sencilla y eficiente de llevar a cabo las tareas necesarias, la vigencia que tiene esta tecnología (está respaldada por una empresa que ofrece soporte y la mantiene actualizada hasta la fecha [12]), y la existencia de varias extensiones que amplían sus servicios.

7.6. Validaciones realizadas a la arquitectura

Para la validación de la arquitectura se acudió siempre en primer instancia al cliente, ya que éste pertenece al sector del desarrollo de software y tiene experiencia en este tema. También se acudió a especialistas de la universidad. Hubo dos etapas marcadas de validación, ya que la arquitectura inicial sufrió solo un gran cambio a lo largo del desarrollo del proyecto.

En la primera etapa, en la cual se planteó realizar la mensajería a través de HTTP, se acudió primero al cliente y luego se conversó con el tutor y con especialistas en arquitectura de la universidad. Estos últimos sugirieron buscar métodos de comunicación alternativos, ya que el protocolo seleccionado podría no ser adecuado. En la segunda etapa, tras investigar posibles tecnologías de comunicación, se llegó a la nueva solución con el uso del protocolo XMPP, la cual fue validada por el cliente y el tutor.

7.7. Estrategia de desarrollo

7.7.1. Estrategia normal

Partiendo del análisis de requerimientos, se decidió dividir al sistema en módulos, cada uno de los cuales agruparía un conjunto de funcionalidades relacionadas. Se llegó así a los siguientes módulos: Chat, Support, Services, Administration, Project, Control. Se hizo una estimación de la duración de construcción de cada módulo, y se determinó que los últimos dos módulos no se iban a considerar para el proyecto por la restricción de tiempo.

Una vez determinados los módulos, estos se priorizaron junto con el cliente, y luego se pasó a desarrollar cada uno en el orden determinado. Para cada módulo se realizaron las siguientes actividades:

- Se valida la funcionalidad a construir con el cliente
- Se diseña el modelo de datos para el módulo
- Se codifica la solución para el lado servidor entre todos los desarrolladores del equipo.
- Se codifica la solución para los clientes Android y iOS, cada una asignada a un desarrollador especializado en la plataforma. Este desarrollo se realiza en paralelo.
- Se verifica la interoperabilidad entre las dos plataformas y se arreglan fallas.
- Se realizan pruebas sobre las nuevas funcionalidades implementadas.
- Se presenta el módulo al cliente para su evaluación.

7.7.2. Pruebas de concepto

Como se mencionó en varias oportunidades en este capítulo, al notar que la arquitectura planteada en primera instancia no era adecuada, se decidió modificar la misma y agregar un servidor XMPP. Para realizar dicha modificación, hubo que frenar el desarrollo del sistema y realizar las siguientes tareas:

- Evaluar la nueva tecnología a utilizar junto con el cliente e instalar un servidor de prueba local.

- Codificar las funcionalidades necesarias con la nueva tecnología en un sistema aislado y probar que estas funcionen de manera satisfactoria.

- Integrar las nuevas funcionalidades a la solución principal y verificar que la integración sea satisfactoria.

- Eliminar las versiones anteriores de dichas funcionalidades en caso de que sea necesario.

7.8. Herramientas y ambientes utilizados para el desarrollo

Para la programación del lado servidor y las componentes web, se podría haber utilizado cualquier editor de texto, pero se optó por el IDE Netbeans, ya que éste tiene soporte de *color coding y intellisense* para PHP, Javascript y CSS, lo cual aceleró mucho el proceso de desarrollo.

Para el desarrollo en la plataforma iOS se utilizó XCode, el IDE de desarrollo de Apple, tanto para iOS como para OSX. En este caso no se consideraron otras opciones, ya que parecen no existir, o de existir no están aprobadas por Apple. Además, XCode está diseñado específicamente para integrarse con el lenguaje Objective-C por lo que otras opciones podrían ser contraproducentes.

Para la plataforma Android se utilizaron las Android Developer Tools (ADT) de Eclipse ya que en el momento de construcción era el IDE recomendado por Google. Un nuevo IDE, llamado Android Studio parece ser el reemplazo, pero aún está en versión beta, por lo que se descartó esta opción.

Como sistema operativo para el ambiente de desarrollo se utilizó OSX Mavericks, por las restricciones del IDE XCode. Se utilizó una terminal con sistema operativo Windows 7 como repositorio de versionado. Para el ambiente de producción se utilizó un servidor Linux por recomendación de la empresa de *hosting*.

Se utilizaron los browsers Chrome, Firefox, Safari e Internet Explorer junto con sus *developer tools* para el desarrollo web. Además se utilizó el plugin SQLite Manager de Firefox para visualizar las bases de datos de los dispositivos móviles y phpMyAdmin para las bases de datos de los servidores.

Para el diseño y manipulación de las distintas imágenes utilizadas en el sistema se utilizaron las herramientas Photoshop y Ilustrator.

Para el manejo de versiones se utilizó Versions y como herramienta de FTP se utilizó FileZilla.

7.9. Principales paquetes o módulos implementados

Se implementaron cuatro módulos del sistema, cada uno de los cuales agrupa un conjunto de funcionalidades relacionadas. Estos fueron:

Administration: Este módulo abarca toda la funcionalidad de registro y administración de empresas y sus usuarios. Es un *backend* web, al que solo pueden ingresar usuarios con privilegios de administración. Es el único módulo al cual se accede únicamente por la web, sin necesidad de un dispositivo móvil.

Chat: Es el módulo que brinda la funcionalidad central de la aplicación, o sea el envío de mensajes instantáneos entre dispositivos móviles. Su implementación significó desarrollar parte de la API REST del servidor HTTP para manejar las solicitudes de contacto, los contactos en sí, las preferencias de usuario, el desarrollo de la API XMPP para la creación de conversaciones entre dos clientes, o múltiples mediante *group chats,* y la creación del servidor XMPP. También abarcó el diseño y desarrollo de las interfaces gráficas necesarias.

Support: Este módulo permite conectar a usuarios anónimos desde la web con los clientes móviles de usuarios registrados. Su implementación incluye los siguientes ítems:

- Un *plugin* que se instala en servidores externos y accede al servidor HTTP mediante CORS (ver Anexo 11: Plugin web: Detalle e instalación).
- La creación de parte de la API REST del servidor HTTP que abarca la comunicación del cliente web con este servidor y la comunicación de los clientes móviles con el mismo servidor, para la recepción de solicitudes de consulta.
- La creación de las interfaces gráficas necesarias para el manejo de las consultas originadas desde la web, junto a su lógica asociada, y la funcionalidad necesaria para los servicios XMPP, tanto en clientes móviles como web.

Services: Módulo que permite interconectar a distintas empresas, con el fin de que una empresa pueda buscar los servicios de otras. Para este módulo se tuvo que implementar parte de la API REST, la cual se encarga de obtener la información de las empresas. También, en los clientes móviles, se creó la lógica para realizar los listados, clasificados por rubro y priorizados por reputación, de los servicios que las empresas ofrecen y los medios de comunicación entre las empresas (aprovechando la funcionalidad del módulo de chat).

8. Aseguramiento de la calidad

8.1. Introducción

En este capítulo se describen todas aquellas actividades relacionadas a la calidad del proyecto. Para llegar a un mínimo de calidad fue necesario planificar-hacer-revisar-actuar actividades a lo largo del proyecto, atacando varias áreas de la ingeniería de software y en varios puntos en el tiempo.

8.2. Descripción del Proceso de SQA

Para comenzar a describir este capítulo, lo más indicado es empezar por el proceso de SQA. Este proceso tuvo varias fases. Al inicio del proyecto, se definieron los estándares a utilizar, para asegurar la calidad en el proyecto, y se definieron las actividades que se realizarían a lo largo del mismo. Luego, una vez que se había determinado la metodología de trabajo, se estableció que los *sprints* deberían:

- Definir tareas
- Definir pruebas
- Establecer reuniones
- Incorporar lecciones aprendidas en la ejecución del *sprint*, tanto de los aspectos positivos como los negativos

Por último llegando al final del proyecto se llevaron a cabo una serie de pruebas de usabilidad. El siguiente diagrama ayuda a visualizar el proceso.

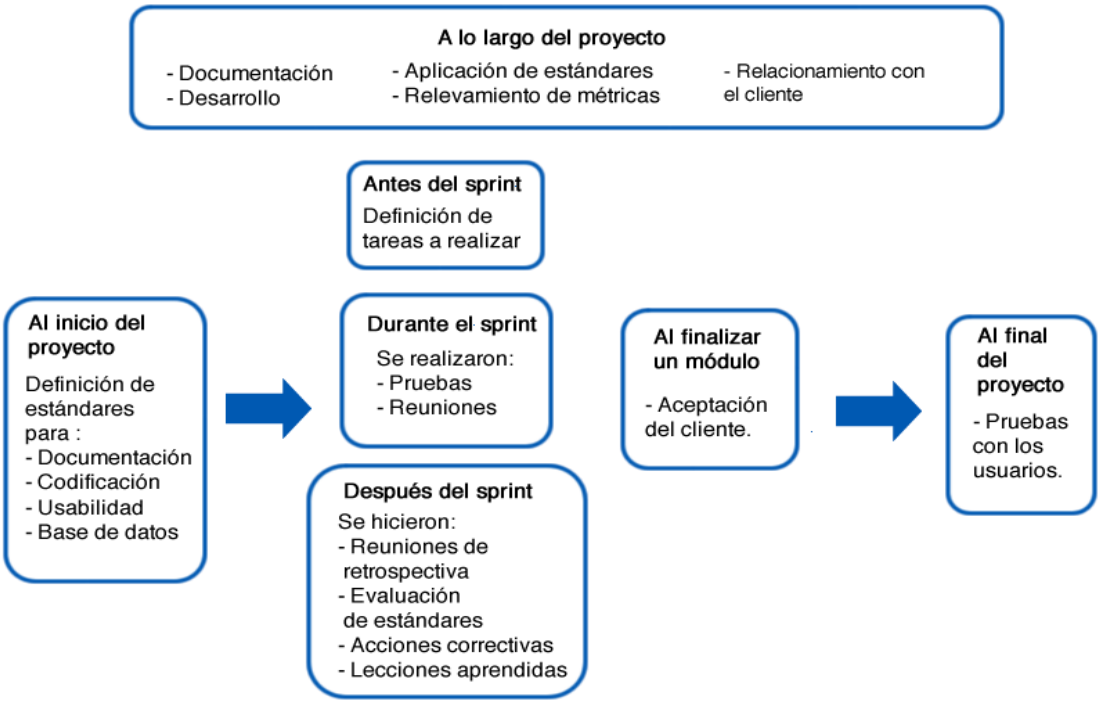


Figura 8.1 – Proceso de SQA

8.3. Visión de la Calidad

Para este proyecto se optó por perseguir una visión de la calidad específica, la cual se basa en cinco principios: mejora continua, satisfacción del cliente, disminuir re-trabajo, prevenir y corregir errores.

- **Mejora continua.** Cuando se desarrolla un producto en base a una metodología ágil, el cierre de cada *sprint* presenta una oportunidad para buscar mejoras e implementarlas en la siguiente corrida.

- **Satisfacer al cliente.** Es vital entender y analizar al cliente de modo de satisfacer sus expectativas.

- **Disminuir re-trabajo.** A medida que un grupo de personas crece como equipo debería ir mejorando su estimación en las tareas y rendimiento previniendo así el re-trabajo de tareas.

- **Prevenir errores.** Los costos de prevenir errores son generalmente mucho menores al costo de corregirlos, por lo tanto se buscará prevenir la mayor cantidad de errores posibles.

- **Corregir errores.** Aunque se intente prevenir errores no podremos prevenir todos. Aquellos que se filtren deben ser corregidos para lograr un producto de calidad.

Algunos de estos principios son muy difíciles de medir y por ello se hizo foco en ciertas características de los mismos. Por lo tanto, a raíz de esta visión bajamos a tierra los objetivos que veremos en la siguiente sección.

8.4. Objetivos de Calidad

Para satisfacer al cliente fue necesario atacar dos grandes conceptos que existen dentro de la ingeniería de software: el producto y el proceso que nos lleva a ese producto. Teniendo en cuenta estos conceptos, la visión establecida, las herramientas y el tiempo disponible se establecieron los siguientes objetivos.

8.4.1. Objetivos del producto

- Satisfacer al cliente.
- Satisfacer al usuario.
- Tener un producto con bajo porcentaje de defectos.

8.4.2. Objetivos del proceso

- Evaluar impacto de re-trabajo.
- Evaluar impacto de mejoras introducidas.

8.5. Actividades

Para poder satisfacer los objetivos anteriormente nombrados se tuvieron que llevar a cabo varias actividades. Éstas fueron: validaciones, verificaciones, revisiones técnicas con docentes, estándares de programación, estándares de documentación, una puesta a punto y pruebas.

8.5.1. Actividades y Entregables

A lo largo de proceso de ingeniería de software se fueron generando distintos entregables (productos) en sus respectivas fases. A continuación se presenta un plan de calidad indicando que actividades se hicieron en cada fase.

La nomenclatura usada en la tabla es:

Revisión	Rev
Validación	Val
Verificación	Verif
Estándares	Est
Testing	Test

Figura 8.2 – Nomenclatura

Fase	Producto	Rev	Verif	Val	Est	Test
Ingeniería de Requerimientos	ESRE			X		
	Product Backlog		X	X		
Diseño de Arquitectura	Arquitectura de Sistema	X				
Desarrollo	Aplicación Móvil					X
	Aplicación Web					X
	Código Fuente				X	
Pruebas	Plan de pruebas	X				
Gestión	Alcance	X		X		
	Documentos de Riesgos	X				
	Cronograma	X				
	Presentación para la revisión	X				
SQA	Plan de Calidad	X				
	Plan de SQA	X				
SCM	Plan de SCM					
	Respaldos					

Figura 8.3 – Plan de calidad

Se realizaron revisiones para controlar el avance de ciertos productos de software como se puede apreciar en la tabla. Ocasionalmente sucedía que una revisión

encadenaba más revisiones, por ejemplo el caso del diseño arquitectónico. Primero se revisó con un docente del área de arquitectura, a raíz de esto fue necesario informar al cliente acerca de determinados aspectos de la arquitectura los cuales podían afectar el desarrollo del sistema.

Las validaciones fueron hechas únicamente para los productos de ingeniería de requerimientos y del alcance. Fue necesario validar que los requerimientos deseados por el cliente eran exactamente los que se habían documentado. Para el alcance se tuvo que validar con el tutor si era lo suficientemente extenso como para un proyecto de ingeniería.

8.5.2. Definición de Estándares

Luego de haber pasado la etapa inicial del proyecto, eligiendo una metodología de trabajo acorde a lo que demandaba el mismo, el equipo se enfoco en definir los estándares a utilizar a lo largo del proyecto.

Estándares de Documentación

Con el propósito de mantener un orden y facilitar la integración de los documentos elaborados para la documentación, se utilizaron los estándares de documentación brindados por la Universidad ORT.

Estándares de Codificación

Para facilitar la comprensión y mantenibilidad del código se utilizaron varios estándares de codificación.

Para los lenguajes Php, Objective C y Java se utilizaron estándares de programación, los mismos se encuentran en el Anexo 3 : Estándares de Desarrollo.

Para los lenguajes Html, Css y JavaScript se utilizaron estándares especialmente dedicados a los mismos sumado a los de Php, Objective C y Java. Los mismos se encuentran en las secciónes correspondientes a Html, Css y JavaScript en el Anexo 3 - Estándares de Desarrollo.

Criterios para la Base de Datos

Para crear las tablas y atributos en la base de datos, se decidió establecer ciertos criterios definidos por el propio equipo que se detallan en el Anexo 3: Estándares de Desarrollo.

Estándares de Usabilidad

Se adoptaron las Heurísticas de Nielsen para que el usuario (tanto web como móvil) se sienta a gusto y pueda utilizar las aplicaciones de forma fluida y sin complicaciones. Las heurísticas se pueden ver en el Anexo 3: Estándares de Desarrollo.

Todos estos lineamientos contenidos en los estándares se tuvieron siempre presentes para desarrollar las aplicaciones. El encargado de calidad era responsable de controlar que se estuvieran cumpliendo, esto se hacía en las reuniones de finalización de los *sprints*.

8.5.3. Reuniones de Equipo

Los integrantes del equipo trabajaron mayoritariamente en un mismo lugar físico. Claro que existieron momentos de trabajo individual, pero el rendimiento aumentaba al trabajar todos en un mismo espacio. Esta forma de trabajo permitía evacuar dudas y tomar decisiones de forma rápida y eficiente.

Más allá de trabajar en un mismo espacio se establecieron instancias especiales en las cuales el equipo se reunía para enfrentar ciertos temas. De forma de lograr un avance parejo y conocer la situación del resto de los integrantes, se realizaron reuniones llamadas *daily meetings* en las cuales se respondían las siguientes preguntas:

¿qué se hizo el día de ayer?

¿qué problemas encontramos?

¿qué vamos a hacer hoy?

Al finalizar el *sprint* se hacían dos reuniones, de *review* y retrospectivas. En la de review se veía a que se había llegado y si coincidía con lo que el cliente había pedido. En las de retrospectiva se analizaba el cumplimiento de los estándares

(mediante muestreos), las lecciones aprendidas y las acciones correctivas, de forma de no repetir los mismos errores en nuevos *sprints*.

8.6. Pruebas

Respecto a lo que refiere a pruebas hay un capítulo especialmente dedicado el cual explica la metodología de pruebas utilizadas. Además, se muestran métricas de incidentes y dedicación en horas por área. En la siguiente sección se presentan el resto de las métricas utilizadas.

8.7. Métricas

Luego de establecidos los objetivos de calidad, el equipo tuvo que buscar la forma de asegurarse que los mismos estaban siendo cumplidos. Para esto, se propusieron un conjunto de métricas las cuales permitieron ir conociendo el estado de los objetivos.

Las métricas utilizadas fueron:

- Cantidad de mejoras por *Sprint*: Son la cantidad de mejoras introducidas por el equipo o el cliente en un *sprint*. Las mejoras propuestas fueron a nivel de estrategias de codificación y tecnologías a utilizar.
- Cantidad de incidentes por *sprint*: Son la cantidad de incidentes encontrados en un *sprint*. La evolución de esta métrica se encuentra en el capítulo de pruebas.
- Horas de re-trabajo por *Sprint*: Son la cantidad de horas dedicadas a rehacer funcionalidades ya desarrolladas de un *sprint*.
- Porcentaje de aceptación del cliente: Es el porcentaje de aceptación t medido en las instancias de aceptación del cliente al final de cada módulo.
- Porcentaje de satisfacción de la usabilidad de parte del usuario: Es el porcentaje de satisfacción con foco en la usabilidad orientada al cliente, medido mediante encuestas basadas en los principios de Nielsen.

Cantidad de Mejoras por *Sprint*

A lo largo del proyecto existieron dos grandes mejoras. Las mismas se dieron en el *sprint* 2 y el 5. En la siguiente sección se verá como impactaron de forma cuantitativa.

En el *sprint* 2 el cliente impuso una implementación diferente a la que había puesto en práctica el equipo para realizar las actualizaciones de los dispositivos móviles. Ésta favoreció enormemente el funcionamiento de los dispositivos móviles.

En el *sprint* 5 fue el equipo quien propuso la mejora. Debido al impacto de la mejora se decidió abortar el *sprint*. La idea era introducir un protocolo contemporáneo llamado XMPP para la comunicación de dispositivos móviles. Al comenzar a implementar el protocolo el equipo se encontró con que no contaba con mucha documentación acerca del mismo, pero de todas formas esto no detuvo al equipo, ya que este protocolo aceleraba notoriamente la comunicación entre dispositivos y no podía ser descartada.

En el análisis de la siguiente métrica (horas de re-trabajo por sprint) se apreciarán las conclusiones de esta métrica.

Horas de re-trabajo por *sprint*

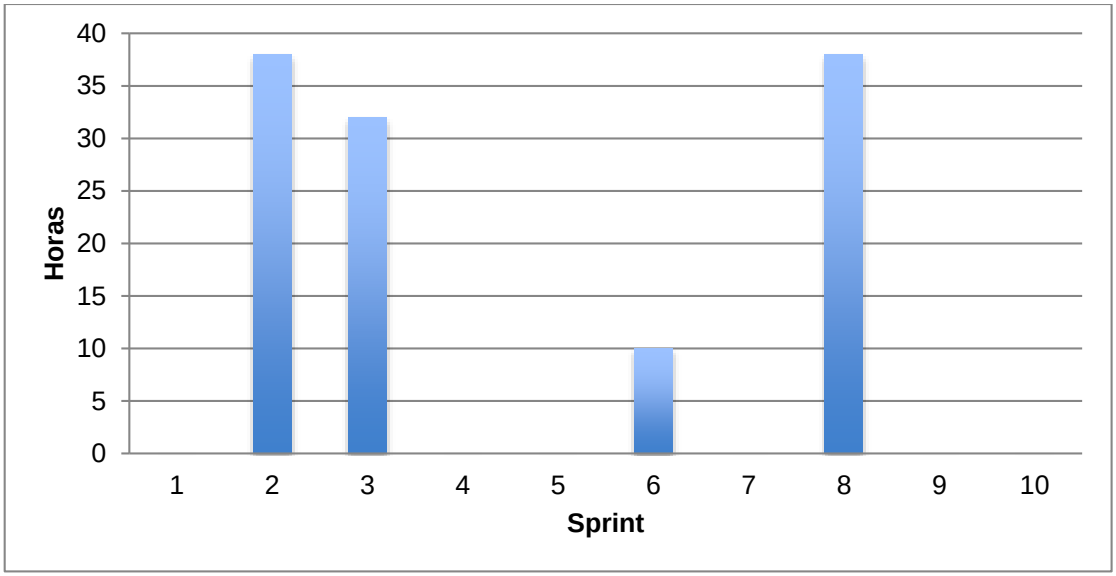


Figura 8.4 – Gráfica de re-trabajo

La métrica anterior tiene una relación directa con las horas de re-trabajo por *sprint*. A raíz de las mejoras introducidas se tuvieron que dedicar horas para realizarlas. Por lo tanto las dos mejoras introducidas generaron re-trabajo en el *sprint* 2 y 6. Es importante aclarar que las horas dedicadas a re-trabajo en lo que figura como el *sprint* 2 corresponden a ese mismo *sprint* a diferencia del sexto que provino del *sprint* 5 (en el capítulo de Gestión se explica la razón de esto).

En la gráfica se ve que en el *sprint* 3 también hubo re-trabajo (originado en ese mismo *sprint*) debido a una mala práctica. El equipo de desarrollo estaba utilizando

el *registration id* provisto por GCM y APNS para identificar un dispositivo móvil. Más allá de que esto no era incorrecto, el equipo manejaba de otra forma la identificación de entidades, y por eso se decidió cambiar la identificación del dispositivo a la forma clásica y tener un único enfoque con este propósito.

El último *sprint* con re-trabajo fue el 8. Esto fue debido al incremento en complejidad de las notificaciones, la cual surgió a raíz de la creación de grupos. Hubo que replantear el manejo de las notificaciones ya que la aplicación creció notoriamente y generó varios escenarios posibles.

Esta métrica sumada a la anterior sirvió para dos cosas. En primer lugar y referido a las mejoras, muestran que éstas no deben realizarse de corrido en un *sprint*. Si se estima que la mejora va a tener un fuerte impacto es preferible abortar el *sprint*, re planificar y volver a comenzar. En segundo lugar, nos muestran que el rehacer funcionalidades puede surgir en cualquier punto del proyecto debido a que el producto va mutando y es necesario adaptarse de forma rápida para poder satisfacer al cliente.

Porcentaje de aceptación del cliente

Como describe el capítulo de gestión, el cliente fue validando el producto durante los *sprints* y al finalizar cada módulo. Más allá de que la validación a lo largo de los *sprints* fue muy positiva para la elaboración del producto, fue la evaluación por módulo la que nos permitió obtener resultados más tangibles.

Para medir la satisfacción del cliente se estableció la siguiente metodología. En primer lugar se definió una escala con cuatro valores posibles (porcentajes), los mismos son : 0 - 25 - 50 -75 -100. Luego, se le entregaba el producto al cliente para que el mismo probara todas las funcionalidades de un módulo específico. Para esto el equipo listaba todas las *user stories* del *product backlog* que contenían funcionalidades del producto y el cliente probaba una a una las *user stories* listadas. Al terminar de probar una *user story* dictaba al equipo el valor de la escala que consideraba adecuado para expresar la aceptación de la funcionalidad evaluada (para ver detalles de los resultados ir a Anexo 8: Pruebas de Aceptación). A continuación se presenta la satisfacción del cliente por módulo.

- Módulo de Administración: 92% aceptación - 8% rechazo.
- Módulo de Comunicación: 85% aceptación - 15% rechazo.
- Módulo de Soporte: 88% aceptación – 12% rechazo.
- Módulo de Servicios: 90% aceptación – 10% rechazo.

- Resultado final de la satisfacción del cliente:

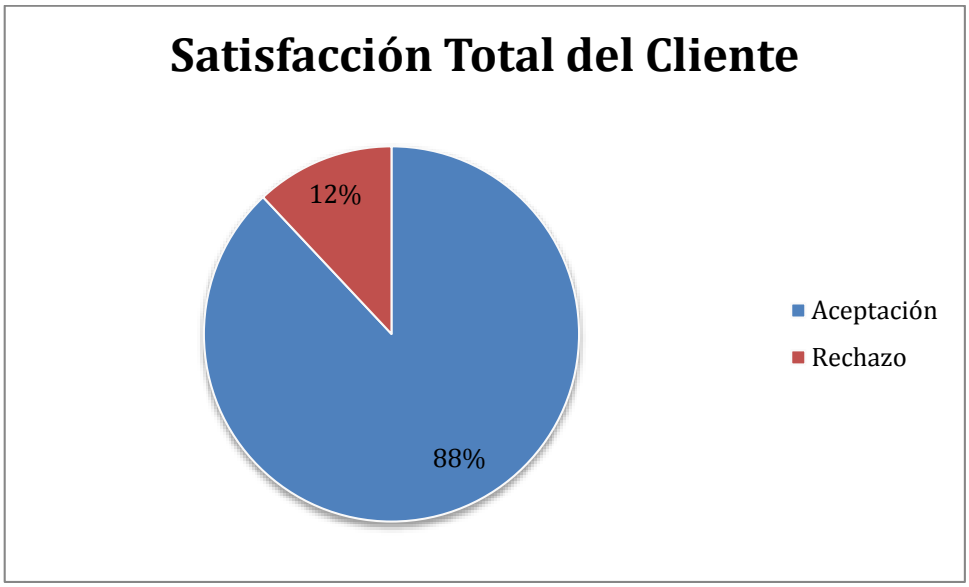


Figura 8.5 – Satisfacción del cliente

Afortunadamente se logró una gran satisfacción por parte del cliente. No hubo ningún módulo con aceptación por debajo de 85% lo cual el equipo considera muy satisfactorio. El porcentaje más bajo proviene del módulo de Comunicación. Esto fue debido a la gran complejidad que tuvo el módulo. Para el de soporte, el equipo tuvo que enfrentarse con funcionalidades muy parecidas al de comunicación. Por esta razón se reutilizó gran parte del código para su implementación.

Porcentaje de satisfacción de la usabilidad de parte del usuario

Para medir la satisfacción de usuario a nivel de usabilidad se formuló un cuestionario basado en los principios de Nielsen. Se reformularon los enunciados adaptándolo a las características del proyecto. Por lo tanto se obtuvo un cuestionario con diez principios los cuales fueron entregados a un grupo selecto de usuarios para obtener el grado de satisfacción de los mismos.

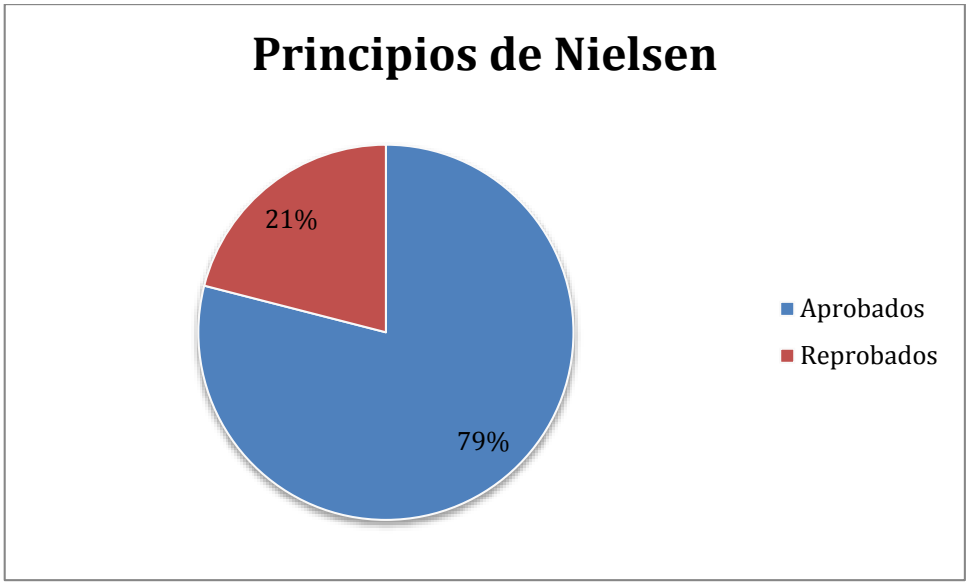


Figura 8.6 – Evaluación de satisfacción

El equipo consideró que tener por encima del 75% de aceptación en los principios de Nielsen era aceptable. Los resultados de las encuestas muestran que el nivel de aceptación sobrepasó el 75% deseado. Además se analizó que el porcentaje de rechazo no tuviese aspectos críticos relacionados a la funcionalidad. Si hubiera sido así, el equipo de desarrollo hubiera tenido que focalizarse en reparar esos aspectos y corregirlos hasta llegar a satisfacer los mismos. Más allá de haber llegado al mínimo establecido, el equipo consideró que para futuras versiones del producto el nivel de satisfacción debería ser mayor a un 85%.

La principal deficiencia del producto se encontró en el principio que establece que el sistema debe permitir deshacer una acción siempre y cuando la acción sea de tipo reversible. Lo que sucede es que hay algunas acciones, como “aceptar un contacto”, que no pueden deshacerse. La aplicación fue pensada para no permitir eliminar contactos pero para futuras versiones se permitirá realizar ésta acción y otras del estilo.

9. Pruebas

9.1. Introducción

En la primera fase del proyecto, cuando se realizó la planificación general, uno de los planes involucrados fue el Plan de Pruebas. El mismo incluía la definición de los tipos de pruebas que era necesario realizar, la definición de un proceso o secuencia de tareas mínimas que deberían realizarse para ejecutar las pruebas funcionales, la definición de cómo y cuándo se llevarían a cabo las pruebas de integración, cuándo deberían realizarse las pruebas de aceptación, y una estimación de cuándo se podrían realizar pruebas de usabilidad y pruebas alfa.

En esta sección, se explicarán cada uno de los aspectos mencionados en el párrafo anterior y de qué manera colaboran con la calidad del producto y del proceso. A continuación describiremos cómo se llevaron a cabo las pruebas funcionales, de integración, de aceptación, pruebas Alfa y de usabilidad.

9.2 Proceso de pruebas

El proceso de pruebas, se definió con el fin de adaptar la ejecución de este tipo de tareas a la realidad y características del equipo. Consistía en una secuencia de pasos que se debían tener en cuenta, sin perder el foco en la flexibilidad y, sobre todo, la dinámica, brindando libertad de acción a la persona que se encontrara ejecutando las pruebas. Este proceso, se representó mediante un diagrama de flujo, el cual se colocó en el repositorio de documentos, para que estuviera disponible a todos los integrantes del equipo. Este último punto era importante, ya que todos los miembros del equipo ejecutaban tareas de *testing*.

Para comenzar el procedimiento, el encargado de ejecutar la prueba, tomaba una *user story* que no hubiera sido desarrollada por él y se encontrara en estado "*Done*" con la mayor prioridad posible. Luego, se leía la descripción y las notas realizadas sobre la misma, y por último se discutían los términos de aceptación, asignados para esa *user story*, con los demás integrantes del equipo. Este intercambio de ideas, se utilizaba con el fin de explorar los límites de la *user story*.

De esa forma, quién estaba por ejecutar las pruebas, podía idear o tomar nota de los posibles estímulos que podría intentar sobre el sistema. A partir de ese momento, se ejecutaban las pruebas funcionales o de integración, según correspondiera. Las pruebas se daban por finalizadas cuando el ejecutante consideraba que había cubierto el universo de estímulos posibles para una *user story*.

En caso de que se encontrase alguna falla, se procedía a registrar un incidente en la herramienta de gestión por cada una de ellas. Este proceso se explicará en detalle más adelante. Si la falla era trivial, el mismo ejecutante, luego de registrar el incidente, podía repararla. El diagrama de flujo del proceso de pruebas, se puede observar en el Anexo 13 : Plan de pruebas.

9.3. Pruebas de aceptación

Las pruebas de aceptación se adaptaron a la metodología de trabajo del equipo, en este caso, se realizaban luego de la finalización de cada módulo. Esta ceremonia, se realizaba en la Reunión Semanal, que se explicará en el capítulo de Gerencia, y podía llevarse a cabo al finalizar un sprint o durante el mismo.

Para llevarlas a cabo, el equipo se reunía con el *product owner* y realizaba una introducción de las nuevas funcionalidades que se habían desarrollado. Luego, se realizaba una demostración por parte del equipo, y por último, el cliente utilizaba la herramienta, realizaba preguntas y comentarios, críticas o brindaba su opinión sobre posibles mejoras.

9.4. Pruebas Alfa

Un punto fuerte, desde el punto de vista de la calidad que se planificó para el proyecto, fue realizar una versión Alfa del producto y liberarla a un conjunto de usuarios finales para analizar y probar su comportamiento. El cliente, brindó la posibilidad de que algunas empresas, clientes de Moon Ideas, utilizaran el producto durante una semana y registraran todas las fallas que encontraran. Estas pruebas, eran lo más similar a un ambiente de producción que se podía obtener y ofició como filtro para depurar la mayor cantidad de fallas posible antes de la entrega final.

Uno de los problemas enfrentados, que se explicarán en la sección de Gestión de Riesgos, fue los retrasos en el desarrollo de un módulo central para las funcionalidades del producto. Por ese motivo, la liberación de la versión se postergó casi un mes. Esto impactó en varios frentes del proyecto. En primer lugar, elevó el riesgo de no tener suficiente tiempo para corregir la totalidad de las fallas detectadas en estas pruebas, en caso de que alcanzaran un número importante. En segundo lugar, imposibilitó realizar una segunda instancia de pruebas de usabilidad, lo cual se explicará más adelante cuando se describan dichas pruebas.

Para realizar las pruebas Alfa, uno de los objetivos del equipo para mantener un mínimo nivel de calidad, era no contar con fallas pendientes de reparación, con nivel de criticidad medio y/o alto, luego que se finalizaran las pruebas funcionales y de integración. En primer lugar, porque eran el tipo de fallas que entorpecían o no

permitían el correcto funcionamiento del producto y segundo, porque eran las que insumían mayor esfuerzo en la reparación.

El resultado final fue sumamente positivo. Más allá de las fallas registradas en este período, lo que se destaca de esta actividad, fue brindarle al cliente la tranquilidad de que la primera versión de su producto era una realidad. Se podría “salir en producción”, tal vez con una versión Beta para mantener el producto funcionando y a su vez continuar desarrollando más funcionalidades, pero la herramienta con los servicios deseados hoy funciona.

9.4.1. Metodología

Para llevar a cabo las pruebas, primero, se instaló la aplicación en el servidor del cliente y el módulo de soporte en la página web. Luego, se le explicó a los empleados involucrados como se iba a proceder, por qué se estaba realizando la actividad y como debían registrar la información para colaborar con el equipo.

Los empleados, utilizaron la herramienta durante una semana y registraron los incidentes encontrados en planillas. Cada planilla, indicaba el nivel de criticidad de la falla encontrada (ésta escala se les había explicado antes de comenzar la actividad), una descripción de la tarea que estaban realizando y por último, una descripción de la falla. Para facilitar la comunicación, se les pidió que en caso que fuera necesario, adjuntaran imágenes como parte de la evidencia. Cada empleado, luego de la finalización del período de pruebas, entregó un archivo con todas las planillas que había registrado.

Luego de recibir los archivos por parte de los empleados, el equipo realizó una depuración de los resultados, registrando todos los incidentes en una única planilla. El objetivo, era tener un registro único de incidentes, clasificados por nivel de criticidad y evitar los registros repetidos.

El total de incidentes registrados en las pruebas alfa, serán reparados luego de la entrega de la documentación final por cuestiones de tiempo y recursos disponibles.

9.5. Pruebas de usabilidad

En un principio, el plan de pruebas de usabilidad contaba con dos instancias. La primera se iba a realizar aprovechando las pruebas Alfa y la segunda, luego de realizar correcciones, a partir de las opiniones relevadas en la primera instancia. De esta manera, se iba a poder detectar la percepción que tenían los potenciales usuarios del producto. En caso de que no fuera buena, se iba a tener tiempo de mejorarlo para la entrega final. Por otro lado si la respuesta fuera positiva, se iba a

poder continuar trabajando en la misma línea y modificando simplemente aspectos estéticos del producto.

Lamentablemente, se presentaron algunos inconvenientes, hubo retrasos en la construcción del producto y esto llevó a desestimar la viabilidad de realizar dos instancias de encuestas. Los retrasos en la construcción del producto también retrasaron la ejecución de las pruebas Alfa, por este motivo, se re-planificaron para ejecutar ambas en la última semana de julio. En definitiva, se ejecutaron las pruebas Alfa y luego se realizó la evaluación de satisfacción, basada en los diez principios de Nielsen. La planilla que contiene la evaluación, se encuentra en el Anexo 9: Pruebas de Usabilidad.

9.6. Registro de incidentes

El registro de incidentes prestaba dos tipos de utilidades que debemos destacar. En primer lugar, la utilidad operativa, que le permitía al *tester* de turno transmitirle información, lo suficientemente precisa, al resto de los integrantes del equipo que pudieran encargarse de resolver el incidente. Y en segundo lugar, la utilidad desde el punto de vista de la gestión, ya que permitía categorizar la información y medir las diferentes tareas de la operativa.

Durante la ejecución de las pruebas, cuando se encontraba una falla en el sistema, se procedía a registrar la misma en la herramienta de gestión. Para el registro, se debían completar un mínimo de campos que el equipo definió como necesarios para una eficiente identificación y localización de las fallas. Desde el punto de vista operativo, resultaron claves campos como: el título de identificación del defecto, el sprint en el cual se detectó y una descripción detallada, en la que se podían agregar imágenes como parte de la evidencia para identificar puntos específicos del sistema.

Por otro lado, desde el punto de vista de gestión, se tenía en cuenta: el valor en *story points* estimados, donde se asignaba siguiendo la sucesión de Fibonacci, un valor que correspondiera con el esfuerzo; el estado de la falla (si estaba reparada, en reparación o se iba a reparar en el futuro), y por último, la prioridad. El nivel de prioridad, se correspondía con el nivel de criticidad de la falla, es decir, fallas críticas se corresponden con prioridad alta y fallas triviales con prioridad baja.

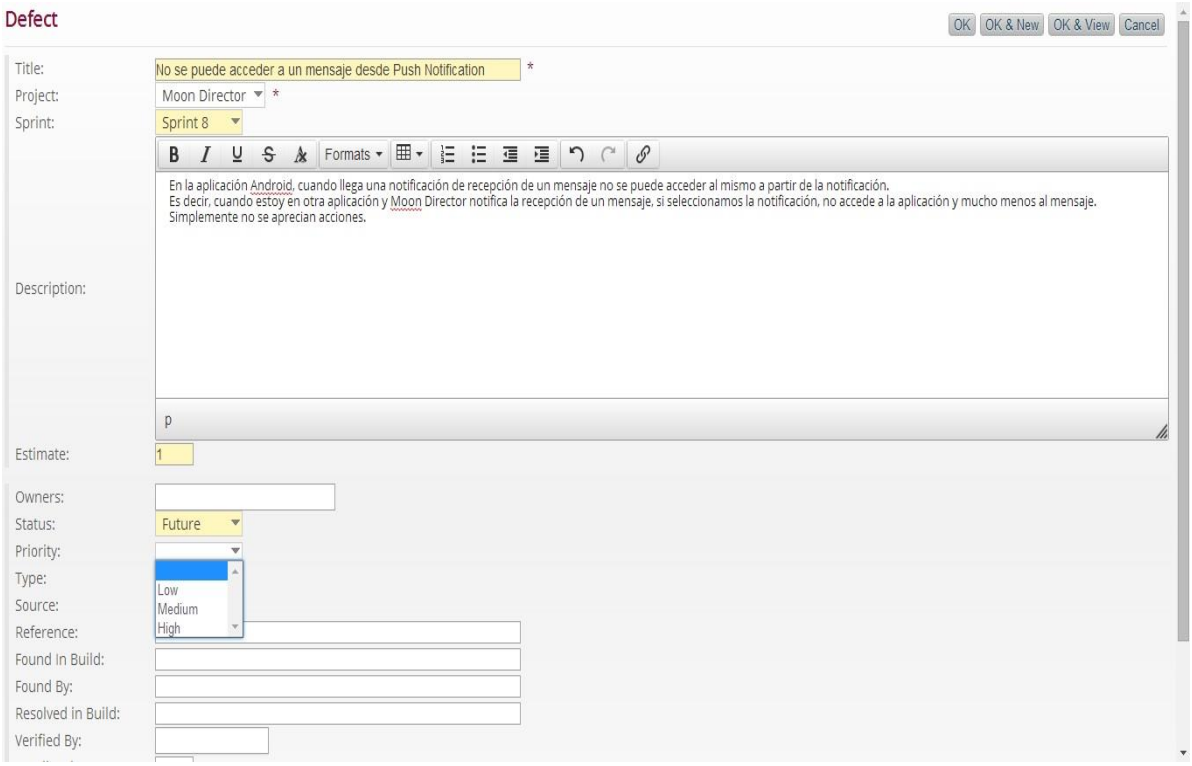


Figura 9.1 - Ventana de registro de incidentes.

La herramienta, ofrece muchos más campos que los mencionados anteriormente, como por ejemplo el propietario de la tarea, quién verificó la falla o el tipo de falla. Ofrece tanto detalle como amerite la gestión. En nuestro caso, como se trataba de un equipo de trabajo reducido, donde la comunicación era muy fluida, se definieron como prioritarios únicamente los campos mencionados. En la metodología utilizada no se maximizó el potencial de la herramienta, sino que se utilizó básicamente para actividades de gestión de baja complejidad. Explotar esta herramienta hubiese permitido, entre otras cosas, generar informes cuantitativos o sincronización de las fallas registradas con las correcciones en el repositorio de datos. Esto hubiera significado un paso muy importante en materia de calidad, pero también hubiese tenido un costo muy importante en materia de tiempo y dedicación.

9.7. Métricas

9.7.1. Cantidad de fallas por sprint

La cantidad de fallas por sprint, tuvo como objetivo monitorear la calidad del proceso de desarrollo del sistema. La evolución del valor de esta métrica, fue un indicador de qué tan bien se estaban realizando las actividades de desarrollo del sistema.

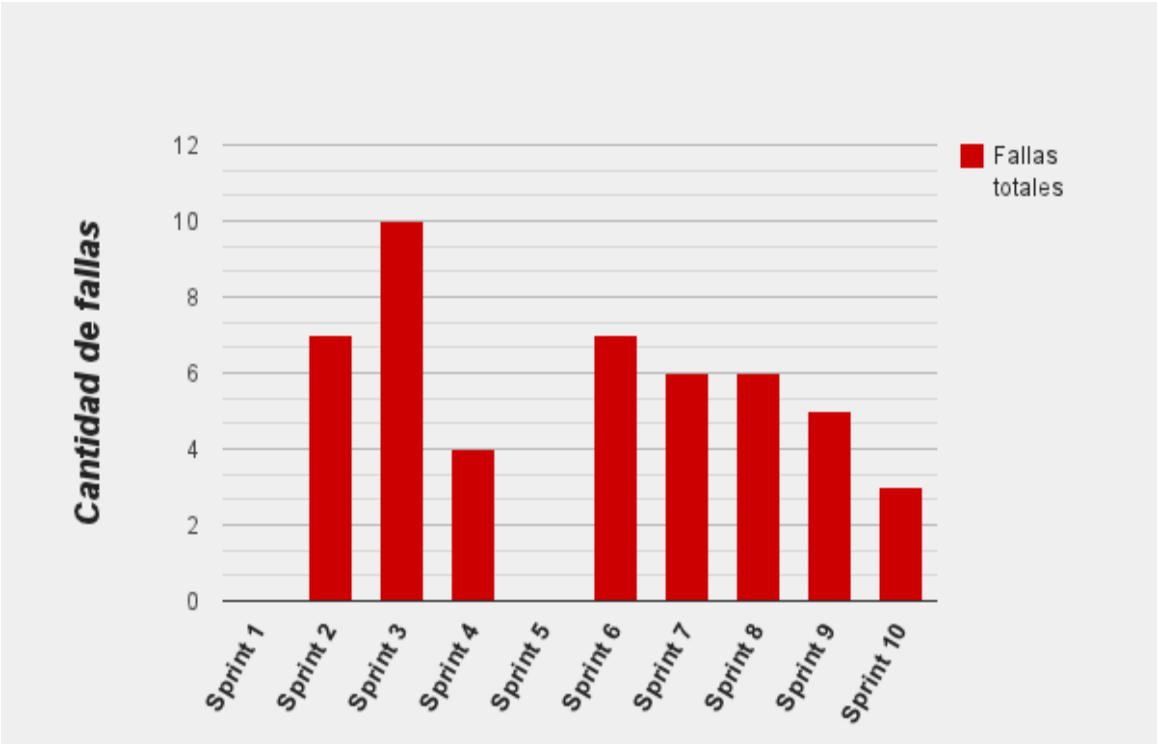


Figura 9.2 - Cantidad de fallas encontradas por *sprint.*

9.7.2. Cantidad de fallas por nivel de criticidad

Se definieron tres niveles criticidad, alineados a los niveles de prioridad que la herramienta de gestión permitía asignarle a las fallas y/o tareas.
Esta métrica, se definió con el fin de definir un mínimo nivel de calidad a la hora de realizar la liberación de la versión Alfa del producto.

El equipo decidió internamente, como condición necesaria para realizar la liberación, que no podría contar con fallas de nivel de criticidad medio ni alto. Es decir, en caso de que no se pudieran reparar todas las fallas para realizar la liberación, los incidentes registrados debían tener nivel de prioridad bajo.

- Criticidad alta: la falla genera que el sistema finalice abruptamente, sin permitir continuar con la utilización de la herramienta.
- Criticidad media: la falla no permite finalizar el flujo de determinada funcionalidad, el sistema continúa activo, pero no se logra cumplir con el objetivo de ese flujo.
- Criticidad baja: falla en la presentación de la información o en la estética de la interfaz de usuario.

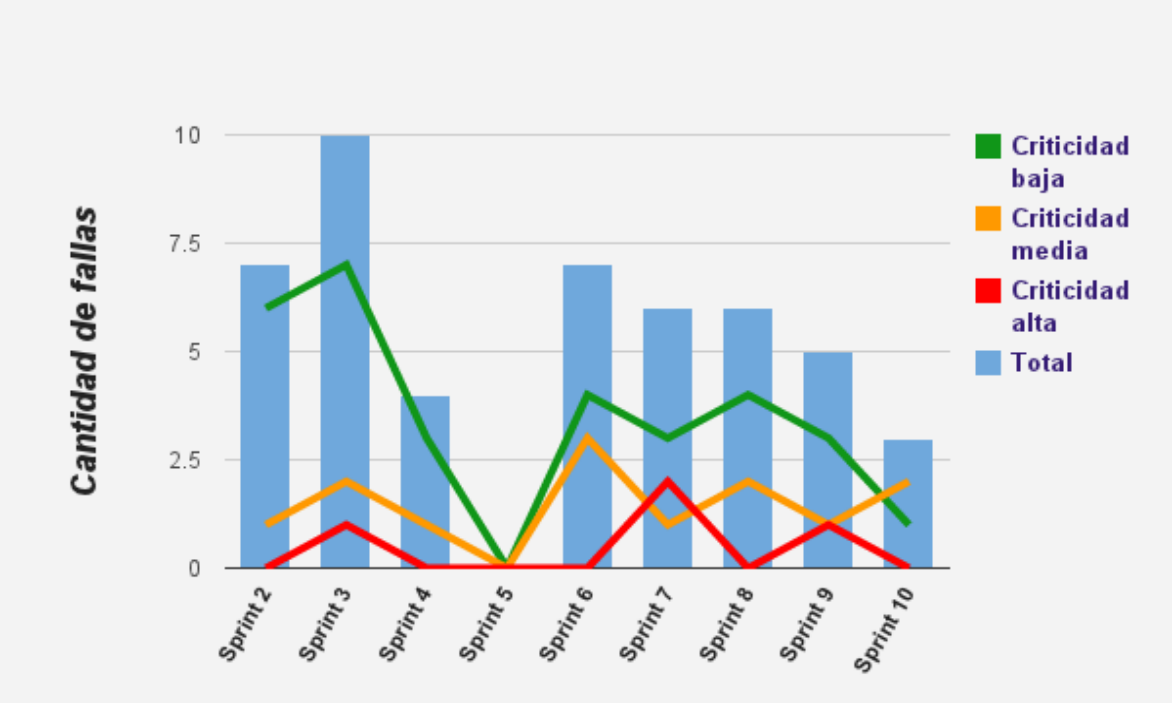


Figura 9.3 - Cantidad de fallas por nivel de criticidad, por *sprint*, con respecto al total de encontrados.

9.7.3. Porcentaje de fallas reparadas

Siguiendo con la estrategia definida para la métrica anterior, definimos esta métrica para monitorear o definir cuando era necesario destinar esfuerzo a la creación de nuevos *user stories* en lugar de la reparación de fallas. Normalmente, cuando las fallas tenían criticidad media o alta se tendía a priorizar la reparación.

Para el mantenimiento de esta métrica, era necesario también, tener definidas tanto la cantidad total de fallas acumuladas en los *sprints*, como el total acumulado de fallas reparadas.

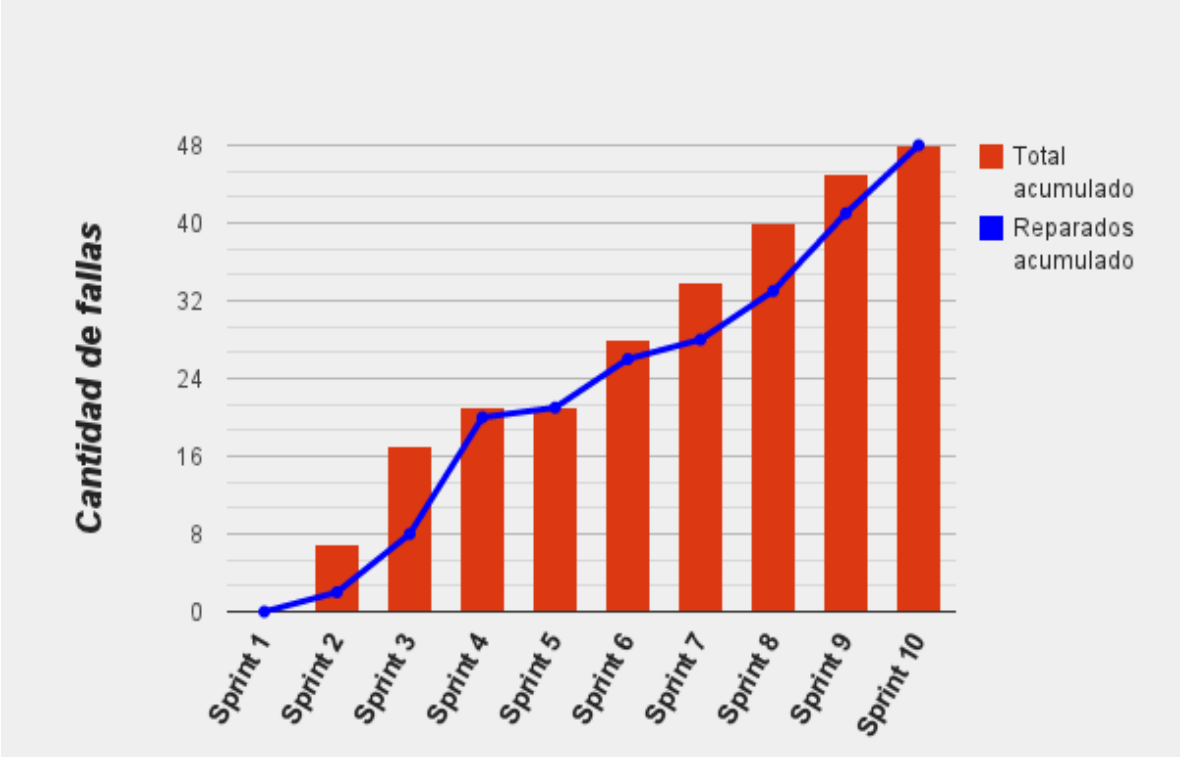


Figura 9.4 - Cantidad de fallas reparadas por *sprint,* respecto al registro total de fallas acumuladas.

10. Gestión de la Configuración del Software

10.1. Introducción

Cuando se trabaja con un equipo en el cual los miembros estás constantemente utilizando recursos compartidos, se deben establecer políticas de acceso y/o reglas para no sobrescribir el trabajo de otros miembros. Todo el equipo debe ser consciente de esto en todo momento, ya que siempre puede haber alguien trabajando en el mismo recurso. Por lo tanto el comienzo de la gestión de la configuración comienza con la concientización del equipo; una vez adquirida comienzan las actividades que se presentan a continuación.

10.2. Elementos de la configuración de *software* (ECS)

Según la naturaleza del proyecto a realizar es que surgen los tipos de elementos a utilizar para su desarrollo. Estos se denominan elementos de la configuración de software (ECS).

El proyecto realizado contiene dos grupos de ECS, a continuación se detallan los mismos :

- **Código fuente**. Son todos aquellos archivos generados para la ejecución del sistema (base de datos, aplicación web y móvil).
- **Documentos**. Son todos aquellos archivos destinados para la gestión de las áreas de conocimiento. Los mismos son :
    - Planes
    - ESRE
    - *Product backlog*
    - Diseño de arquitectura
    - Estándares
    - Capítulos y documento final del proyecto
    - Presentaciones para la revisión
    - Otros documentos. Aquí entrarían imágenes, planillas de cálculos,

archivos de texto, grabaciones de audio (ej. con cliente, tutor, docentes expertos y revisores).

10.3. Selección del Repositorio

10.3.1. Repositorio de Código, Base de Datos y Diseño

A Moon Ideas le correspondía tener el código fuente de la aplicación desarrollada. Por eso el primer día propuso utilizar su servidor local y gestionar los ECS utilizando la herramienta de versionado que se utiliza actualmente en la empresa, SVN.

Del lado del servidor se utilizó la herramienta VisualSVN Server (instalado en *Windows* OS). Del lado de las máquinas cliente se utilizaron dos herramientas, un *plugin* propio de la IDE Eclipse (*Windows* OS) llamado Subclipse y la herramienta Versions (Mac OS).

Existió una problemática más; el servidor de Moon Ideas no contaba con una *ip* fija. Para solucionar esto el equipo utilizó los servicios brindados por la página NoIp la cual permitía acceder de forma remota al repositorio del proyecto. Fue necesaria la instalación del *Dynamic Update Client* (*DUC*) en el servidor y dejarla siempre activa para su uso en todo momento.

Moon Ideas también proveyó parte del diseño para la aplicación móvil y el *front* de la página web. Los mismos también fueron almacenados en este repositorio, dejando allí únicamente los recursos relacionados a Moon Ideas.

10.3.1. Repositorio de Documentos

El equipo optó por utilizar un repositorio diferente para los documentos: Google Docs. Antes de la selección definitiva se evaluaron los beneficios frente a otras herramientas como por ejemplo SVN y Dropbox. Las características más importantes que definieron la elección de la herramienta fueron la rapidez de actualización de la información y la capacidad de poder ver en tiempo real quien estaba editando el documento.

10.4. Estructura del Repositorio

Dado que se utilizaron dos repositorios se detallarán ambas estructuras. Primero se presentará la estructura del repositorio del código fuente. La misma está constituida

por una carpeta raíz llamada “MoonDirector” y cuatro sub carpetas, una para el desarrollo de iOS, otra para Android, otra para el servidor y una última para el diseño.

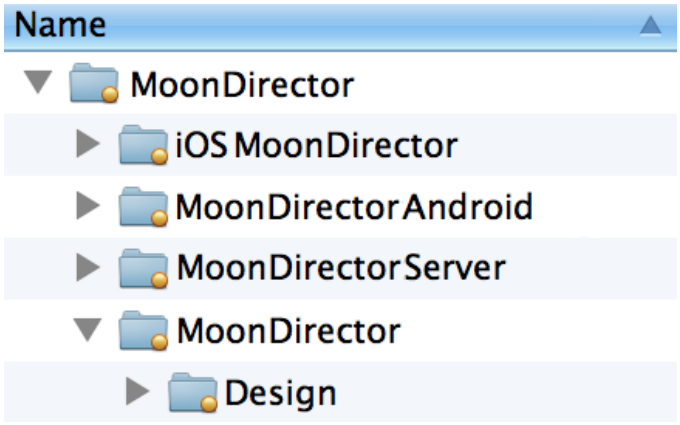


Figura 10.1 – Estructura del Repositorio de código fuente

El repositorio de documentos está formado por una carpeta raíz llamada “Proyecto Final” y 10 subcarpetas. La mayoría de las subcarpetas corresponden a las etapas del proceso de Ingeniería de *software* ya que fue necesario crear documentos para su gestión. Las restantes carpetas contienen material específico como por ejemplo: presentaciones de la revisión o información relacionada a los *sprints*.

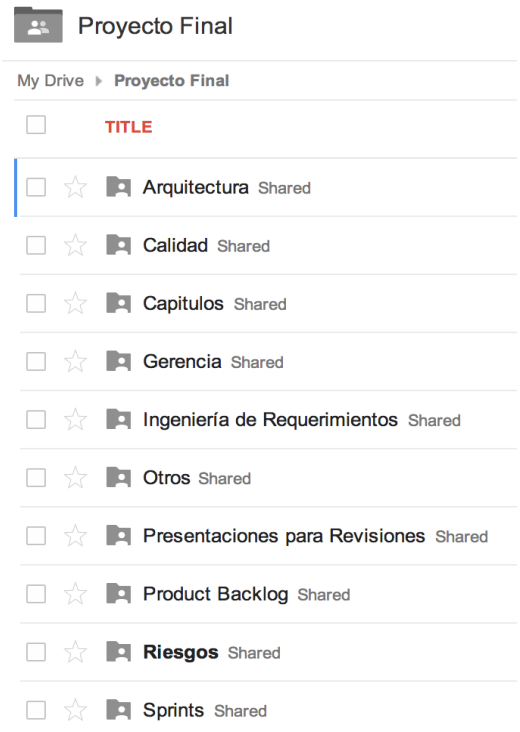


Figura 10.2 – Estructura del repositorio de documentos

10.5. Estrategias de *branching*

El equipo consideró tres estrategias de *branching* debido a las características del proyecto. Las dos más utilizadas fueron *branching* por “tarea” y por “tecnología”. La tercera, utilizada una única vez, se denomina de “promoción de código”.

Branching **por tarea**

Para evitar sobrescribir tareas realizadas por otras personas y disminuir la productividad, el equipo adoptó este tipo de *branching*. Fueron útiles para tareas breves, las cuales era necesario resolver rápidamente. A modo de ejemplo, la siguiente imagen muestra la realización de dos tareas y como se separan en dos ramas.

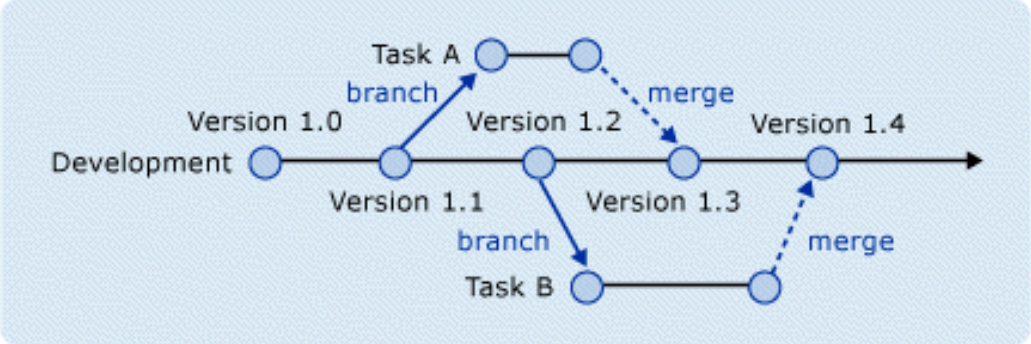


Figura 10.3 – Branching por tarea

Branching **por tecnología**

Esta estrategia fue útil ya que se programó para iOS, Android y un servidor común. Cada una de estas ramas podrían haber generado a su vez ramas por tarea. En la rama de iOS y Android no sucedió ya que había un programador por área, pero para la rama común donde sí habían varios programadores si sucedió.

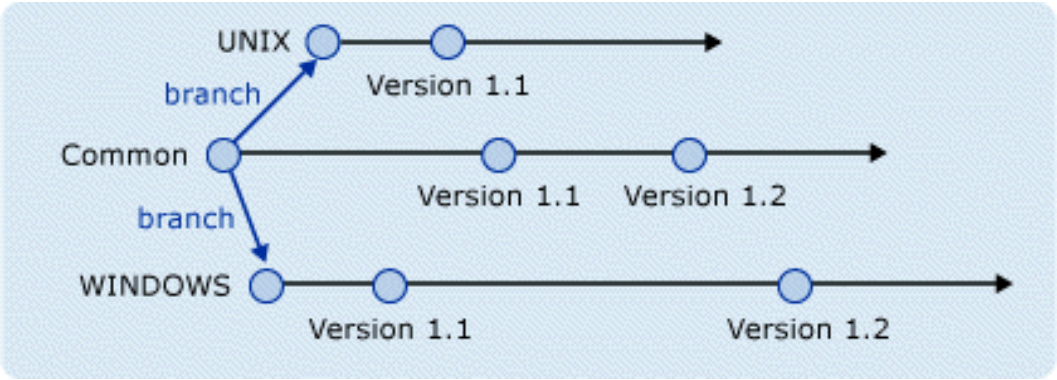


Figura 10.3 – *Branching* por tecnología

Branching por promoción de código

La tercer estrategia utilizada fue *branching* por promoción de código, la cual permitió ramificar versiones de todo el código fuente. Primero se separó a una rama de *testing* para realizar las pruebas necesarias. Luego, una vez terminadas las pruebas se separó a una rama de producción para ser desplegado.

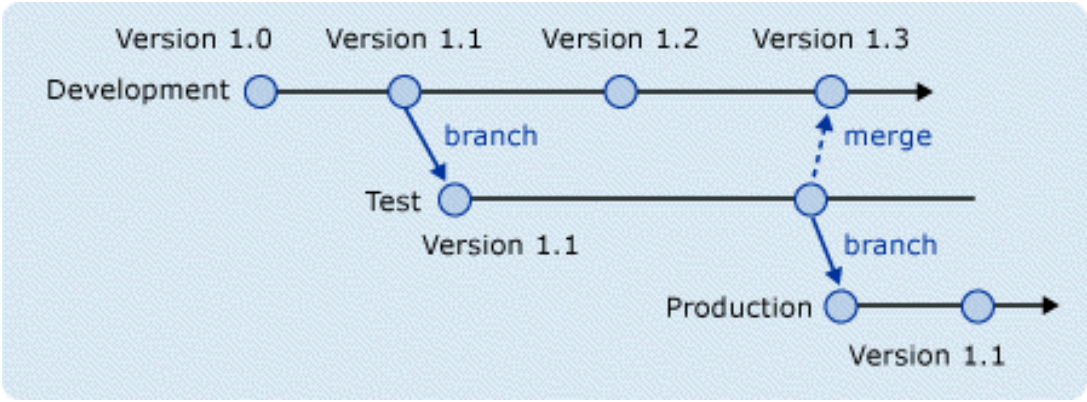


Figura 10.3 – *Branching* por promoción de código

10.6. Circuito de control de cambios

Como era predecible surgieron mejoras, cambios y errores en la construcción del producto. Para gestionarlos se siguió el siguiente circuito. El mismo se disparó cuando surgía algún cambio.

- **Área de SCM analiza el cambio.** El encargado de SCM reúne al equipo para analizar el cambio solicitado y decidir si el mismo está dentro de los parámetros del proyecto.

- **Equipo de desarrollo evalúa la viabilidad de la implementación.** Una vez aprobado por el área de SCM los desarrolladores evalúan si es posible desarrollar el cambio en términos de recursos y *story points*.

- **Agregarlo al *product backlog*.** Se prioriza la *user story* que contiene el cambio y se agrega al *product backlog*.

- **Agregarlo al *sprint backlog*.** Se agrega la *user story* en el sprint correspondiente.

- **Desarrollar la *user story* con el cambio.** El equipo de desarrollo construye la *user story* con el cambio incluido.

- ***Testear* lo implementado.** Se lleva a cabo el proceso de *testing* .

- **Actualizar el repositorio.** Por último, se actualiza el repositorio para generar una nueva versión estable del código y finaliza el circuito de control del cambios.

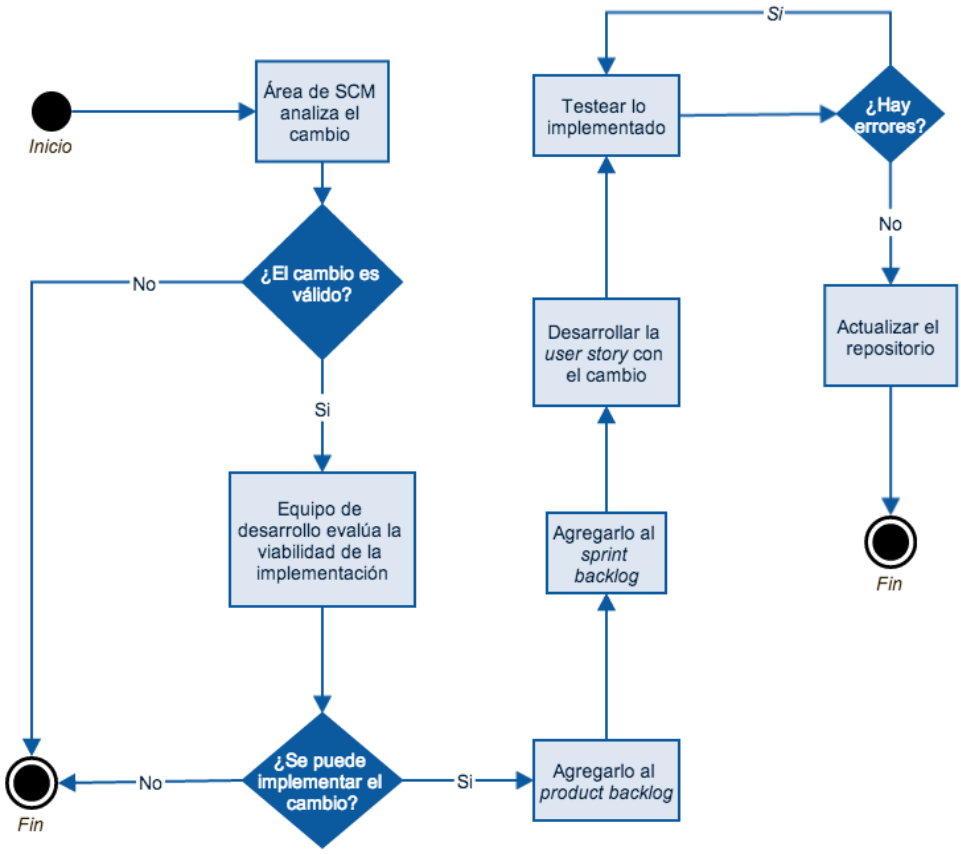


Figura 10.4 – Circuito de control de cambios

11. Gerencia

11.1. Introducción

En este capítulo se busca informar y describir las estrategias y actividades realizadas a lo largo del proyecto para gestionar el calendario y las tareas, medir el avance, realizar seguimiento de riesgos y administrar recursos, con el objetivo de lograr un producto de la mejor calidad posible.

11.2. Metodología de trabajo

Como se mencionó anteriormente, se decidió utilizar para el proyecto una metodología de trabajo específica, ágil y con características de *scrum*. Dicha elección, se basó en la necesidad de cumplir con las expectativas del cliente, respecto al avance del producto, así como en la obtención de resultados tempranos y satisfactorios.

Para cumplir con estos requisitos, era de suma importancia que los integrantes del equipo mantuvieran una comunicación fluida y adoptaran una forma de trabajo flexible. Más específicamente, que éstos estuvieran capacitados para realizar múltiples tareas y desempeñarse en diversos roles, ya que de esta manera podrían adaptarse con mayor rapidez a los cambios que surgieran.

Por lo tanto, se consideró que dicha metodología era la más adecuada para el proyecto, ya que se aplicaba a las características particulares tanto del cliente como del equipo. Con respecto al cliente, se tomó en cuenta su alta disponibilidad y su interés por participar activamente en el desarrollo del producto. Por otro lado, en lo que respecta al equipo, la coordinación de reuniones se podía llevar a cabo con mayor facilidad, ya que se trataba de un grupo reducido, donde sus integrantes residían en viviendas cercanas.

11.2.1. Etapas del proceso

La primera etapa del proceso de gestión comenzó antes de la fase de construcción del producto. Se extendió desde que la cátedra le asignó un tutor al equipo, hasta la revisión del primer *sprint* de puesta a punto que realizó el equipo. En este *sprint,* se

llevó a cabo la definición y especificación de los diferentes planes y procesos que luego, se iban a utilizar como guía durante el proyecto. Los documentos obtenidos como resultado de esa iteración, se encuentran listados como resultado del *sprint* puesta a punto (ver Anexo 16: Evolución de los *sprints)*.

A partir de allí, comenzó una segunda etapa de la gestión, que se llevó a cabo durante el proceso de construcción del producto. Por ese motivo ésta fue denominada “fase de construcción”.

Esta etapa, fue la más importante desde el punto de vista de la gestión, ya que fue la más extensa en cuánto a duración, más compleja en cuanto al impacto de las decisiones y donde se llevó a cabo la metodología explicada en la sección anterior. En la primera etapa, las decisiones se tomaban en base a lo que podía suceder, mientras que en ésta, debían tomarse decisiones en base a lo que sucedía en el proceso de construcción y para evitar o mitigar situaciones no deseadas durante el mismo.

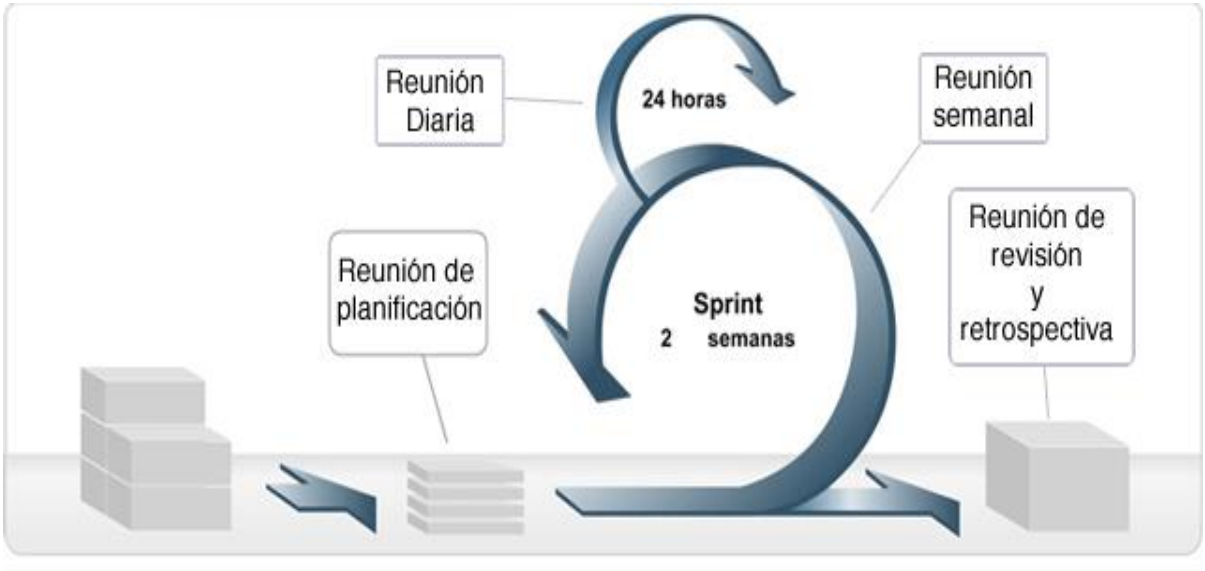


Figura 11.1 - Diferentes instancias donde se desarrollaban actividades de gestión.

Más adelante, cuando se expliquen las fases identificadas en el proyecto, se mencionará que durante la fase de construcción del producto se realizaron algunas modificaciones en la forma de trabajo. Por ese motivo, para cada una de las actividades de gestión que se mencionan a continuación, si corresponde, se explicará cómo se trabajó antes y después de realizar dichas modificaciones.

11.2.2. Planificación del sprint

Cada sprint comenzaba con una reunión con todos los integrantes del equipo donde se definía cuales *user stories* iban a conformar el *sprint backlog* que se implementaría durante las siguientes dos semanas.

Para dicha selección se tenían en cuenta varios aspectos. Primero se evaluaba la prioridad asignada a las *user stories* según lo relevado con el p*roduct owner*.

En segundo lugar, se evaluaba si había *user stories* sin terminar dentro del *product backlog* que precedían a las *user stories* correspondientes al sprint actual. Por último, si había defectos pendientes de reparación encontrados en iteraciones anteriores y dependiendo del nivel de importancia que tenían y la prioridad del *user story* del que provenían, se podían incluir en el sprint.

Luego de la selección, se realizaba una lista de tareas necesarias para la construcción de cada *user story* y se estimaba el esfuerzo de cada tarea mediante la asignación de un valor en *story points*. El número seleccionado como valor de estimación para cada tarea, se obtenía de la secuencia de Fibonacci y como resultado de un p*lanning póker* [3] entre los integrantes del equipo. Si la suma de la estimación de las tareas de un *user story* superaba la estimación previa (realizada en la construcción del *product backlog*), se ajustaba el valor estimado para esa *user story*. Cada *user story* podía tener asignado un responsable, o no, lo mismo sucedía para las tareas asignadas a la misma. Si luego de comenzada una iteración, una tarea no tenía un responsable asignado, cualquier integrante del equipo que fuera quedando liberado de tareas se la auto asignaba y la resolvía.

A continuación, se creaba un nuevo *sprint backlog* en la herramienta de gestión y allí se ingresaban las *user stories* seleccionadas. Estas *user stories* ya se encontraban pre cargadas en el *product backlog* de la herramienta, con lo cual solo había que indicar que *user stories* debían cargarse al sprint. Luego, para cada una de las *user stories* se cargaban sus tareas con su respectiva información. Cuando comenzaba el *sprint*, y en la medida que se iban realizando las tareas, se debían registrar los cambios de estado de las mismas.

Por último, se enviaba un correo electrónico al *product owner* donde se le informaba sobre las *users stories* a desarrollar y se buscaba clarificar dudas o ambigüedades que se habían generado en la especificación del *product backlog*. Éste, respondía el correo con comentarios correspondientes a las dudas, y normalmente marcaba temas pendientes de especificación que luego se trataban en la reunión semanal (ver Anexo 18: Evidencia de comunicación con *product owner*).

11.2.3. Seguimiento del sprint

El seguimiento de las iteraciones, se realizó de dos formas durante la fase de construcción. Al principio, el equipo se reunía día por medio o cada dos días y además se mantenía en contacto diariamente vía telefónica o por correo electrónico.

El objetivo era comunicar el estado en que se encontraban las tareas que cada uno tenía asignado, comunicando que había hecho el día anterior, que pensaba hacer ese día y si había tenido problemas para resolver algo, como pensaba solucionarlo.

En caso de que alguno de los integrantes tuviera problemas para resolver una tarea, lo comunicaba al resto, si alguno de los demás integrantes pensaba que podía resolverla, intercambiaba tareas con quien no podía resolver su tarea. En otro caso, si el problema era complejo o simplemente a nadie se le ocurría una solución, se discutía en la siguiente reunión y se buscaba una solución en conjunto.

En la segunda parte de la fase de construcción, el equipo se reunía todos los días, exceptuando los domingos. Las reuniones diarias no tardaban más de veinte minutos, y al igual que en la primera parte de la fase de construcción, el objetivo era realizar un seguimiento del estado de las tareas del sprint. Si alguno de los integrantes del equipo tenía inconvenientes para resolver alguna tarea, se discutía la solución o intercambiaba tareas con alguno que supiera como resolverla.

11.2.4. Reunión semanal

Esta ceremonia, es una de las adaptaciones que el equipo debió realizar para optimizar el funcionamiento de su metodología, dado que el *product owner* no cumplía horarios regulares en su oficina. Por este motivo, era complejo coordinar una reunión con él, que coincidiera con la fecha de la revisión.

Como solución, sugirió que el equipo continuara con su operativa sin incluirlo, mientras que él se comprometería a mantenerse actualizado. Las actualizaciones, se llevaban a cabo de manera informal durante el horario de oficina de su empresa, donde trabajaban dos integrantes del equipo.

De esa forma, un día aleatorio de la semana y en cualquier intervalo laboral de quince a treinta minutos, él se reunía con cualquiera de los integrantes del equipo o con ambos. Éstos le informaban sobre el estado de las funcionalidades que se estaban desarrollando, o también preguntaban dudas específicas sobre el producto. En caso de contar con funcionalidades concluidas, se le mostraban para que pudiera probarlas, realizar comentarios sobre las mismas y devolver una opinión al respecto.

Por lo tanto, la implementación de estas reuniones semanales suplía las instancias de planificación de *sprints* y validación de resultados, que normalmente según s*crum*, se llevarían a cabo en las reuniones de planificación y revisión respectivamente. De esta manera, las revisiones se convirtieron en una mera formalidad para brindar un marco de cierre y análisis a los *sprints*.

11.2.5. Revisión del sprint

Las revisiones se realizaron de dos formas: al principio de la fase de construcción, la revisión se llevaba a cabo al finalizar el último día del *sprint* y en los días siguientes se realizaba la retrospectiva. Luego de algunos *sprints* esta secuencia se modificó, y se comenzó a realizar la revisión y la retrospectiva en un mismo día.

La reunión de revisión del *sprint*, tenía como objetivo revisar lo implementado y obtener las métricas definidas, algunas al principio y otras en el transcurso del proyecto. En la primera parte de la fase de construcción, estas reuniones se llevaban a cabo en la empresa cliente, incluían al equipo y al *product owner*. Éste tenía libertad para opinar, realizar críticas constructivas que considerara oportunas y alertar al equipo sobre posibles riegos o planes de acción para éstos.

En la segunda parte, las reuniones incluían únicamente a los integrantes del equipo y en ellas se llevaban a cabo las mismas tareas. Por último, cabe destacar que las validaciones sobre las funcionalidades desarrolladas en el *sprint,* se llevaban a cabo en las reuniones semanales con el cliente.

11.2.6. Retrospectiva del sprint

Al igual que la revisión, la forma en que se llevaba a cabo la retrospectiva se modificó al momento de cambiar la forma de trabajo en la fase de construcción. Al principio, esta ceremonia de *scrum* requería más de una reunión, por lo tanto, como mínimo se tardaba tres días en finalizar. Se realizaba recopilación de métricas, organización de datos, análisis, conclusiones y toma de decisiones respecto a los

mismos. El retraso en el trabajo se debía a la inexperiencia en la metodología y la falta de consciencia de los objetivos y las tareas involucradas en estas reuniones.

En la segunda parte, eso cambió y se decidió realizar la retrospectiva del *sprint* conjuntamente con la revisión, tratando de finalizarla en un día o como máximo en dos pero incrementando el número de horas de la reunión.

Básicamente, se buscaba ordenar las métricas obtenidas y analizarlas. A partir de ese primer análisis, teniendo en cuenta los objetivos planteados para la iteración, se analizaba si había existido re-trabajo, contratiempos con alguna actividad o inconvenientes para el desarrollo de las tareas del *sprint*.

Con dicha información se analizaban posibles mejoras o aspectos a tener en cuenta para las próximas iteraciones. Por último, se analizaban los riesgos y se determinaba si había que incluir nuevos, y si los planes de acción eran adecuados o era necesario modificarlos.

11.3. Roles

11.3.1. *Product owner*

El *product owner* es el representante del cliente para el proyecto. Con él se realizó el relevamiento de requerimientos, se priorizaron las *user stories*, se realizó la validación de lo implementado para cada *sprint*, así como también la validación final de la primera versión del producto, que fue el resultado del proyecto. En este caso, su nombre es Matías Burgos y es socio de Moon Ideas, empresa que creó el ElSuperDT.com.

11.3.2. *Scrum master*

Este rol no se utilizó dentro de la metodología desarrollada. El equipo trató de compensar esta falta mediante la colaboración de todos los integrantes. Cada uno de ellos, aportaba tratando de apegarse lo máximo posible a la metodología y corrigiendo a sus compañeros en caso de observar que algo no se realizaba como estaba establecido.

11.3.3. Equipo de desarrollo

El equipo estuvo compuesto por tres integrantes, todos ellos realizaron tareas de relevamiento, análisis, diseño, implementación, pruebas, documentación y gestión, por lo cual debieron desarrollar múltiples habilidades. Además, todos en conjunto fueron responsables de que se cumplieran los tiempos de entrega del producto acordados para el proyecto. Los miembros del equipo fueron: Rafael Etcheverry, José Luis Obes y Diego Lussich.

11.4. Adaptación de *scrum*

Scrum Puro		Scrum Implementado
Conceptos Generales	Construye el producto de forma incremental a través de iteraciones breves.	Si
	Orientado al cliente	Si
	Las iteraciones (en *scrum* llamadas *sprints*) se repiten de forma continua hasta que el cliente da por terminado el producto.	En la primera parte de la fase de construcción los *sprints* eran planificados según la agenda de los participantes, eso hacía que no fueran continuos sino esporádicos. Luego se modificó y se realizaron de forma continua. Las iteraciones finalizan formalmente en una fecha convenida con el cliente, de acuerdo a la fecha de entrega del proyecto.
	Se comienza con la visión general del producto, especificando y dando detalle a las funcionalidades o partes que tienen mayor prioridad de desarrollo y que pueden llevarse a cabo en un periodo de tiempo breve (según los casos pueden tener duraciones desde una semana hasta no más de un mes).	Si, el orden en que se desarrollaron los módulos se realizó por precedencia y/o por la prioridad asignada por el *product owner*.

	Cada uno de los *sprints* finaliza con la entrega de una parte(incremental) del producto	No se realizaban entregas parciales en cada *sprint*, las instancias de validación del producto con el cliente se daba cuando se finalizaba cada módulo. En los *sprints* se realizaban revisiones por parte del equipo, asegurando que se cumplieran los criterios de aceptación. Las validaciones sobre las funcionalidades desarrolladas en cada iteración se realizaban de manera informal a lo largo de las mismas, ya que el *product owner* no cumplía horarios regulares de oficina. Cuando coincidía con alguno de los integrantes del equipo que trabajaban en su empresa, solicitaba información sobre el estado del proyecto y realizaba pruebas informales sobre las nuevas funcionalidades finalizadas.
	Colaboración de cada integrante del equipo de forma abierta con los demás, según sus capacidades y no según su rol o su puesto.	Si
	Sprints de hasta 1 mes de duración	Los *sprints* duraban dos semanas. Durante los primeros tres *sprints*, la semana posterior se continuaba trabajando en las fechas que no se trabajaba de forma conjunta con el resto del equipo.
	Elementos: • *Product backlog* • *Sprint backlog* • Incremento	Los incrementos no coinciden con entregas al cliente, no hay relación entre el número de *sprints* con el número de entregas.
	Priorización del trabajo	Si
	Tener un "*product owner*": persona que toma las decisiones del cliente	Si

	Se recomienda un tamaño de equipo entre 4 y 8 personas	Nuestro equipo contó con tres integrantes.
	En los proyectos predictivos, los requisitos del sistema suelen especificarse en documentos formales; mientras que en los proyectos ágiles toman la forma de *product backlog* o de *user stories*.	Si
	En *scrum*, la visión del cliente es conocida por todo el equipo (“el cliente forma parte del equipo”) y el *product backlog* se realiza y evoluciona de forma continua con las aportaciones de todo el equipo.	Si
Diseño y Arquitectura	Durante el desarrollo se genera el diseño y la arquitectura final de forma evolutiva.	Definimos una arquitectura antes de comenzar con la implementación, luego del quinto *sprint* se realizaron algunas modificaciones, por lo cual consideramos que se realizó de forma evolutiva. Por su parte, el diseño también fue evolucionando en la medida que transcurrieron los *sprints*.

Product Owner	• Debe conocer perfectamente el entorno de negocio del cliente, las necesidades y el objetivo que se persigue con el sistema que se está construyendo • Tener atribuciones suficientes para tomar las decisiones necesarias durante el desarrollo. • Conocer *scrum* para realizar con solvencia las tareas que le corresponden. • Desarrollo y administración del *product backlog*. • Presentación y participación en la reunión de planificación de cada *sprint*. • Recibir y analizar de forma continua retro-información del negocio (evolución del mercado, competencia, alternativas) y del proyecto (sugerencias del equipo, alternativas técnicas, pruebas y evaluación de cada incremento).	• Tiene atribuciones suficientes • El *product owner* conocía *scrum*, de cualquier forma acordamos esta forma de trabajo y estuvo de acuerdo. • El *product backlog* lo creamos en conjunto con él, se definieron los módulos, las funcionalidades que debía incluir cada módulo y la prioridad de cada *user story* incluido. • No participaba en la reunión de planificación, salvo casos excepcionales. Sí existía alguna duda respecto a la selección de user stories, ante igualdad de prioridades, se consultaba con el *product owner.* • Validación del producto luego de la finalización de cada módulo.
Equipo	Es un equipo multidisciplinar, en el que todos trabajan de forma conjunta para realizar el trabajo de cada *sprint*	Si, se delegó la responsabilidad de algunas áreas a cada uno, pero todos realizan tareas de todo tipo.
	El equipo tiene un propósito común: conseguir el mayor valor posible para la visión del cliente.	Si
	Auto organización: En *scrum* los equipos son auto-organizados, con margen de decisión suficiente para tomar las decisiones que consideren oportunas.	Si

	• Todos los miembros conocen y comprenden la visión del propietario del producto. • Aportan y colaboran con el propietario del producto en el desarrollo del *product backlog.* • Comparten de forma conjunta el objetivo de cada *sprint* y la responsabilidad del logro. • Todos los miembros participan en las decisiones. • Se respetan las opiniones y aportes de todos • Todos conocen el modelo de trabajo con *scrum*.	• Si • Si • Si • Si • Si • Ninguno de los integrantes tenía experiencia en *scrum*. Eso se reflejó en los primeros *sprints* del proyecto.

Scrum Master	• Asesoría y formación al Propietario del producto. • Asesoría y formación al equipo. • Revisión y validación de la pila del producto. • Moderación de las reuniones. • Resolución de impedimentos que en el *sprint* pueden entorpecer la ejecución de las tareas. • Gestión de la "dinámica de grupo" en el equipo • Respeto de la organización y los implicados, con las pautas de tiempos y formas de *scrum* • Configuración, diseño y mejora continua de las prácticas de *scrum* en la organización.	No se utilizó este rol. Por lo tanto todos los integrantes del equipo se esforzaban por apegarse a la metodología de trabajo, recordarle a sus compañeros sobre las distintas ceremonias o inclusive corregirlos, si consideraban que no estaban adoptando la metodología para realizar una tarea.
Product Backlog	<u>Requisitos del Sistema :</u> • Las funcionalidades que incluye dan forma a una visión del producto definida y conocida por todo el equipo. • Las funcionalidades están individualmente definidas, priorizadas y pre-estimadas. • Están realizados y gestionados por el cliente (*product owner*)	Si

Sprint Backlog	<u>Requisitos del software:</u> • Incluyen todas las tareas necesarias para construir la parte de producto de un *Sprint*. • El equipo ha estimado el esfuerzo de cada tarea. • El equipo ha asignado cada tarea a un miembro. • Cada tarea del *sprint backlog* tiene asignada una persona y el tiempo que aún falta para terminarla	Sí, el esfuerzo de las tareas estaba estimado en *story points*.
Tareas	• División en tareas realizada de forma conjunta por todos los miembros del equipo. • Realizar todas las tareas identificadas por el equipo para conseguir el objetivo del *sprint*. • El tamaño de cada tarea está en un rango de 2 a 16 horas de trabajo. • Es visible para todo el equipo. Idealmente en una pizarra o pared en el mismo espacio físico donde trabaja el equipo.	• SI • SI • Esta precaución se comenzó a tener en cuenta luego del quinto *sprint*. • Era visible para todo el equipo en la herramienta de gestión
	El *sprint backlog*, es la lista que descompone las funcionalidades del *product backlog* en las tareas necesarias para construir un incremento	Si
Incremento	Se realiza un incremento en cada iteración, parte del producto realizada en un *sprint*, terminado y probado	Si

Planning Meetings	Jornada en la cual se define el trabajo y los objetivos que se deben cumplir en esa iteración. Esta reunión consiste en dos partes: En la primera, que puede tener una duración de una a cuatro horas, se decide qué elementos de la pila del producto se van a desarrollar. En la segunda, se desglosan éstos para determinar las tareas necesarias, estimar el esfuerzo que necesita cada una y asignarlas a las personas del equipo. La planificación del *sprint* no debe durar más de un día.	La planificación del *sprint* consistía en determinar qué *user stories* del *product backlog* se iban a desarrollar. Cada *user story* se desglosaba para definir las tareas que requería. No duraba más de 4 horas.
	Los miembros del equipo se auto asignan las diferentes tareas tomando como criterios sus conocimientos, intereses y distribución homogénea del trabajo.	Si
	La reunión es un trabajo de colaboración activa entre los dos protagonistas: cliente y equipo, donde concluye con un acuerdo sobre el incremento de producto que van a realizar en el *sprint*.	Si
Daily Meetings	*Scrum* gestiona su evolución en reuniones breves diarias donde todo el equipo revisa el trabajo realizado el día anterior y el previsto para el siguiente.	En la primera parte del proyecto las reuniones eran cada dos días. A partir del cuarto *sprint* las reuniones comenzaron a realizarse diariamente ya que comenzamos a trabajar todos juntos en un lugar y horario definido.
	En el *sprint* se pone el objetivo establecido, duración, funcionalidades del *product backlog* comprometidas, hora fijada para las reuniones diarias y fecha prevista para la reunión de revisión del *sprint*.	Si

	Resultados de la *daily meeting*: *sprint backlog* y gráfico de avance (*burn-down*) actualizados. Identificación de necesidades e impedimentos. Se marca el trabajo que queda pendiente a las tareas que tiene asignadas, o marca como finalizadas las ya completadas	Cada integrante indicaba en que estaba trabajando, que problemas tenía, como pensaba solucionarlos y siempre se encontraba abierto a sugerencias. En caso de no saber cómo resolver un problema, podía intercambiar tareas con otro integrante que supiera cómo resolverla. Luego de finalizar cada tarea el responsable debía actualizar su estado en la herramienta.
Revisión del *sprint*	Al finalizar cada iteración (*sprint*) se lleva a cabo una revisión con todas las personas implicadas en el proyecto Al ver y probar el incremento, el propietario del producto, y el equipo en general obtienen *feedback* clave para evolucionar y dar más valor al *product backlog.* El equipo hace una introducción general del *sprint* y demuestra el funcionamiento de las partes construidas. Se abre una instancia de preguntas y sugerencias sobre lo visto.	El *product owner* no formaba parte de estas reuniones, las validaciones se realizan en aquellos incrementos que coincidían con la finalización en la construcción de un módulo. Sin embargo existían instancias informales de interacción con el *product owner*, que se utilizaban para mantenerlo al tanto del estado del proyecto y para discutir algún aspecto del sistema, en caso de que fuera necesario. A éstas ceremonias las llamamos Reuniones semanales.

Figura 11.2 – *Scrum* puro [3] vs Metodología implementada

11.5. Organización del proyecto

11.5.1. Fases del proyecto

Consideramos que el proyecto tuvo tres fases marcadas. En primer lugar, una fase a la cual denominamos “puesta a punto”. Esta fase comenzó con el relevamiento de requerimientos para presentar la propuesta a la cátedra *software factory* y se extendió hasta el final del primer *sprint* que realizamos como prueba. Luego, la fase de construcción del producto, se caracterizó por ser un período dividido en iteraciones donde se iban obteniendo avances. Por último, en la fase de documentación fue donde se construyó el documento de entrega del proyecto.

11.5.2. Puesta a punto

El nombre de esta fase es en alusión al nombre del primer *sprint*, el cual se utilizó para tomar contacto o experimentar con ciertas situaciones, como por ejemplo, trabajar junto a un compañeros a los cuales que no se conocía, definir procesos propios, asignar recursos, etc.

Esta fase comenzó antes de la aprobación del proyecto, con una reunión con quien luego fue el *product owner*, para relevar funcionalidades generales del sistema a construir. Una vez que se consiguió un núcleo de requerimientos, se presentó la propuesta. En ese tiempo de evaluación y asignación del tutor se investigó a cerca de las tecnologías, ya que el cliente había solicitado que la aplicación se desarrollara en dos plataformas.

Luego de la aprobación, se llevó a cabo la primera reunión con el tutor, que fue aproximadamente veinte días después de que lo asignaron al equipo. Esto se debió a complicaciones de agenda de cada integrante, además de que se estaba llevando a cabo el período de exámenes. A partir de allí comenzaron las reuniones presenciales semanales con el tutor, con quien también se verificaban contenidos que el equipo iba desarrollando mediante correo electrónico.

Luego se definió la metodología de trabajo, se realizó una planificación general, se convirtió el listado de requerimientos en *product backlog* y se diseñó la primera versión de la arquitectura conjuntamente con el cliente. A partir de ese momento, se

decidió dar comienzo al desarrollo del producto más allá de que existían riesgos, fruto de la inexperiencia, principalmente en materia de gestión. Por este motivo, con el fin de amortiguar el impacto de comenzar a construir el producto sin siquiera haber trabajado juntos, se realizó un *sprint* "tipo" como proceso de adaptación.

En este *sprint*, que se le llamó puesta a punto, se utilizó por primera vez la metodología planificada. Sin embargo, en la planificación de este *sprint* no se estimaron esfuerzos, se decidió que se iba a incluir un conjunto de tareas, que no estaban orientadas directamente al producto pero iban a aportar al proceso de construcción. Este *sprint* tuvo una duración de aproximadamente 3 semanas, donde se realizaron documentos que luego fueron utilizados como guía o simplemente aportaron evidencia para la documentación final. Durante ese período de tiempo se realizó la especificación de un proceso para la Gestión de Requerimientos, la especificación del proceso para llevar adelante las pruebas y la especificación de un proceso general, entre otros. Luego de terminado este *sprint*, el equipo se empezó a conocer, tuvo sus primeras impresiones sobre el funcionamiento, se sacaron algunas conclusiones y se tomaron decisiones acerca de cómo continuar con el trabajo.

11.5.3 - Construcción

Como ya se mencionó, la fase de construcción se dividió en iteraciones. Para cada una de ellas, se realizaban tareas de desarrollo, pruebas, gestión y en algunos casos de documentación.

Antes de comenzar a trabajar en las iteraciones, se realizó una priorización del *product backlog* que se había estimado en la fase de puesta a punto. Esta instancia se volvió a repetir más adelante en el quinto *sprint* del proyecto, cuando se ajustó el alcance del producto y se terminó de completar el *product backlog*. Una vez que se estimó el alcance del proyecto se comenzó a trabajar en la implementación de la solución.

Esta fase tuvo dos partes importantes, desde enero hasta abril, cuando finalizó el cuarto *sprint*, y desde el quinto *sprint* a principios de agosto, cuando se finalizó el décimo.

En la primera parte, las iteraciones no eran consecutivas, es decir, no se realizaban inmediatamente a continuación de la otra, sino que se planificaban basándose en la disponibilidad de los integrantes del equipo en cuanto a su agenda personal. El

equipo trabajaba por separado, en el horario que cada integrante tenía disponible según su horario laboral y se reunía cada dos días para trabajar en conjunto. Además, las reuniones de revisión y retrospectiva tomaban dos o tres reuniones de equipo, lo que equivalía a un mínimo de cinco días. A su vez, quienes no habían terminado sus tareas durante el *sprint*, seguían trabajando en ellas luego de finalizado el mismo en los días en que el equipo no se reunía. Todo esto llevó a que durante los primeros *sprints* el equipo tuviera problemas para llevar a cabo la metodología.

A partir de ahí, antes del *sprint* cinco, se tomaron varias decisiones que tuvieron un impacto positivo en el proyecto y se realizaron cambios que dieron un envión importante en materia de productividad.

El equipo comenzó a reunirse a trabajar todos los días en un lugar específico y en un mismo horario, se realizó gracias al esfuerzo de todos, ya que cada uno tuvo que coordinar el horario laboral. Esto hizo que las reuniones comenzaran a realizarse diariamente, lo cual comenzó a mejorar el funcionamiento del equipo, en cuanto a la utilización de la metodología y la coordinación de las tareas.

En cuanto a la gestión, se modificó la duración de la revisión y la retrospectiva. Comenzaron a realizarse el último día de cada *sprint*, con lo cual, la planificación del siguiente *sprint* se realizaba uno o dos días después. Esto aumentó la frecuencia de *sprints* en el tiempo, lo que permitió realizar un total de diez, finalizando con la construcción del producto el 6 de agosto.

A continuación se muestra un cuadro con las fechas de inicio y finalización de cada *sprint*:

Evento	Duración	Inicio	Fin
Sprint 1	14 días	6/1/2014	20/01/2014
Sprint 2	14 días	27/01/2014	10/2/2014
Sprint 3	14 días	26/02/2014	12/3/2014
Sprint 4	14 días	1/4/2014	15/04/2014
Sprint 5	14 días	6/5/2014	9/5/2014
Sprint 6	14 días	10/5/2014	23/05/2014
Sprint 7	14 días	27/05/2014	10/6/2014

Sprint 8	14 días	11/6/2014	24/6/2014
Sprint 9	14 días	8/7/2014	22/07/2014
Sprint 10	14 días	23/07/2014	6/8/2014

Figura 11.3 - Comienzo y fin de los *sprints* a lo largo del proyecto.

Si bien esta fase del proyecto se identifica como de construcción de producto, cabe destacar que a partir del sexto *sprint* también se comenzaron a realizar tareas orientadas a la documentación final. Esta iniciativa, fue sugerida por el tutor, y no se realizó con el fin de adelantar trabajo de documentación, sino que fue para visualizar aspectos de algunas áreas que le presentaron dificultades al equipo. Es decir, si bien se realizaban tareas de calidad y gestión, no se percibía el valor que estaban aportando. Luego de comenzar a escribir estos capítulos, el equipo se autoevaluó y percibió la importancia de determinadas tareas y a partir de allí hubo una mejoría, se comenzaron a tomar decisiones basadas en números obtenidos en la práctica y no simplemente en sensaciones.

Un ejemplo de estas mejoras puede ser la estimación del esfuerzo para cada *sprint*. En la planificación de los *sprints*, la estimación se realizaba en base a la complejidad de lo que se desarrollaba, a juicio de cada integrante, mediante *planning póker* y teniendo en cuenta la cantidad de horas que cada uno estimaba podía dedicar. Luego de escribir el capítulo de gestión, a partir del *sprint* 6, se comenzó a tener en cuenta el valor de productividad en una hora de proyecto. Con ese valor y la cantidad de horas que se estimó que el equipo iba a dedicar, se estimaba la cantidad de *story points* que se podrían construir en el *sprint*. Luego, hubo otros inconvenientes que llevaron a que la productividad no fuera la esperada, por eso en las dos iteraciones siguientes la estimación no fue exacta. Pero de allí en más, se estimó en base a datos concretos, lo cual permitió que ante una productividad regular la estimación fuera más exacta.

Por último, al final de esta fase, se realizó una evaluación de satisfacción de usuario basada en heurísticas de Nielsen [13], luego de que se llevaran a cabo las pruebas alfa.

11.5.4. Documentación

Las últimas tres semanas del proyecto, se destinaron a realizar la documentación final correspondiente, se complementaron los capítulos que se habían cubierto en la

fase anterior y se realizaron otras tareas orientadas a la finalización del proyecto. También se dedicaron algunas horas al relevamiento y análisis de los resultados de las evaluaciones realizadas en la fase anterior, con el fin de obtener conclusiones sobre el producto entregado.

11.6. Gestión de la comunicación

11.6.1. Comunicación interna del equipo

La comunicación fue un punto de vital importancia para el proyecto, ya que al ser pocos integrantes en el equipo y utilizar metodologías ágiles, debía ser un aspecto fuerte del mismo. A su vez, el hecho de que no todos los miembros del grupo se conocían y jamás habían trabajado juntos, era un desafío que se logró superar, ya que desde el principio hubo buen entendimiento.

Para lograr esto, se utilizaron varias herramientas que no solo permitían la comunicación, sino también establecer y coordinar próximas actividades y reuniones a lo largo de todo el proyecto.

El equipo se comunicaba todos los días con muy pocas excepciones a lo largo del proyecto, como pueden ser las fechas festivas o los días en que alguno se tomó licencia en su trabajo. La comunicación se basaba en reuniones del equipo, como mínimo tres veces por semana, y los días que esto no ocurría, se mantenían en contacto vía correo electrónico, llamadas o mensajería instantánea.

Las herramientas utilizadas para la comunicación y coordinación se mencionan a continuación:

- **Herramienta de gestión y coordinación:** Para la gestión del proyecto en general se utilizó Versión One, que además de asignar las tareas y obtener métricas, permite saber a quién le pertenece determinada tarea, cuando se espera que termine, etc. Es decir, permitía que cada uno de los integrantes pudiera evaluar sus tareas respecto al tiempo estimado para las mismas.

- **Reuniones**: Como ya se mencionó, el equipo se reunía no menos de tres veces a la semana, sin contar las reuniones con el tutor. En estas reuniones, además de chequear el estado de avance de las tareas asignadas, también se planificaban los próximos pasos a seguir. También se analizaban aspectos

de la forma de trabajo, con el fin de ir mejorándolo en la medida que avanzaba el proyecto. Por las características de la metodología seleccionada, las actividades que se llevaban a cabo en las reuniones dependían del tipo de reunión, es decir, variaban dependiendo si era una reunión de planificación, una reunión diaria, una revisión o retrospectiva.

- **Llamadas:** Cuando los integrantes del equipo no se podían reunir por cuestiones de agenda personal, la comunicación se hacía vía teléfono celular. Este tipo de contacto se hacía para que el resto de los integrantes del equipo supieran en que tareas se iba a trabajar ese día, que dedicación se le iba a dar, en qué estado habían quedado el resto de sus tareas asignas y/o lo que le restaba por realizar. De esta manera, todos los miembros del equipo tenían noción del estado de avance de cada *sprint*.

- **Correo electrónico:** Esta herramienta se utilizó internamente para el intercambio de información de tareas específicas, como por ejemplo, luego de una reunión con el tutor, se realizaba una minuta de la reunión donde se detallaban los lineamientos y correcciones realizadas. Esa minuta luego era reenviada a todos los integrantes. El correo, no se utilizaba únicamente en este tipo de instancias, sino que también se utilizaba para notificar información importante cuando el equipo no estaba reunido, o que algún documento había sido compartido en el repositorio de documentos. Por último, en la segunda parte de la fase de construcción fue clave para la comunicación con el cliente, ya que la coordinación de las reuniones se tornó demasiado compleja (ver Anexo 18: Evidencia de comunicación con *product owner*).

- **Google drive:** Se utilizó Google Drive como repositorio para el almacenamiento de contenidos que se iban generando a lo largo del proyecto. Se consideró que podía ser una buena política ir realizando documentos que servirían para la gestión del proyecto, pero también para completar la documentación final más rápidamente. Con ese fin, se compartió entre los integrantes del equipo un directorio en la nube.

11.6.2. Comunicación del equipo con actores externos

En lo que se refiere a las vías de comunicación del equipo con actores externos, ya sea tutor, cliente o futuros consumidores del producto, se utilizaron herramientas similares a las que utilizadas para la comunicación interna. La principal diferencia estaba en la forma de expresión de las conversaciones o de los correos electrónicos.

Para la coordinación de reuniones, normalmente se utilizó el correo electrónico, mensajes de texto o llamadas. A través de estas herramientas, se acordaban fechas y horarios disponibles que más se adaptaran a la agenda del actor externo y del equipo. Una vez que se llegaba a un acuerdo, se fijaban las reuniones mediante Google Calendar, para que oficiara como recordatorio y enviara aviso a todos los participantes de la reunión.

La herramienta que más aportó en cuanto a claridad y entendimiento fue la reuniones presencial. Las reuniones con el tutor se llevaron a cabo casi siempre de forma semanal, salvo algunas excepciones, donde se aplazaron e hicieron una semana después, dando una frecuencia de una reunión cada dos semanas. En cuanto al cliente, salvo alguna excepción, no había reuniones programadas, pero al tener dos integrantes del equipo trabajando en su empresa, hacía que la comunicación fuera muy fluida y siempre estaba abierto a responder dudas o realizar sugerencias.

Por último, pero no menos importante, una herramienta que sirvió para obtener *feedback* o referencias sobre la aceptación del producto que construimos, fueron las evaluaciones de satisfacción. Esta forma de comunicación, permitió hacer un análisis del producto final alcanzado, ya que no solo se pidió *feedback* a potenciales usuarios finales, sino que también al *product owner*.

11.7. Gestión de riesgos

La gestión de los riesgos a lo largo del proyecto consistió en identificar, controlar y eliminar las distintas fuentes de riesgos antes de que éstos se materialicen y puedan incidir en el no cumplimiento de los objetivos del proyecto.

El Plan de riesgos se ejecutó durante las fases de construcción y documentación del proyecto, y fue analizado y monitoreado durante ese período, dándole seguimiento a los riesgos identificados.

Las tareas de seguimiento y control de los riesgos consistieron en monitorear el progreso de los riesgos identificados e identificar nuevos riesgos a lo largo del proyecto.

Dentro de esta sección se explicará la forma en la cual el equipo fue gestionando los riesgos a lo largo del proyecto, la cual se fue corrigiendo a medida que el proyecto avanzaba.

11.7.1. Identificación de riesgos

Al comienzo del proyecto el equipo se reunió para realizar la identificación de los riesgos a tener en cuenta para el proyecto. Mediante técnicas como tormenta de ideas, juicio experto y análisis de riesgos evaluados en proyectos anteriores se logró definir una lista inicial de riesgos.

Luego de definidos los riesgos, el equipo fue *sprint* a *sprint* analizando nuevamente la lista, actualizándola cuando se identificaban nuevos riesgos y re-evaluando las posibilidades de impacto en los objetivos del proyecto.

11.7.2. Análisis cualitativo de los riesgos

Todos los riesgos identificados son clasificados cualitativamente. En este análisis, se determina la probabilidad de ocurrencia del riesgo e impacto sobre los objetivos del proyecto si se materializa.

A continuación se detallan las escalas que se utilizaron para determinar las probabilidades e impactos de los riesgos. Cuanto mayor impacto tenga un riesgo, mayor va a ser el seguimiento y las acciones que se deberán tomar para tratar de mitigar dicho riesgo.

11.7.3. Escalas aplicadas al análisis

Se utilizó una escala de 1 a 5, donde 1 representa el menor impacto posible y 5 sería el valor más crítico en términos orientados al proyecto.

La siguiente tabla representa el significado de cada valor de la escala para el proyecto:

Escala	Descripción
1	Impacto mínimo, no afecta el calendario ni el alcance del proyecto.
2	Impacto moderado, puede causar un retraso leve en el calendario del proyecto
3	Impacto importante, puede causar retrasos de días en el calendario del proyecto, puede afectar el alcance.
4	Impacto crítico, puede causar pérdidas importantes de tiempo. Puede llevar a que el proyecto se detenga.
5	Impacto catastrófico, pérdidas irrecuperables y fracaso del proyecto.

Figura 11.4 - Escala de los diferentes valores de impacto para los Riesgos

Escala para probabilidad de ocurrencia:

Escala	Descripción
0.0	No es un riesgo real
0.1	Muy poco probable
0.2	Poco probable
0.4	Medianamente probable
0.6	Probable
0.8	Muy probable
1.0	No es un riesgo, es un problema

Figura 11.5 - Escala probabilidad de ocurrencia de los Riesgos

Escala de prioridad de riesgos:

Prob\Impacto	1	2	3	4	5
0.1	0.1	0.2	0.3	0.4	0.5
0.2	0.2	0.4	0.6	0.8	1.0
0.4	0.4	0.8	1.2	1.6	2.0
0.6	0.6	1.2	1.8	2.4	3.0
0.8	0.8	1.6	2.4	3.2	4.0
1.0	1.0	2.0	3.0	4.0	5.0

Figura 11.6 - Escala de prioridad de Riesgos

En el cuadro presentado anteriormente, quedan explícitamente delimitadas las franjas de prioridad y atención tenidas en cuenta para los riesgos. Dependiendo del valor de la magnitud de impacto de cada riesgo, la atención y prioridad que se le daría al mismo en cada *sprint*. En el análisis previo a cada *sprint*, y según el valor que tenía cada riesgo en ese momento dentro de la escala, se determinaba cuán riguroso debía ser su seguimiento.

De acuerdo a las franjas explicitadas en el cuadro, la escala quedaba definida de la siguiente manera:

- De 0,1 a 0,9: determinado por la franja de color verde en el cuadro, indica que es un riesgo aceptado.

- De 1,0 a 2,0: determinado por la franja de color anaranjado, indica que es un riesgo considerable, se debía tener en cuenta durante el *sprint*.

- De 2,1 a 5,0: por la franja de color rojo, indica que es un riesgo con el que hay que tener cuidado, realizar seguimiento y control para evitar que se convirtiera en problema.

A continuación, se muestra la evolución de los riesgos considerados como más importantes a lo largo del proyecto:

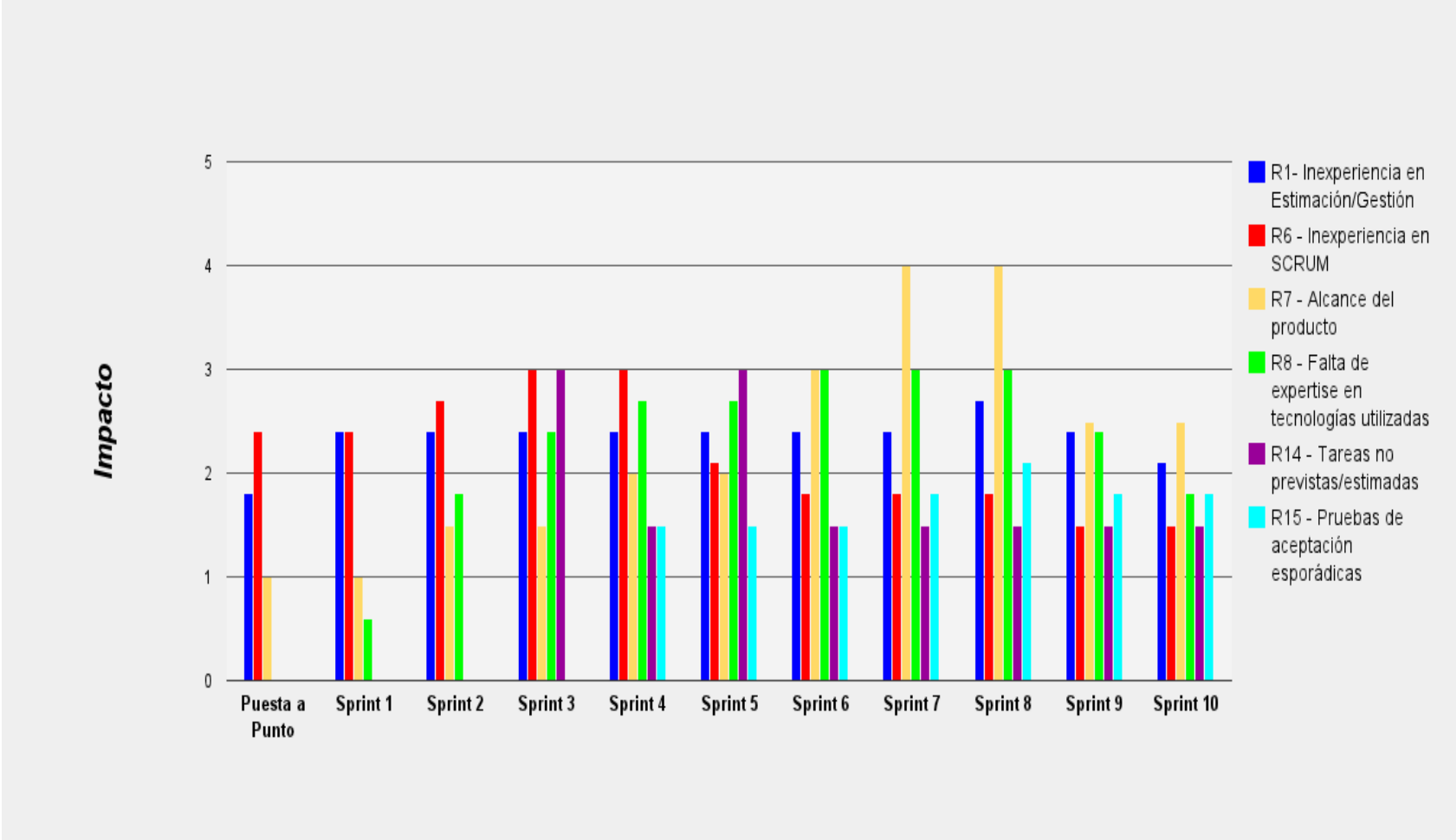


Figura 11.7 - Evolución de los Riesgos a lo largo del proyecto

11.7.4. Riesgos más importantes

A continuación, se destacan riesgos que a lo largo del proyecto fueron especialmente tenidos en cuenta y se les realizó un control y seguimiento más riguroso. En total, se tuvieron en cuenta dieciséis Riesgos en el transcurso del proyecto (ver Anexo 13: Plan de riesgos).

Retrasos en la construcción de requerimientos complejos por falta de conocimientos avanzados en las tecnologías utilizadas.

Fue una dificultad inminente desde el principio del proyecto, ya que se estaba trabajando en dos plataformas simultáneamente, y si bien los desarrolladores con los que contaba el equipo tenían experiencia, no eran expertos en las mismas. Para mitigar este riesgo, se invirtió esfuerzo en investigación y pruebas de concepto, tanto en la fase puesta a punto, como en la de construcción. Dicho esfuerzo, se puede observar en las métricas de horas dedicadas a desarrollo y pruebas, durante la fase de puesta a punto.

Luego, a pesar del esfuerzo destinado a investigación durante la fase de construcción, se volvió un problema durante el desarrollo del módulo de comunicación y hubo que dedicarle un esfuerzo mayor del estimado a las *user stories* involucradas.

Ineficiencia por falta de experiencia en el uso de metodologías ágiles.

Para mitigar este riesgo, durante la primera parte del proyecto previo a comenzar la fase de construcción, se realizó una planificación detallada y consensuada entre todos los integrantes del equipo sobre la forma de trabajo. A pesar de eso, el trabajar de forma distribuida físicamente y en diferentes horarios durante los primeros *sprints* presentó complicaciones. Estas complicaciones no se ven reflejadas en el avance de las primeras tres iteraciones, ya que nuestros desarrolladores tenían experiencia en las tecnologías utilizadas para el desarrollo de dichas funcionalidades. A partir del tercer *sprint*, se comenzó a trabajar con otras tecnologías y se recibieron algunos consejos por parte del tutor, fue entonces cuando se detectó ineficiencia en la forma de trabajo.

A partir de ese momento, se comenzaron a realizar los *sprints* de forma continua, se estableció un lugar fijo y horario de trabajo, lo cual tuvo un impacto positivo en la forma de trabajar, el equipo adoptó la metodología y mejoró su coordinación colectiva.

Desvíos en el cronograma por falta de experiencia en la estimación de tareas y en gestión de proyectos.

El equipo siempre consideró que éste era un riesgo a tener en cuenta en el seguimiento y control de los riesgos, por el impacto que podría significar para el proyecto, no poder realizar las tareas planificadas en el tiempo estimado. Para prevenirlo, siempre se buscó sobreestimar el esfuerzo que demandaría cada tarea, y de esa forma brindar un tiempo extra, de respaldo, para finalizarlas en caso de que hubiera retrasos. Durante los primeros *sprints* esta estrategia funcionó pero luego, cuando se comenzó a desarrollar en tecnologías desconocidas por los desarrolladores del equipo, hubo desvíos significativos. Esto se puede apreciar en las métricas que se presentan en la evolución de los *sprints*, tanto en el *sprint* seis como en el siete.

Retraso por detectar fallas en forma tardía por no utilizar casos, ni datos de prueba para realizar los diferentes tipos de pruebas.

Al momento de definir el proceso de pruebas dentro de la metodología de trabajo, se priorizó que fuera un proceso flexible y ágil. Siguiendo esa línea se decidió que para realizar pruebas unitarias y de integración, no se iban a desarrollar documentos previamente, sino que únicamente se registrarían fallas y comportamientos anómalos del sistema. Esto hizo que la responsabilidad de la completitud de las pruebas recayera sobre la persona de turno que las realizaba, y además en caso de que no cubriera los casos de borde, las fallas podrían manifestarse en *sprints* posteriores, probando o integrando otras funcionalidades. Esto iba a impactar directamente en el cronograma del proyecto, ya que se tendría que destinar esfuerzo a la localización y reparación de la falla inmediatamente para poder continuar.

Si bien esto fue un problema inminente a lo largo del proyecto, se decidió asumirlo. El hecho de contar con un equipo pequeño, que conocía bien el código, permitía arriesgar a que, si este problema se manifestaba, el error se localizaría y resolvería rápidamente. En definitiva, en la evaluación del costo/beneficio se consideró que la velocidad que brindaba esta forma de trabajar, superaba el costo que podía tener localizar fallas de forma tardía, donde además en última instancia se tendría las

pruebas alfa como filtro final. Para controlar este riesgo, sobre el final del proyecto el equipo volvió a pedir licencia laboral para destinar sus esfuerzos completamente al proyecto y poder reparar todas las fallas encontradas.

No cumplir con el alcance del producto por retrasos en la construcción del software.

El hecho de que este riego estuviera ligado directamente al producto, hizo que fuera cobrando importancia con el transcurso de los *sprints*. A su vez, también se encuentra ligado a otros riesgos que por momentos se convirtieron en problema, principalmente sobre la parte final del proyecto donde el equipo consideró cambiar de estrategia en cuanto al plan de acción.

La estrategia ante un riesgo de estas características fue mitigarlo, dado que no se pudo prevenir, no por el hecho de no haberlo intentado sino por consecuencia de otros problemas que no se pudieron prevenir. El plan de acción en la fase de construcción fue llevar a cabo programación de a pares, tratar de agilizar las tareas de construcción y que esto impactara en la curva de aprendizaje de los desarrolladores.

Insatisfacción de los usuarios finales con el producto.

En la medida que se fueron retrasando las tareas de construcción de los módulos, la *release* que se había planificado con la versión beta de la versión 1.0 de la aplicación también se retrasó.

Era más beneficioso obtener *feedback* y realizar pruebas alfa sobre un producto más básico, que no obtener *feedback* por falta de tiempo sobre un producto más pulido. Por ese motivo se decidió hacer la *release* con un producto menos maduro y recoger resultados que sirvieron como guía de evaluación.

Más allá de los resultados obtenidos en las pruebas de usabilidad, el equipo no encontró otra opción más que asumir este riesgo y tratar de trabajar de la mejor manera posible para que el producto satisficiera al cliente.

Retrasos en la construcción de funcionalidades por tareas no previstas/estimadas en la planificación del *sprint*.

Este riesgo se detectó luego de que en el tercer *sprint* se tuvieran que realizar diversas tareas que no habían sido planificadas al comienzo del *sprint*. Esto, sumado a la inexperiencia en la utilización de la metodología, llevó a que se desarrollaran muchos más *story points* de los estimados pero luego de vencido el plazo del *sprint*. Luego de este incidente, el tutor recomendó que en casos de este tipo, era conveniente abortar el *sprint* y volver a estimarlo. Desde ese *sprint* en adelante, siempre estuvo la probabilidad de no tener en cuenta alguna tarea importante para la construcción de una *user story*, motivo por el cual definimos un plan de acción.

11.8. Gestión de problemas

11.8.1. Introducción

En esta sección se busca mostrar que, si bien no supimos mitigar algunos riesgos y éstos se convirtieron en problemas, supimos encontrar la forma de adaptarnos y seguir adelante con el proyecto. En este caso, seguir adelante con el proyecto indica resolver los problemas, evitar que el proyecto fracase y cumplir con los objetivos planteados al principio del mismo.

11.8.2. Problemas

Los tres riesgos enumerados más abajo, en diferentes momentos del proyecto, se convirtieron en un problema para nuestro equipo. Esto quiere decir, que el riesgo inminente que representaban para los objetivos del proyecto se materializó. En este caso, el impacto que tuvieron sobre el proyecto fue el retraso del cronograma real con respecto al estimado. El hecho de que se manifestaran estos riesgos, hizo que otro riesgo, que en parte dependía de estos, tomara un nivel de criticidad muy importante. Este riesgo era el hecho de no cumplir con el alcance del proyecto. Afortunadamente, el equipo pudo gestionar estos problemas y resolverlos sin afectar el alcance.

Para cada uno de estos problemas vamos a explicar por qué se manifestaron y como se gestionaron para poder resolverlos.

Retraso del proyecto por falta de experiencia en la utilización de la metodología *scrum*.

Durante los primeros *sprints* el equipo no conocía la metodología, además trabajaba por separado y en horarios diferentes. Todo eso era contraproducente a la hora de adoptar y automatizar una forma de trabajo; las reuniones tomaban más tiempo del debido y costaba coordinar y asignar tareas.

Como medidas a este problema, se seleccionó un lugar de trabajo donde todos los integrantes pudieran concurrir, se definió un horario y un mínimo de horas en que todos los integrantes debían estar presentes. Cada uno de los integrantes acordó en su empleo un horario que le permitiera cumplir con esta premisa. Estos cambios hicieron que se comenzara a interactuar de forma más fluida y que los integrantes del equipo se pudieran ayudar mutuamente. Paulatinamente el equipo comenzó a mejorar en coordinación y fue adoptando la forma de trabajo, lo cual mejoró considerablemente la productividad.

Retrasos en la construcción del producto por tareas inesperadas o no estimadas en la planificación del *sprint*.

Como se mencionó anteriormente, al describir este riesgo, luego de que se manifestó el problema por primera vez en el *sprint* tres, el riesgo siempre fue inminente dado que siempre existe la probabilidad de olvidar planificar una tarea o simplemente desconocerla.

Por esa razón, se definió un plan de acción para enfrentar situaciones similares a lo largo del proyecto. Ante la aparición o detección de tareas no planificadas, se estimaba el esfuerzo de la o las tareas detectadas, si esta/s insumía/n menos de ocho horas, se seguía adelante con el *sprint*. En caso contrario, se abortaba el *sprint* y se re planificaba incluyendo todas las tareas involucradas.

Más adelante, en el quinto *sprint* del proyecto, este problema se volvió a manifestar y el esfuerzo que implicaban las tareas superaba las ocho horas de esfuerzo, motivo por el cual el quinto *sprint* fue abortado.

Retraso en la construcción del producto por complejidad de tareas en las plataformas seleccionadas.

Este riesgo se manifestó durante los *sprints* seis, siete y ocho. El mayor inconveniente que presentaba este riesgo es que no se puede resolver un problema que no se conoce y por eso nuestro plan para mitigarlo era realizar pruebas de concepto e investigación. Esto nos conducía a otro problema que era la falta de tiempo, el cual era una de las limitantes de nuestro proyecto. Por este motivo, la única solución que encontró el equipo fue aumentar la capacidad horaria y de dedicación al proyecto.

Para realizarlo, se coordinó una fecha conveniente entre los integrantes y todos solicitaron licencia en sus respectivos empleos. Con esto se logró dedicación total al proyecto durante dos semanas seguidas, lo que generó un impacto positivo en el avance de las tareas y se logró reducir el riesgo de no cumplir con el alcance del proyecto.

11.9. Métricas

11.9.1. Velocidad

La velocidad es una métrica típica de *scrum*, que sirve para medir el avance respecto al total de *story points* incluidos en el *product backlog,* es decir, la cantidad de *story points* construidos en cada *sprint.* En este proyecto se calcularon dos promedios de velocidad, el primero, desde el inicio hasta el tercer *sprint* y el segundo, luego de que se actualizara el *product backlog*, desde la cuarta iteración hasta el final del proyecto.

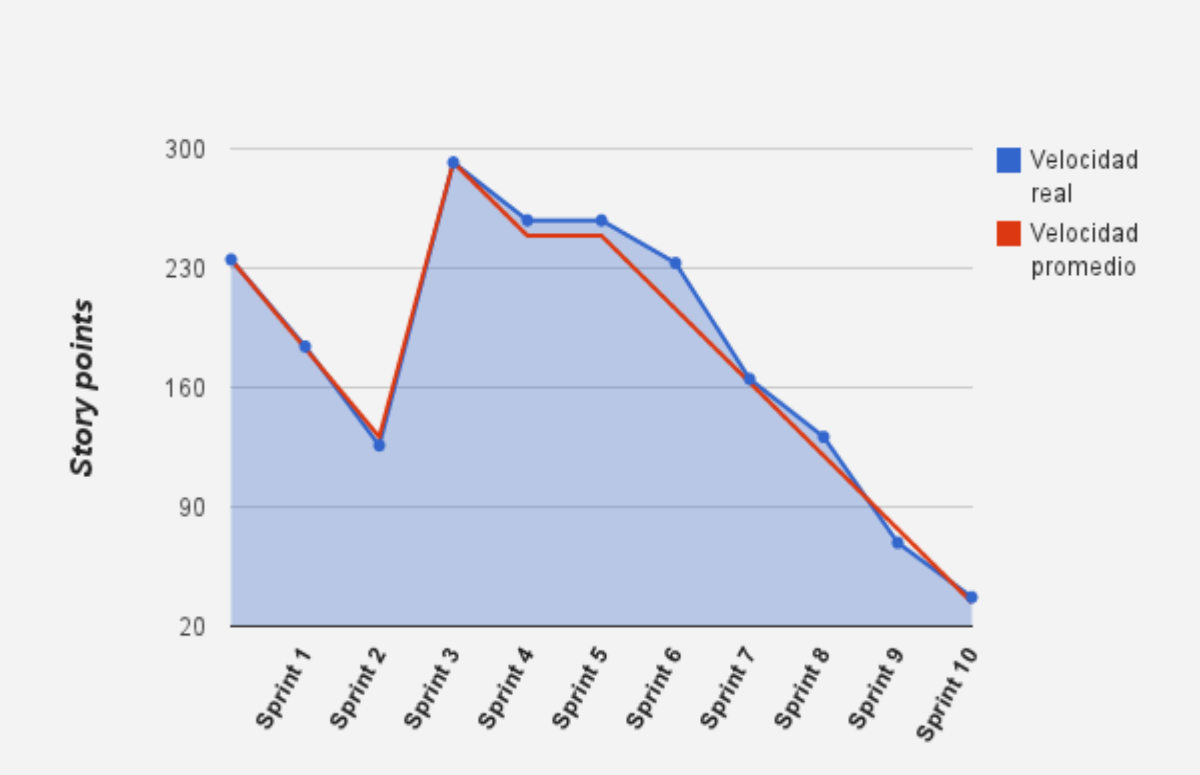


Figura 11.8 - Velocidad del proyecto vs velocidad promedio

Como se puede observar, el proyecto está marcado por la actualización del *product backlog* que se llevó a cabo en el tercer *sprint*. Para completar la primera versión del *product backlog*, el promedio de velocidad de avance fue de 52 *story points,* mientras que la construcción de la segunda versión, tuvo un promedio de 43 *story points*.

11.9.2. Desvío en horas

Esta métrica, busca mostrar la diferencia de horas que se habían estimado previo al desarrollo de un *sprint* y las horas que se dedicaron luego del mismo. En este caso, se tuvieron en cuenta únicamente las horas estimadas y dedicadas a la construcción del producto. Esta categoría, incluye horas dedicadas al análisis, diseño, desarrollo y pruebas.

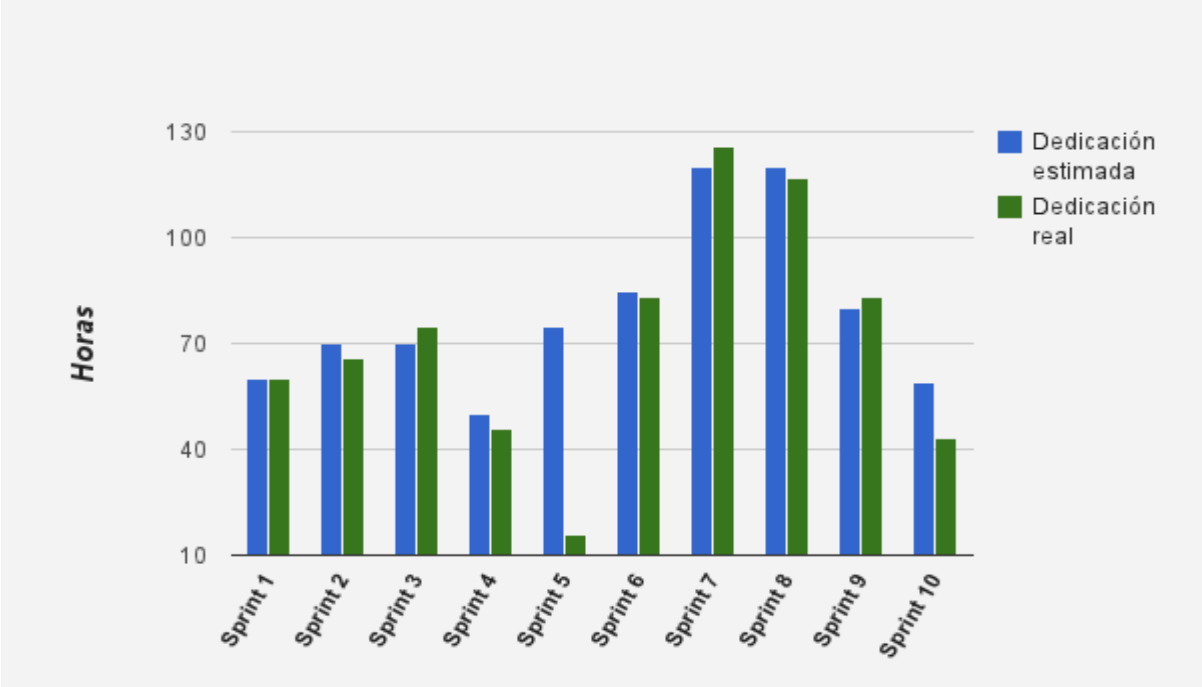


Figura 11.9 - Desvío de estimación de horas

Los *sprints* con mayor desfasaje son el quinto y el décimo. El *sprint* 5, se abortó luego del segundo día, mientras que en el 10, no se pudieron dedicar las horas estimadas porque el foco principal del equipo estaba en la documentación final.

11.9.3. Horas dedicadas

Esta métrica, muestra el total de la dedicación horaria que se dedicó en cada *sprint* dividida en dos categorías: horas dedicadas a gestión y/o documentación, y horas dedicadas a la construcción de producto.

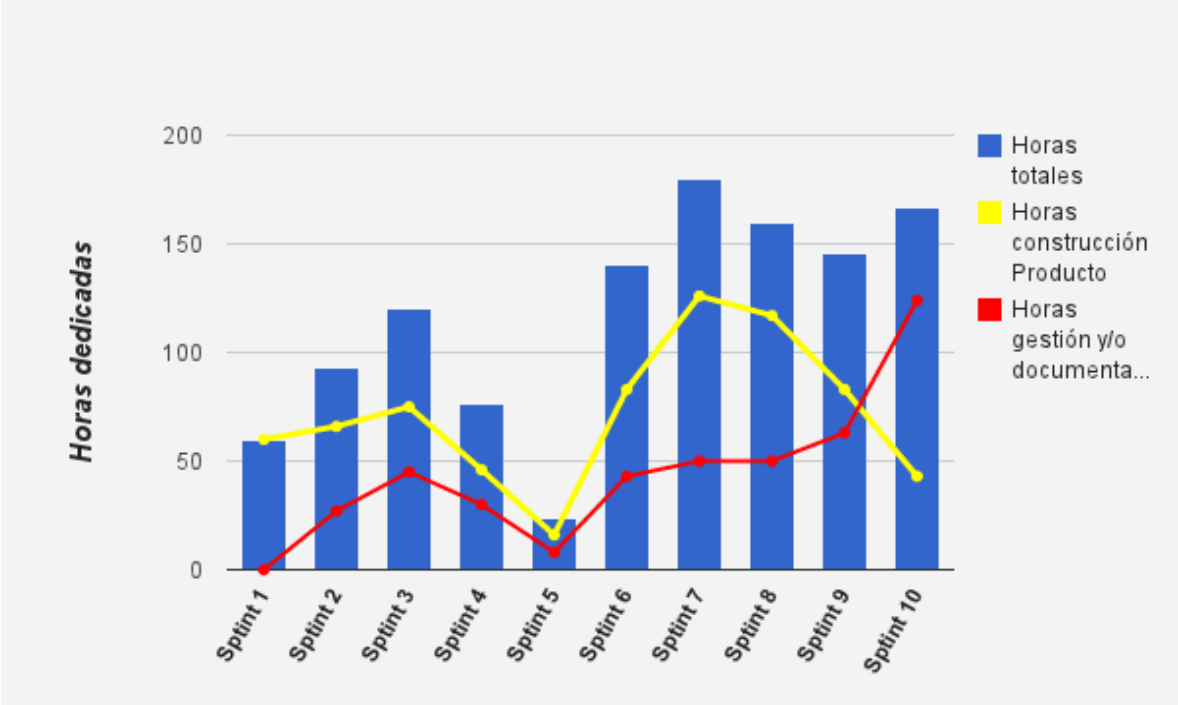


Figura 11.10 - Total de horas dedicadas. División entre horas de gestión y de desarrollo

Como se puede observar, durante los últimos *sprints* se encuentra la mayor dedicación horaria. En el séptimo y octavo, todos los integrantes del equipo solicitaron licencia laboral para dedicarse al proyecto en tiempo completo. Mientras que en el décimo, dos de los integrantes solicitaron licencia sin goce de sueldo para dedicarse al cierre del proyecto.

El total de horas dedicadas durante la fase de construcción es de 1166 horas, sumadas a las 387 horas dedicadas a la fase de puesta a punto y las 240 horas dedicadas a la documentación, da un total de 1793 horas de proyecto.

11.10. Evolución de los *sprints*

En la sección de métricas presentada anteriormente, se puede observar que los gráficos muestran la evolución de dichos números *sprint* a *sprint*. Las métricas presentadas, refieren a las características que creímos, eran más importantes de resaltar y no al total de métricas obtenidas en cada *sprint*.

Para cada iteración, se definieron objetivos de construcción, se obtuvieron métricas y conclusiones respecto a los objetivos y a los diferentes aspectos del funcionamiento del equipo y el proyecto (ver Anexo 15 : Evolución de los *sprints*).

Sprint	**Objetivos**	**Objetivo cumplido**	**Métricas**	**Conclusiones**
Sprint 1	- Registrar Empresa. - Editar Empresa - Listar Empresas. - ABM de administradores. - Inicio de sesión de usuario básico. - Inicio de sesión de súper-administrador. - Cerrar sesión - Registro de un usuario básico a una Empresa	Si	51 *story points* de avance, 60 horas de dedicación a la construcción de producto, no hubo dedicación a actividades de gestión, se dedicaron la cantidad de horas que se habían estimado y no se realizaron actividades de prueba por lo cual, no se encontraron fallas.	El equipo contó con un integrante menos. Las tareas de construcción se finalizaron luego de las dos semanas de duración del *sprint*, es decir, se continuó trabajando luego del cierre del mismo.
Sprint 2	- Cambio de contraseña - Iniciar sesión - Listar usuarios básicos registrados	No	58 *story points* de avance, 66 horas dedicadas a construcción, 27 horas dedicadas a gestión,	No se pudo finalizar Aceptar solicitud de contacto. Se siguió desarrollando luego de finalizadas las dos

	- Aceptar solicitud de contacto		desvío de 6 horas menos de las estimadas, 7 fallas encontradas.	semanas del *sprint*.
Sprint 3	- Listar secciones de aplicación - Enviar solicitud de contacto - Enviar mensaje	Si	47 *story points* de avance, 75 horas dedicadas a construcción, 45 horas dedicadas a gestión, desvío de 5 horas más que las estimadas y 10 fallas encontradas.	Surgieron cambios que debieron realizarse a mitad de *sprint*. Se continuó trabajando luego de transcurridas las dos semanas de *sprint*. Se realizarán modificaciones en la forma de trabajar.
Sprint 4	- Recibir mensaje.	Si	34 *story points*, 46 horas dedicadas a construcción, 30 horas dedicadas a gestión, desvío de 4 horas menos de las estimadas y 4 fallas encontradas.	Se contó con un integrante menos, motivo por el cual la estimación fue menos ambiciosa.
Sprint 5	- Crear grupo. - Agregar usuario básico a un grupo - Eliminar usuario básico de un grupo - Abandonar grupo - Eliminar grupo	No	Se dedicaron 16 horas a construcción, 8 horas a gestión, luego el *sprint* fue abortado.	Surgieron necesidades de modificación en la arquitectura, motivo por el cual se abortó el *sprint,* para volver a re-planificar.
Sprint 6	- Pruebas de concepto XMPP	No	25 *story points* de avance, 83 horas dedicadas a la	La integración de la nueva tecnología al proyecto fue

	- Integrar XMPP con aplicación - Crear grupo		construcción, 43 horas dedicadas a gestión y documentación, desvío de 2 horas menos de las estimadas y 7 fallas encontradas.	exitosa, pero no se pudieron finalizar todas las tareas estimadas.
Sprint 7	- Agregar usuario básico a un grupo - Eliminar usuario básico de un grupo - Abandonar grupo - Eliminar grupo	No	68 *story points* de avance, 126 horas dedicadas a la construcción, 50 horas dedicadas a la gestión y documentación, desvío de 6 horas más de las estimadas t 6 fallas encontradas.	La complejidad de algunas tareas desde el punto de vista tecnológico, hizo que la productividad fuera baja.
Sprint 8	- Agregar usuario básico a un grupo - Abandonar grupo - Eliminar grupo - Aspectos de configuración para plataforma iOS	Si	34 *story points* de avance, 117 horas dedicadas a la construcción, 50 horas dedicadas a gestión y documentación, desvío de 3 horas menos de las estimadas y 6 fallas encontradas.	Se finalizó la construcción del módulo de comunicación, pero restan realizar algunas pruebas. Se comenzó la construcción del módulo de soporte.
Sprint 9	- Enviar mensaje desde dispositivo a la web - Listar conversaciones con usuarios web	Si	62 *story points* de avance, 83 horas dedicadas a la construcción, 63 horas dedicadas a la gestión y documentación, desvío de 3 horas más de las	Se finalizó la construcción del módulo de soporte. Se realizarán pruebas alfa sobre el sistema.

	- Listar solicitudes de usuarios web - Aceptar solicitud de usuario web - Eliminar solicitudes aceptadas		estimadas y 5 fallas encontradas.	
Sprint 10	- Listar Empresas por rubro - Ver datos de Empresa - Listar contactos de Empresa - Listar empresas por nombre - Búsqueda (por nombre o rubro)	No	32 *story points* de avance, 43 horas dedicadas a la construcción, 124 horas dedicadas a la gestión y documentación, desvío de 16 horas menos de las estimadas y 3 fallas encontradas.	No se logró finalizar el total de *user stories* ya que se asignaron la mayoría de los recursos a tareas de documentación para la entrega final.

Figura 11.11 - Resumen de la evolución de los *sprints*.

12. Conclusiones

12.1 De Gestión

Durante el transcurso del proyecto se toman muchas decisiones, de hecho algunas se realizan casi de forma automática, por lo cual en esta sección se desea resaltar aquellas decisiones que en cierta forma marcaron el proyecto.

Una de las decisiones que el equipo consideró como acertada, al inico del proyecto, fue el hecho de realizar un *sprint* de puesta a punto. Si bien este *sprint* no estaba orientado a la construcción del producto, se construyeron herramientas para luego facilitar el trabajo. El objetivo principal era empezar a generar una sinergia de equipo y tomar contacto con la realidad que se iba a enfrentar, ya que esto iba a permitir tener una percepción real de las dificultades que había que superar. Si bien luego de este *sprint* fue necesario realizar más sprints para adoptar la metodología y ritmo de trabajo, éste sirvió para generar un marco de trabajo y para comenzar a ajustar detalles orientados a cómo realizar las tareas específicamente.

Siguiendo en orden cronológico, un aspecto negativo, que cobró importancia en la medida que avanzó el proyecto fue la estimación. Luego de terminar un proyecto real, el quipo considera que la estimación en general dentro de un proyecto es vital. Genera certezas en lugar de incertidumbre, y la incertidumbre generalmente se manifiesta con tensión en todo el equipo. En este caso, se comenzó estimando en base al juicio de cada uno de los integrantes, orientados por las horas que se estimaba dedicar y la complejidad que se percibía en las tareas. Luego, en el séptimo *sprint*, se comenzó a estimar en base a números obtenidos a partir del rendimiento anterior, con la certeza de que se debía mejorar en cuánto a la productividad. En esos momentos había preocupación dentro del equipo, pero no por no saber qué era lo que se hacía, sino por la urgencia a mejorar el rendimiento.

Por otro lado, una decisión importante, que marcó el proyecto y ya se ha mencionado, fue haber visto la necesidad del equipo de trabajar en conjunto. Conseguir una oficina, trabajar en un mismo lugar físico y compartir un mínimo de horas diarias, por muchas razones cambió el rumbo del proyecto. Desde el punto de vista del equipo, le dio sentido de pertenencia, generó buen clima laboral y permitió interactuar de forma más fluida, inclusive hasta para ayudar a algún compañero. Desde el punto de vista del proyecto, simplificó las actividades de gestión y

comunicación, y todo esto impactó en la productividad y la automatización de la forma de trabajo.

Por último, lo mencionado en el párrafo anterior fue clave para incorporar y aprender el proceso de ingeniería. El hecho de poder discutir, intercambiar opiniones e ideas hizo que todos los integrantes fueran conscientes del tipo de actividad que se estaba realizando dentro del proceso, y de la importancia de cada una de éstas. La calidad de la comunicación entre los integrantes del equipo y del equipo con actores externos, es clave en cualquier proyecto. No solo para la coordinación de tareas, sino también para la mejora continua de los procesos.

12.2. Alcance

El alcance fue un tema muy sensible y mantuvo con cierto grado de preocupación al equipo a lo largo del proyecto. Llegando a la fecha de entrega, se cerró el último módulo y el equipo celebró el cumplimiento del alcance pactado con el cliente, que además fue avalado por el tutor.

12.3 Satisfacción del cliente

Las pruebas de aceptación por módulo fueron un gran indicador de la satisfacción del cliente, pero no el único. El cliente demostró una actitud positiva y proactiva luego de las pruebas de aceptación ya que veía que estábamos materializando exitosamente el producto. El equipo se veía motivado por esta reacción, lo cual lo impulsaba a seguir trabajando con ganas.

12.4 Producto

El desarrollo del producto final permitió comprender como funcionan las aplicaciones de comunicación más utilizadas del planeta. Se pudieron imitar muchas de las funcionalidades básicas de éstas aplicaciones. Para lograrlo hubo que enfrentar tecnologías complejas las cuales requirieron de mucho esfuerzo y paciencia para entenderlas.

Se llegó a un producto de software que efectivamente funciona, permitiendo comunicar a sus usuarios de forma eficaz y eficiente. De acuerdo a los resultados obtenidos en las pruebas alfa, realizando pequeñas correcciones de fallas

detectadas, el producto estaría en condiciones de operar y de ser comercializado.

12.5 Conocimiento técnico adquirido

Todos los miembros del equipo obtuvieron conocimientos valiosos acerca de las tecnologías utilizadas durante este proyecto, principalmente con aquellas relacionadas a la mensajería instantánea (XMPP, Openfire), ya que ninguno de los miembros se había enfrentado a un desarrollo de esta naturaleza. También se adquirió experiencia valiosa al afrontar una situación crítica, en la cual se tuvo que hacer una análisis de la situación para seleccionar las tecnologías y recursos arquitectónicos óptimos para resolver el problema.

12.6 General

En definitiva, el equipo generó conocimiento y experiencia en todas las áreas del proceso de ingeniería de software. Cada integrante tuvo que interiorizarse en su área, investigar, procesar y aplicar nuevo conocimiento para el proyecto. Paralelamente también tuvo que retomar lo aprendido a lo largo de la carrera. Estas dos características en conjunto hicieron posible que cada integrante del equipo logre gestionar sus actividades y roles de forma profesional, adquirir un lenguaje técnico en el área de trabajo y acercarse un poco más a la figura de ingeniero.

12.7 Líneas de trabajo para el futuro

Moon Ideas está considerando lanzar una versión beta del producto al mercado. Esto generaría un gran *feedback,* el cual podría abrir las puertas a algunas modificaciones del producto retomando con el desarrollo del mismo. Puede suceder además que la versión beta tenga buenas repercusiones. Aquí el paso a dar sería desarrollar los módulos propuestos por el cliente en un principio y que no entraron en el alcance del proyecto y continuar madurando el producto.

13. Bibliografía

[1] WhatsApp in numbers: 500 million users, 700 million photos, 100 million videos (2014, Aug 10) [Online]. Disponible: http://timesofindia.indiatimes.com/tech/tech-news/WhatsApp-in-numbers-500-million-users-700-million-photos-100-million-videos/articleshow/34083916.cms

[2] Luke Johnson (2012). Texts overtake calls as most used communication method. [Online]. Disponible: http://www.t3.com/news/text-messaging-overtakes-calls-as-most-used-communication-method

[3] J. Palacio. (Abril 2014). Gestión de Proyectos Scrum Manager (Scrum Manager I y II) [Online]. Versión 2.5. Disponible: http://www.scrummanager.net/files/sm_proyecto.pdf

[4] Gestión de Proyectos (2014, Aug. 10) [Online]. Disponible: http://www.gestiondeproyectosit.es/blogit/2013/03/%C2%BFque-se-espera-de-un-product-owner-ii/

[5] Aplicando Scrum (2014, Aug. 10) [Online]. Disponible: http://www.aplicandoscrum.com/rol-perfil-product-owner/

[6] What is XMPP? (2014, Aug.10). [Online]. Disponilbe: http://xmpp.org/2007/10/what-is-xmpp/

[7] 4 + 1 Architectural view model (2014, Aug. 10). [Online]. Disponible: http://en.wikipedia.org/wiki/4%2B1_Architectural_View_Model

[8] PHP (2014, Aug. 10). [Online]. Disponible: http://en.wikipedia.org/wiki/PHP

[9] Features of Yii Framework (2014, Aug. 10). [Online]. Disponible: http://www.yiiframework.com/features/

[10] Aplication fundamentals (2014, Aug. 10). [Online]. Disponible: http://developer.android.com/guide/components/fundamentals.html

[11] XEP – 0124: Bidirectional – streams over synchronous http (BOSH) (2014, Aug 1). [Online]. Disponible: http://xmpp.org/extensions/xep-0124.html

[12] Openfire changelog (2014, Aug.1). [Online]. Disponible http://www.igniterealtime.org/builds/openfire/docs/latest/changelog.html

[13] Webusable. ¨Principios Heurísticos de Usabilidad ¨ [Online]. Disponible: http://www.webusable.com/heuristicos.htm

[14] Henrik Knieberg. (2007). Scrum and Xp from the trenches [Online]. 1. Disponible: http://wwwis.win.tue.nl/2R690/doc/ScrumAndXpFromTheTrenchesonline07-31.pdf

[15] Advantages of the “As a user, I want” user story template (2014, Aug. 1). [Online]. Disponible: http://www.mountaingoatsoftware.com/blog/advantages-of-the-as-a-user-i-want-user-story-template

[16] Cross origin resource sharing, W3C Recommendation 16 January 2014 (2014, Aug.1). [Online]. Disponible: http://www.w3.org/TR/cors/

[17] XMPP Standards Foundation (2014, Aug.10). [Online]. Disponilbe: http://xmpp.org/xmpp-protocols/xmpp-extensions/

14. Anexos

Anexo 1: Reglas base del Equipo

Este anexo contiene las reglas base que el equipo fue forjando a medida que evolucionaba el proyecto. Éstas surgieron poder alcanzar los objetivos de forma satisfactoria y además mantener un buen ambiente de trabajo haciendo más llevadero el largo proceso.

1. Todos los integrantes del equipo se comprometen a realizar el proyecto de principio a fin. Claro que ante problemas de salud o de índole similar el equipo demostrará entendimiento al respecto.

2. Los integrantes del equipo establecen que el proyecto es prioridad junto con sus respectivos trabajos. Se exigirá licencia a todos sin excepciones para aumentar la carga horaria y lograr avances cuando sea necesario.

3. Tanto los méritos como las deficiencias se reconocen al equipo entero y no a individuos.

4. Los integrantes del grupo deben actuar de forma moral y ética.

5. Luego de divididos los roles y tareas es responsabilidad de cada uno de los integrantes cumplir con la misma para llevar adelante el proyecto.

6. Es fundamental estar siempre en contacto con el equipo, no desaparecer y mostrar interés para lograr un avance parejo y equitativo del proyecto.

7. Ante decisiones importantes se debe escuchar la voz de todos los integrantes del equipo y además del tutor, un docente experto o el cliente.

8. Los integrantes se comprometen a asistir a todas las reuniones de equipo.

9. Los miembros se comprometen a no revelar datos confidenciales del equipo y del cliente a terceros.

Anexo 2: Requerimientos del Sistema

Aquí se presentan los requerimientos funcionales del sistema extraídos en la reunión de relevamiento con el cliente. Este anexo incluye módulos que van más allá del alcance del proyecto.

2.1. SITIO WEB DE ADMINISTRACIÓN

RF01 - Registro de una empresa

Descripción: El sistema deberá permitir que una empresa se registre, generando una petición de aceptación. La empresa quedará a la espera de una aceptación por parte del sistema.

Prioridad: Nivel 1.

RF02 - Editar empresa

Descripción: El sistema deberá permitir a los administradores modificar la información referida a una empresa, como ser dirección, teléfono, nombre, etc.

Prioridad: Nivel 1.

RF03 - Listar empresas

Descripción: El sistema podrá mostrar un listado a los administradores con todas las empresas registradas.

Prioridad: Nivel 1.

RF04 - Listar planes

Descripción: El sistema deberá mostrar un listado con todos los planes comerciales posibles para el consumo del producto. Estos planes están disponibles para todo público.

Prioridad: Nivel 1.

RF05 - Agregar Plan

Descripción: El sistema debe permitir a los administradores agregar nuevos planes comerciales.

Prioridad: Nivel 1.

RF06 - Ver estadísticas de empresa

Descripción: El sistema debe permitir visualizar a los administradores estadísticas de una empresa dada. Por ejemplo, total de mensajes enviados dentro de una empresa, cantidad de reuniones agendadas , cantidad de consultas realizadas a través del sitio web de una empresa, etc.

Prioridad: Nivel 1.

RF07 - Ver estadísticas de usuario

Descripción: El sistema debe permitir visualizar estadísticas de un usuario dado. Por ejemplo, cantidad de mensajes enviados entre dos fechas, total de archivos recibidos, cantidad de reuniones agendadas, etc.

Prioridad: Nivel 1.

RF08 - Ver estadísticas de sistema

Descripción: El sistema debe permitir visualizar estadísticas generales , por ejemplo, fluctuación de mensajes enviados por empresa en un periodo determinado, comparación del envío de mensajes entre empresas de un mismo rubro, alertas en caso de que la aplicación entre en desuso, etc.

Prioridad: Nivel 1

RF09 - ABM de administradores

Descripción: El sistema debe permitir crear administradores para la gestión y control del mismo. Los administradores deberán proveer un nombre y una contraseña para ingresar al sistema.

Prioridad: Nivel 1.

2.2. SITIO WEB PARA EMPRESAS

RF10 - Iniciar sesión

Descripción: El sistema permitirá a las empresas registradas entrar al sitio de administración de empresas.

Prioridad: Nivel 1.

RF11 - Cerrar sesión

Descripción: El sistema permitirá a las empresas registradas entrar al sitio de administración de las mismas.

Prioridad: Nivel 1.

RF12 - Registro de usuario de una empresa

Descripción: El sistema deberá permitir registrar empleados a empresas registradas.

Prioridad: Nivel 1.

RF13 - Asignar rol a usuario

Descripción: Se le podrá asignar un rol a un usuario. Estos roles determinar qué acciones puede realizar el usuario. Se le llama usuario a aquellos empleados registrados.

Prioridad: Nivel 1.

RF14 - Ver y modificar datos de la empresa

Descripción: El sistema deberá permitir a empresas registradas ver y modificar sus datos , como ser dirección, teléfono, nombre, etc.

Prioridad: Nivel 1.

RF15 - Cambio de contraseña

Descripción: El sistema deberá permitir cambiar la contraseña a empresas registradas.

Prioridad: Nivel 1.

2.3. WEB CHAT / API

RF16 - Comunicación con API REST

Descripción: El sistema deberá proveer una API REST para permitir que el sitio web de una empresa registrada se comunique con él.

Prioridad: Nivel 1.

RF017 - Consulta Web

Descripción: Los mensajes recibidos por el sistema (aquellos que una empresa envía a través de la API REST) deberán ser recibidos por los operadores responsables en sus dispositivos móviles o terminales correspondientes .

Prioridad: Nivel 1.

2.4. APLICACIÓN MOBILE - Chat entre compañeros de empresa

RF18 - *Login*

Descripción: Un empleado registrado podrá acceder a su cuenta a través de la aplicación en su dispositivo móvil. El mismo ingresará su email y contraseña para entrar.

Prioridad: Nivel 1

RF19 – *Logout*

Descripción: Un empleado registrado autenticado deberá poder cerrar su cuenta.

Prioridad: Nivel 3

RF20 - Listar secciones de la aplicación

Descripción: El empleado registrado de la aplicación tendrá acceso a las cuatro secciones de la misma, estas son: web, chat, proyecto y servicios. La sección web contiene todas las consultas realizadas a través de la página web de la empresa. Estas consultas serán respondidas por aquellos usuarios responsables, o sea según su rol. La sección chat contiene las conversaciones entre el usuario logueado y todos sus contactos. Aquí el usuario puede enviar y recibir mensajes. La sección proyectos contiene todas las tareas asignadas al usuario por proyecto. Al ir completando estas tareas el usuario marcará las mismas. Así, el *project manager* logrará seguir el avance del empleado. La sección servicios permite buscar empresas para comunicarse con las mismas. Además, contiene un listado de las empresas que ya se ha comunicado el usuario.

Prioridad: Nivel 1.

RF21 - Enviar solicitud de contacto

Descripción: Un empleado registrado podrá enviar una solicitud de contacto a cualquier otro usuario de la empresa.

Prioridad: Nivel 1.

RF22 - Aceptar solicitud de contacto

Descripción: Un empleado registrado podrá aceptar o rechazar una solicitud de contacto. Una vez que se acepta la solicitud se agregaran respectivamente a la lista de contactos de los mismos.

Prioridad: Nivel 1.

RF23 - Listar empleados registrados

Descripción: Un empleado registrado podrá listar a todos los empleados registrados de su empresa.

Prioridad: Nivel 1.

RF25 - Enviar mensajes

Descripción: Un usuario podrá enviar un mensaje a cualquiera de sus contactos.

Prioridad: Nivel 1.

RF24 - Recibir mensaje

Descripción: Un empleado registrado podrá recibir mensajes. Estos mensajes provienen de los contactos del receptor.

Prioridad: Nivel 1.

RF26 - Enviar/recibir archivos

Descripción: Un usuario podrá enviar un archivo a cualquiera de sus contactos.

Prioridad: Nivel 1.

RF27 - Listar Contactos

Descripción: Un empleado registrado podrá listar a todos sus contactos.

Prioridad: Nivel 2.

RF28 - Agendar reuniones multiusuario

Descripción: Un usuario debe poder agendar reuniones con uno o más contactos de su lista. A cada uno de los usuarios involucrados en la reunión que acepte la invitación, le quedará fijada una reunión en el calendario de su celular. Las alertas para avisar cada reunión podrán ser configurables por cada usuario.

Prioridad: Nivel 2.

2.5. SERVICIOS

RF29 - Recibir mensajes de empresas

Descripción: Aquellos empleados registrados que tengan el rol de “contacto representante” recibirán mensajes provenientes de empresas externas a la misma, brindando asistencia y servicios a quien lo necesite.

Prioridad: Nivel 1.

RF30 - Buscar empresas registradas por rubro

Descripción: Cada empresa que se registra en el sistema pasa a formar parte de un directorio de empresas. De esta manera, el sistema debe brindar un servicio de búsqueda de empresas por rubro, para que las empresas registradas se puedan interconectar. Al encontrar una empresa deseada, se agrega la misma a lista de empresas.

Prioridad: Nivel 2.

2.6. SECCIÓN PROYECTOS

RF31 - Ver listado de proyectos

Descripción: Un empleado registrado podrá listar en su dispositivo móvil todos aquellos proyectos en los que actualmente se encuentre trabajando.

Prioridad: Nivel 2.

RF32 - Crear proyecto

Descripción: Un empleado registrado con el rol de *project manager* podrá crear proyectos nuevos. Una vez creado, el mismo se agregará de forma automática a los debidos empleados.

Prioridad: Nivel 2.

RF33 - Editar Proyecto

Descripción: Un empleado registrado con el rol de *project manager* podrá editar proyectos que él haya creado, modificando recursos, nuevas tareas, etc.

Prioridad: Nivel 2.

RF34- Crear Tarea

Descripción: Un empleado registrado con el rol de *project manager* podrá crear tareas dentro de un proyecto a su cargo.

Prioridad: Nivel 2.

RF35 - Editar Tarea

Descripción: Un empleado registrado con el rol de *project manager* podrá editar tareas dentro de un proyecto a su cargo.

Prioridad: Nivel 2.

RF36 - Datos de proyecto

Descripción: Cualquier empleado registrado que sea parte de un proyecto podrá ver el avance del mismo, o sea, que tareas se han realizado, cuales faltan por terminar, fechas límite del proyecto, etc.

Prioridad: Nivel 2.

RF37 – Alarma

Descripción: En caso que una tarea esté retrasada se activará una alarma en su dispositivo móvil indicando al empleado indicando su estado de atraso.

Prioridad: Nivel 2.

RF38 - Asignar tarea

Descripción: Un usuario podrá asignarle una o más tareas a cualquiera de los usuarios de su lista de contactos. Para poder asignar tareas se deben cumplir una serie de requisitos que deberán ser especificados (por ejemplo, el usuario al que le asignan tareas debe tener un cargo inferior al del usuario que asigna tareas).

Prioridad: Nivel 2.

Anexo 3: Estándares de Desarrollo

3.1. Estándares de Codificación para los principales lenguajes

3.1.1. Php, Objective C y Java.

1) Todo el código se escribe en inglés (nombre de métodos, atributos, constantes, clases, etc.). Ya que los lenguajes son los tres en inglés, o sea, todas la palabras reservadas, clases de biblioteca, etc. se encuentran en inglés se logra una integración entre estas características de los lenguajes y el código, permitiendo una lectura continua del mismo.

2) Se escribe solamente una declaración por línea. Facilita la lectura y deja el código prolijo para así ser mantenible. Esto también permite a primera vista resaltar variables y funciones fácilmente.

3) Primero se listan todas los atributos seguido de todos los métodos. Ídem a anterior.

4) Se deja una línea en blanco debajo de: el nombre de la clase, el último atributo, el encabezado del método, y el paréntesis de cierre de un método. Ídem a anterior.

5) Se aplica una sangría debajo de la declaración de la clase, el encabezado de un método, estructuras de control (for, while, foreach). Este ítem se aplica luego del ítem 4. Brinda orden al código y permite al lector (o al mismo programador) ubicarse velozmente.

6) Uso de try-catch para controlar las excepciones en los métodos de mayor importancia. Mantiene la aplicación funcionando incluso en casos de errores lógicos de parte de programadores o usuarios malintencionados.

7) Nomenclatura Camel Casing para nombrar a las atributos, funciones y clases. Permite leer fácilmente palabras complejas.

8) Nombres de atributos, funciones y clases mnemotécnicos. Ayudan al lector a contextualizarse y entender el código para así poder ajustarlo rápida y ordenadamente.

3.1.2. Html y Css

1) Escribir todos los estilos de una página en un archivo css independiente con el mismo nombre que la página. Encapsula los estilos de una página separándolos del Html, esto emprolija al Html facilitando la lectura del mismo favoreciendo la mantenibilidad.

2) Todos los archivos css ubicarlos en una carpeta con el nombre "css". Agiliza la identificación del archivo a través de un paralelismo con la página que contiene el Html.

3.1.3. JavaScript

1) Escribir un solo archivo javascript por página con el mismo nombre que la misma. Encapsula las animaciones de una página, esto emprolija al código Html facilitando la lectura del mismo favoreciendo la mantenibilidad.

2) Todos los archivos javascript ubicarlos en una carpeta con el nombre "js". Ídem al ítem 2 de Html y Css.

3) Utilizar un objeto principal con el nombre el mismo nombre del archivo en cuestión. Este objeto debe contener todos los métodos y atributos más importantes del archivo. La nomenclatura para definir los métodos y atributos debe ser : nombreDelArchivo.nombreDelAtributo y nombreDelArchivo.nombreDelMetodo respectivamente.

3.2. Criterios para tablas de la base de datos

1) Nombres de tablas en inglés, plural y minúscula.

2) Para nombres de tablas con palabras compuestas, los mismos deben llevar un guion bajo entre las palabras.

3) Las claves primarias de todas las tablas se llaman ID.

4) Nombre de atributos en Inglés.
5) Para nombres de atributos con palabras compuestas se aplica la misma nomenclatura que para las tablas.

7) El nombre de las “tablas de relación” se compone del nombre de las dos tablas que se relacionan con un guión bajo entre los dos nombres.

3.3. Heurísticas de Nielsen

1) Visibilidad del estado del sistema. El sistema debe informar a los usuarios del estado del sistema, brindando la información apropiada en un tiempo razonable.

2) Utilizar el lenguaje de los usuarios. Para lograr que los usuarios tengan una experiencia amigable y fluida con el sistema utilizamos un lenguaje claro y lo más amigable posible.

3) Control y libertad para el Usuario. En caso de que el usuario quiera deshacer una acción, el mismo podrá hacerlo siempre y cuando la acción sea identificado por el sistema como reversible.

4) Consistencia y Estándares. El usuario debe poder confiar en que el significado de una palabra dada es el mismo en todo el sistema.

5) Prevención de errores. Es vital no dar lugar a la aparición de errores. Para esto es necesario concentrarse en prevenirlos antes de mostrar buenos mensajes de error.

6) Minimizar la carga de la memoria del usuario. El usuario debe usar su memoria lo menos posible. El sistema guía al usuario para realizar la acción que desea.

7) Flexibilidad y eficiencia de Uso. Todo tipo de usuario debería poder utilizar la aplicación de forma fluida, desde los principiantes a los más expertos.

8) Diálogos estéticos y diseño minimalista. Mantener al mínimo la cantidad de elementos en una pantalla para no confundir a los usuarios.

9) Ayudar a los usuarios a reconocer, diagnosticar y recuperarse de los errores. Los mensajes de error deben expresarse en un lenguaje claro, indicar exactamente el problema y ser constructivos.

10) Ayuda y documentación. Es mejor que el sistema sea utilizado sin documentación, pero en algunos casos puede ser necesario proporcionar ayuda y documentación (manual de usuario). Dicha información debe ser fácil de buscar, estar orientada en la tareas del usuario, y no ser demasiado extensa.

Anexo 4: Estructura y formato de *scrum*

De *scrum* se deprenden muchos conceptos que facilitan la aplicación de la metodología de trabajo. En este anexo se presentan particularmente aquellos de estructura de tabla para el *product backlog* y escritura de las *user stories*.

4.1. Estructura del product backlog

De la estructura de tabla para el *product backlog* provista por *scrum* [14] se tomaron las columnas acordes al proyecto y se agregó una columna "Módulo" ya que el producto estaba dividido en módulos. Ésta fue de utilidad ya que al leer una *user story* ayudó a ubicarla rápidamente en un contexto. La siguiente tabla muestra la estructura de columnas planteada para la tabla del *product backlog*.

ID	Módulo	Nombre	Descripción	Estimación

4.2. Formato para la descripción de una *user strory*

El formato para la descripción de *user stories* fue : "As a <type of user>, I want <some goal> so that <some reason> [15]". Este formato puede verse en cada una de las *user stories* del *product backlog*. El formato fue traducido al español para su uso.

Anexo 5: Manual de Usuario para Administradores

El siguiente documento describe detalladamente las acciones que puede realizar una administrador del sistema. No se requieren conocimientos informáticos avanzados para la utilización de este documento.

5.1. Iniciar Sesión

Como administrador de su empresa debe ingresar a través de su browser a http://chat.moonideas.com/site/clientUserLogin.

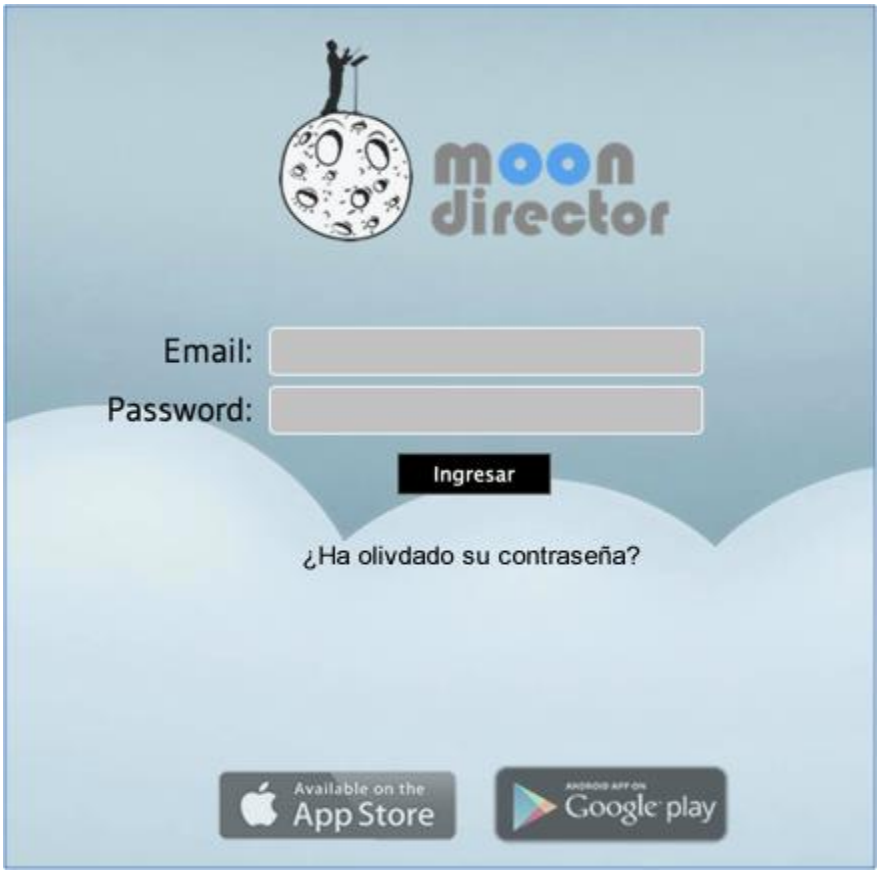


Figura 1 – Iniciar sesión

Ingrese su usuario y contraseña para poder entrar al sistema.

5.2. Crear un Usuario Básico

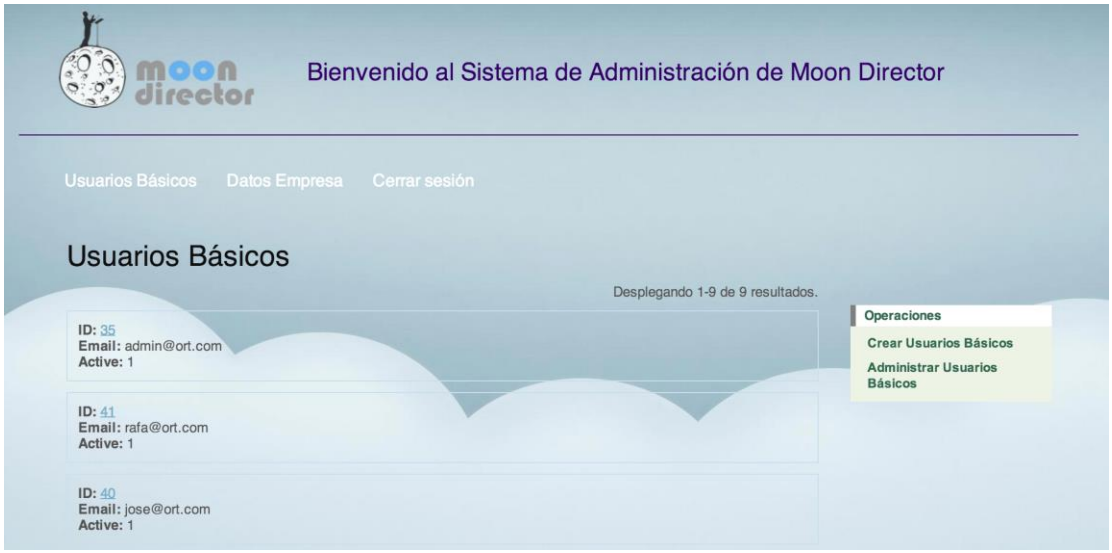


Figura 2 – Listado de Usuarios Básicos

Primero haga clic en “Usuarios Básicos” en el menú principal. Ahora para poder crear un nuevo usuario básico haga clic en “Crear Usuarios Básicos” del menú lateral (figura 2).

Figura 3 – Datos Usuario Básico

Deberá ingresar el email del usuario que desea agregar, darle una contraseña, seleccionar si el mismo es operador, referente y si está activo o no.

5.3. Modificar un Usuario Básico

Para modificar un Usuario Básico vaya a “Usuarios Básicos” en el menú principal, luego haga clic en “Administrar Usuario” (ver figura *2*) en el menú lateral. Será redirigido a una nueva página y verá el siguiente listado.

Figura 4 – Administración de Usuarios Básicos

Haga clic en el lápiz que se encuentra entre la lupa y la cruz roja. Esto lo llevará a otra página donde se desplegarán los datos de usuario básico seleccionado. Edite los datos y sálvelos para terminar.

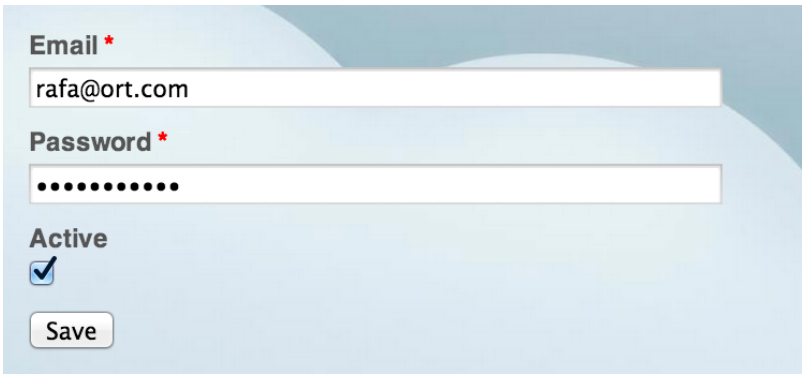


Figura 5 – Modificar Usuario Básico

5.4. Eliminar Usuario Básico

Para eliminar un Usuario Básico vaya a “Usuarios Básicos” en el menú principal, luego haga clic en “Administrar Usuario” (ver figura 2). Haga clic en la cruz roja correspondiente al usuario a eliminar.

5.5. Ver Datos Empresa

Para ver los datos de su empresa haga click en “Datos Empresa” (ver figura 2) del menú principal. Se mostrán los datos como muestra la figura 6.

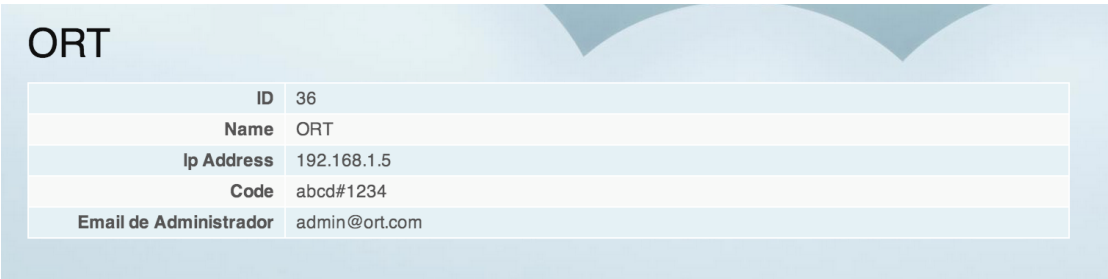


Figura 6 – Datos Empresa

5.6. Cerrar Sesión

Si desea cerrar la sesión haga clic en “Cerrar Sesión” del menú principal (ver figura 2). Será redirigido a la página de inicio de sesión de administradores.

Anexo 6: Manual de Usuario (Aplicación Móvil)

En este manual el usuario podrá ver las acciones permitidas por la aplicación móvil.

6.1. Primeros pasos

6.1.1. Iniciar Sesión

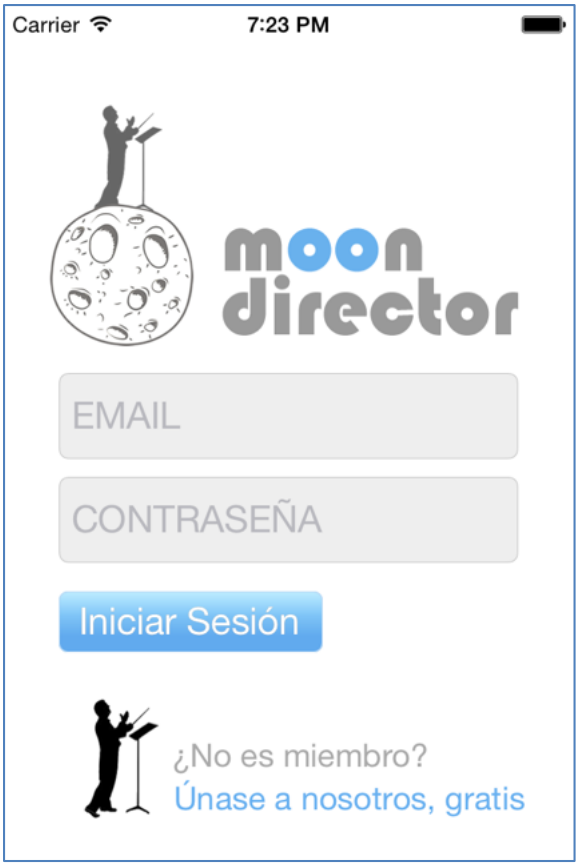


Figura 6.1 – Iniciar sesión

Para comenzar a utilizar la aplicación ingrese su nombre de usuario y contraseña en los campos de textos adecuados y presione el botón “iniciar sesión”.

6.1.2. Acceder a módulo de Comunicación, Soporte y Servicios

Figura 2 - Principal

Para comenzar a utilizar la aplicación deberá seleccionar uno de los tres módulos posibles. Si quiere comunicarse con contactos de su misma empresa apriete “Comunicar”, si quiere atender consultas de la página web de la empresa apriete “Soporte” y por último si quiere buscar información de otras empresas apriete “Servicios”.

6.2. COMUNICACIÓN

6.2.1. Agregar Contacto

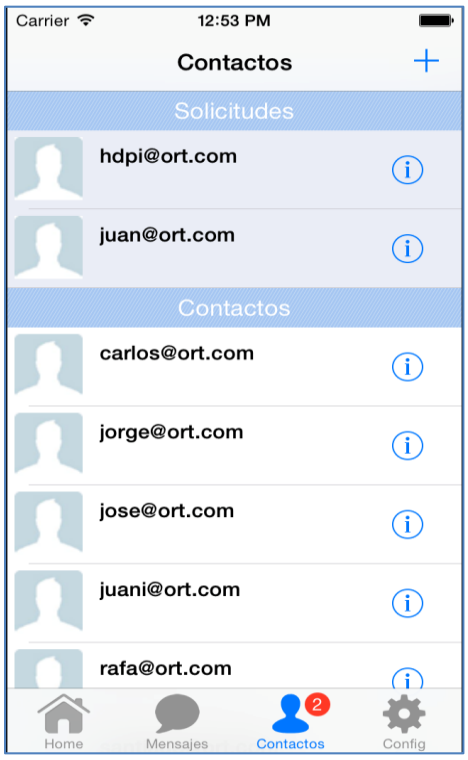


Figura 3 – Solicitudes y Contactos

Una vez dentro del módulo de Comunicación presione la sección de “Contactos” en el menú que se encuentra en la parte inferior de la pantalla. Se le desplegará la panta que se muestra en la figura 3, ahora presione el botón de “+” ubicado en la parte superior derecha de la pantalla. Luego, seleccione el contacto que desea enviarle una solicitud.

Una vez enviada la solicitud, el contacto solicitado podrá o no aceptar la misma. No hay un tiempo límite para que el contacto solicitado acepte, usted se enterará a través de una notificación una vez que esto suceda.

6.2.2 - Aceptar Solicitud de Contacto

Luego de que le llegue una solicitud de contacto usted podrá ver las mismas, como muestra la figura 3, listadas en la sección de “Solicitudes”.

Para aceptar a un usuario presione arriba de la solicitud que usted desea aceptar. Se le desplegará el siguiente mensaje:

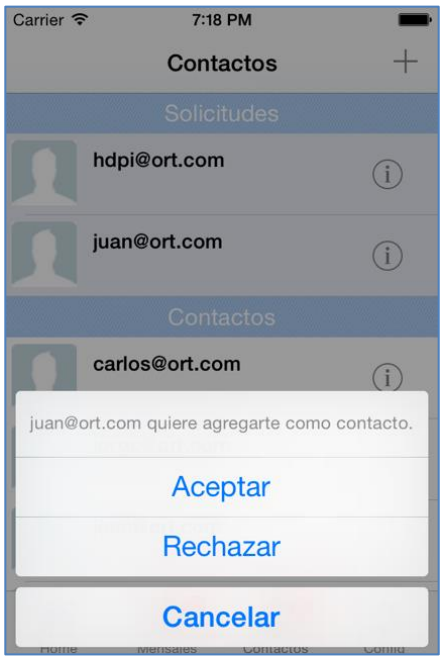


Figura 4 – Aceptar solicitud de contacto

Si usted desea aceptar a esta usuario simplemente presione “aceptar”, de lo contrario presione “rechazar”.

6.2.3. Enviar un mensaje

Una vez dentro del módulo de Comunicación presione la sección de “Mensajes” en el menú que se encuentra en la parte inferior de la pantalla.

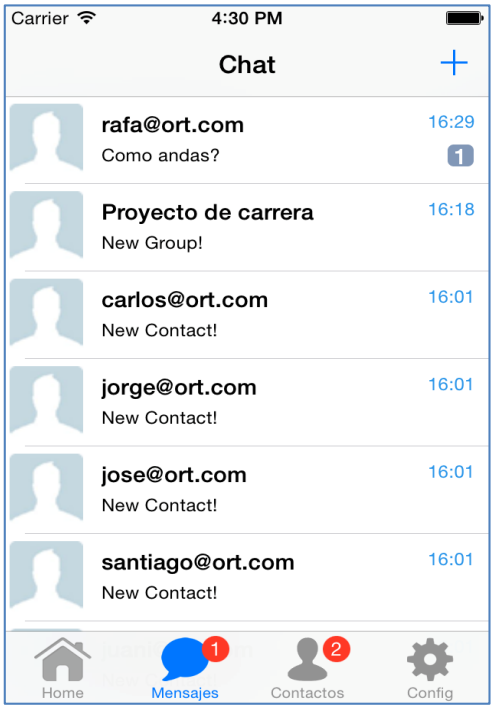


Figura 5 – Listado de chats

Seleccione el contacto al cual usted quiere enviar el mensaje. Se desplegará la siguiente pantalla:

Figura 6 – Mensajes con un contacto

Escriba el mensaje que desea enviar en la caja de texto correspondiente. Una vez que haya terminado de componer su mensaje apriete el botón de “Enviar”.

Aclaración: Para enviar un mensaje usted primero tiene que haber sido aceptado o haber aceptado la solicitud de otro usuario. Para verificar si usted tiene un contacto especifico entre a la sección de “Contactos” y observe en el listado de “Contactos” si visualiza al usuario en cuestión.

6.2.4 Crear un grupo

Para crear un grupo abra la sección de “Mensajes” (figura 4). Presione el botón de “+” situado en la parte superior de la pantalla. Se le desplegará la siguiente pantalla:

Figura 7 – Crear grupo

Complete el nombre del grupo y agregue a aquellos usuarios que quiera sean miembros de su grupo. Una vez que haya finalizado de configurar el grupo apriete el botón “Crear” situado en la parte superior derecha de la pantalla.

6.2.5 - Agregar nuevo miembro al grupo

Abra la sección de “Mensajes” y seleccione un grupo. Luego, en apriete el botón de “Info” situado en la parte superior derecha de la pantalla. Se desplegará la siguiente pantalla:

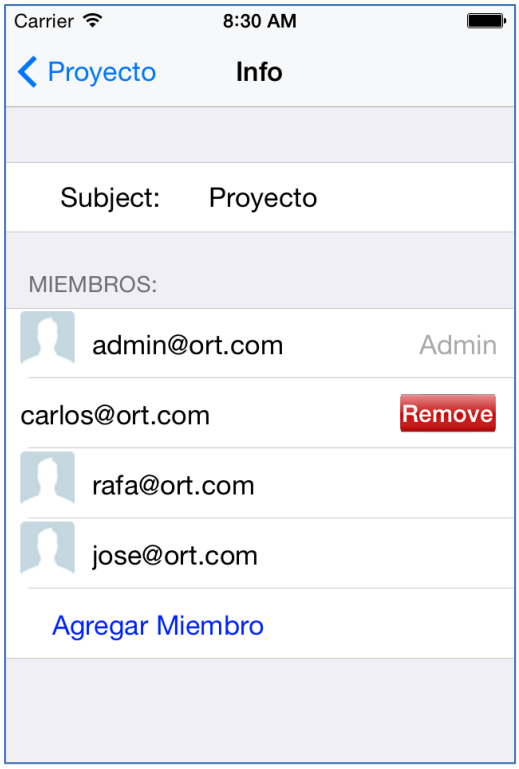


Figura 8 – Agregar miembro

Apriete el botón “Agregar Miembro” ubicado al finalizar el listado de miembros. Se le desplegará el listado de sus contactos. Aquí usted debe seleccionar el nuevo contacto a agregar.

6.2.6. Eliminar miembro del grupo

Abra la sección de “Mensajes” y seleccione un grupo. Luego, en apriete el botón de “Info” situado en la parte superior derecha de la pantalla. Una vez ahí seleccione el contacto que desea eliminar. Se desplegará un botón de “Eliminar”, apriételo.

6.2.7. Abandonar grupo

Abra la sección de “Mensajes” y seleccione un grupo. Luego, en apriete el botón de “Info” situado en la parte superior derecha de la pantalla. Se desplegará la siguiente pantalla:

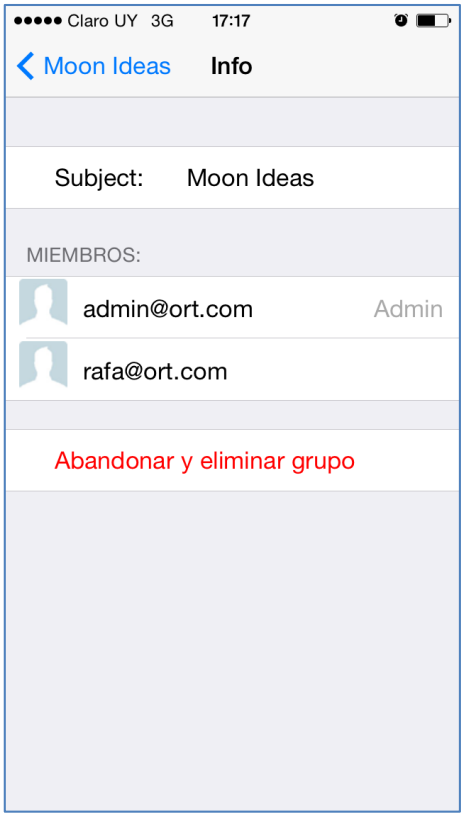


Figura 9 – Abandonar grupo

Apriete el botón de “Abandonar y eliminar grupo”. Se desplegará un mensaje preguntado si realmente desea abandonar el grupo:

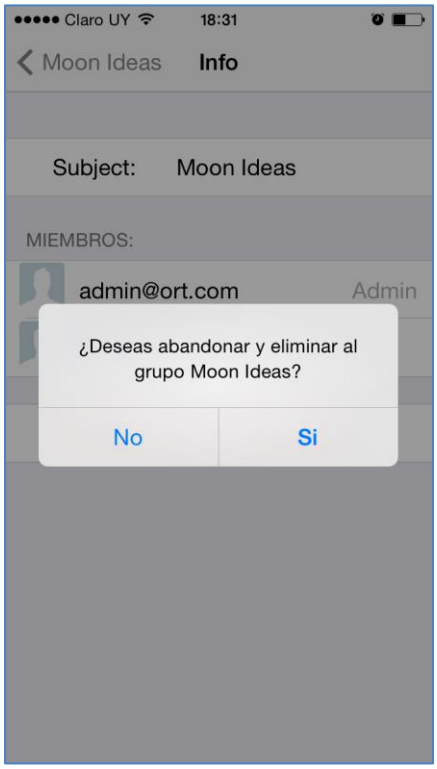


Figura 10 – Confirmar baja de grupo

En caso de que desee terminar la acción presione “Si”, de lo contrario presione “No”.

6.3. SOPORTE

6.3.1. Atender una consulta web

Una vez dentro del módulo de Soporte presione la sección de “Solicitudes” en el menú que se encuentra en la parte inferior de la pantalla. Se le desplegará la siguiente pantalla:

Figura 11 – Solicitudes Web

Usted podrá aceptar solicitudes que no hayan sido aceptadas por otros operadores. Como se ve en la figura 7 aquellas solicitudes que están esperando ser aceptadas se indican con el texto "Aceptar!" y aquellas que aún no han sido aceptadas se indican con "Cancelada!".

6.3.2. Borrar solicitudes por tipo

Presione la papelera que se encuentra en la parte superior de derecha de la pantalla. Se desplegará un mensaje con cuatro opciones:

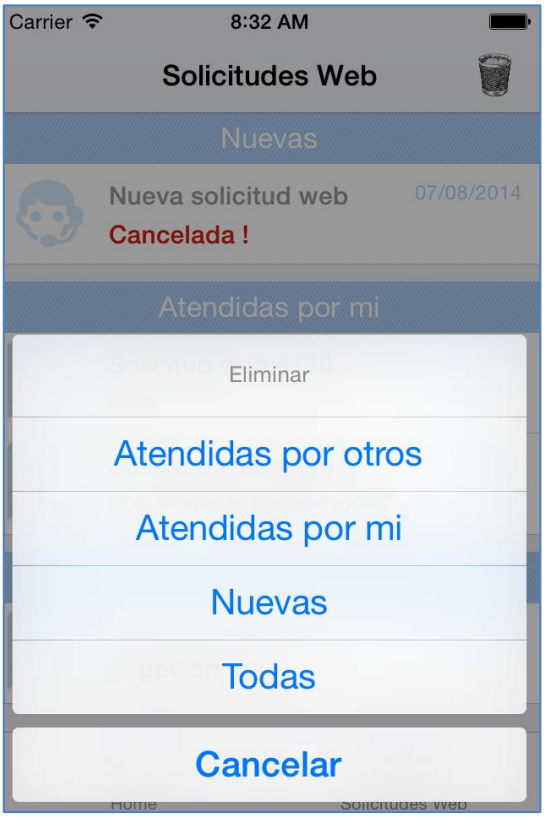


Figura 12 – Eliminar solicitudes

Para eliminar las solicitudes aceptadas por otros operadores seleccione “Atendidas por otros”.

Si quiere eliminar solicitudes aceptas por usted seleccione “Atendidas por mí”.

En caso de querer eliminar solicitudes que aún no ha aceptado ningún operador apriete “Nuevas”.

Si usted quiere borrar las solicitudes de todos los tipos (atendidas por otros, por mí y nuevas) apriete “Todas”.

6.3.2. Enviar mensaje a usuario web

Proceda en este módulo de la misma forma que se envía un mensaje en el módulo de Comunicación.

Anexo 7: *Product backlog*

7.1. Módulo de Administración

ID	Nombre	Descripción	Estimación (S.P.)
1	Registro de una empresa	Como súper administrador quiero poder registrar empresas para que sean parte del sistema y así puedan empezar a gestionarse.	5
2	Editar empresa	Como súper administrador quiero poder modificar la información referida a una empresa para poder actualizar su información cuando esta cambia.	5
3	Listar empresas	Como súper administrador quiero poder obtener un listado de todas las empresas registradas en el sistema para tener una visión global de las mismas para poder accederlas.	3
4	Ver estadísticas del sistema	Como súper administrador quiero poder ver estadísticas generales del sistema (como ser fluctuación de mensajes enviados por una empresa en un período determinado) para poder controlar, evaluar y monitorear el estado general de una empresa y así tomar decisiones de mejora.	8
5	Ver estadísticas de empresa	Como súper administrador quiero poder ver las estadísticas de una empresa para poder controlar y evaluar las empresas.	8
6	Ver estadísticas de usuario	Como súper administrador quiero poder visualizar las estadísticas de los usuarios básicos para una empresa con el fin de controlar y monitorear la actividad de los mismos.	8
7	ABM de administradores	Como súper administrador quiero poder crear nuevos administradores para una empresa con el fin de delegarle la gestión y control de la misma.	8
8	Iniciar sesión usuario	Como administrador quiero poder ingresar a la sección web correspondiente para administrar	13

	básico	usuarios básicos y los datos de mi empresa.	
9	Iniciar sesión súper administrador	Como súper administrador quiero poder ingresar a la sección web correspondiente para agregar y gestionar empresa, agregar planes y ver estadísticas del sistema.	8
10	Cerrar sesión	Cualquier tipo de usuario que contenga una sesión iniciada debe poder cerrarla antes de abandonar el sitio con el fin de filtrar el acceso al sistema a otros posibles usuarios de ese mismo equipo.	1
11	Registro de un usuario básico a una empresa	Como administrador quiero poder ingresar a la sección web correspondiente para poder registrar nuevos usuarios básicos a mi empresa.	5
12	Asignar rol a usuario básico	Como súper administrador quiero poder ingresar a la sección web correspondiente para poder asignar uno o más roles a los usuarios básicos.	3
13	Cambio de contraseña	Como usuario básico (de cualquier rol) debo poder cambiar mi contraseña para tener la libertad de poner un contraseña personal.	13

Figura 1 – Módulo de Administración.

7.2. Módulo de Comunicación

ID	Nombre	Descripción	Estimación (S.P.)
14	Inicio de sesión	Como usuario básico debo poder iniciar sesión desde mi dispositivo móvil con mi email de la empresa para así poder comunicarme y/o atender consultas.	21
15	Cierre de sesión	Como usuario básico debo poder cerrar sesión desde mi dispositivo móvil con el propósito no dejar mi dispositivo al alcance al uso de cualquier persona.	13

16	Lista de secciones de la aplicación	Como usuario básico logueado en mi dispositivo móvil debo tener un menú principal para así poder navegar por la aplicación.	3
17	Enviar solicitud de contacto	Como usuario básico logueado en mi dispositivo móvil debo poder enviar una solicitud de contacto a otro usuario básico registrado para poder comunicarme con él.	13
18	Aceptar solicitud de contacto	Como usuario básico logueado en mi dispositivo móvil debo poder aceptar una solicitud de contacto por parte de otro usuario básico registrado para poder comunicarme con él.	21
19	Listar usuario básico registrados	Como usuario básico logueado en mi dispositivo móvil debo poder visualizar a todos los usuarios básicos de la empresa que están registrados en la aplicación para poder conocer cuáles son mis potenciales contactos en caso de que los requiera.	13
20	Enviar mensaje	Como usuario básico logueado en mi dispositivo móvil debo poder enviar mensajes a cualquiera de mis contactos para poder entablar una comunicación instantánea.	21
21	Recibir mensaje	Como usuario básico logueado en mi dispositivo móvil debo poder recibir mensajes de cualquiera de mis contactos para poder entablar una comunicación instantánea.	34
22	Listar contactos	Como usuario básico logueado en mi dispositivo móvil debo poder visualizar todos mis contactos en un listado para comprobar si puedo entablar comunicación con determinado usuario básico o para seleccionar un destinatario para mi próximo mensaje.	8
23	Crear Grupo	Como usuario básico debo poder crear un Grupo para poder enviar y recibir mensajes a más de un contacto simultáneamente y que todos los miembros pertenecientes al grupo puedan enviar y recibir mensajes de todos los demás miembros en un espacio en común.	55

24	Agregar usuario básico al grupo	Como creador de un grupo debo poder agregar usuario básico, siempre y cuando este se encuentre entre mis contactos, para que pueda interactuar con los demás miembros del grupo.	8
25	Eliminar usuario básico del grupo	Como creador del grupo debo poder eliminar usuario básico para que deje de recibir y enviar mensajes a todos los demás miembros del grupo.	5
26	Abandonar Grupo	Como usuario básico debo poder abandonar un grupo al que pertenezco cuando quiera para dejar de recibir mensajes de todos los miembros del grupo. En los grupos en los que sea creador, al abandonar el grupo, el mismo debe desaparecer.	8
27	Eliminar grupo	Como creador de un grupo debo poder eliminarlo, para dejar de recibir mensajes de cualquiera de sus miembros por esa vía. Al eliminarlo ninguno de los miembros del grupo se podrá seguir comunicando con el resto de los miembros a través de ese espacio.	13

Figura 2 – Módulo de Comunicación.

7.3. Módulo de Soporte

ID	Nombre	Descripción	Estimación (S.P.)
30	Comunicarse de la web al dispositivo	Como operador de una empresa quiero poder recibir consultas de un usuario web en mi celular, a través de nuestro sitio web, para que los usuarios web perciban un valor agregado en materia de eficiencia, velocidad y disponibilidad para su atención.	21

31	Enviar mensaje del dispositivo a la web	Como operador quiero poder responder las consultas de un usuario web desde mi celular, a través de nuestro sitio web, para que los usuarios web perciban un valor agregado en materia de eficiencia, velocidad y disponibilidad para su atención.	21
32	Listar conversación con usuario web	Como operador quiero poder listar las últimas conversaciones con los usuarios web, para conversar con los usuarios web en línea o ver el historial de conversación.	8
33	Listar solicitudes de usuario web	Como operador quiero poder listar las solicitudes web, para ver los usuarios web que realizaron un solicitud recientemente.	8
34	Aceptar solicitud usuario web	Como operador debo poder aceptar una solicitud web, para poder brindarle asistencia al usuario web.	13
35	Eliminar solicitudes aceptadas	Como operador debo poder eliminar las solicitudes aceptadas, para no sobrecargar la pantalla de despliegue con solicitudes que no van a ser utilizadas o ya son antiguas.	8

Figura 3 – Módulo de Soporte.

7.4. Módulo de Servicios

ID	Nombre	Descripción	Estimación (S.P.)
30	Listado de empresas por rubro	Como usuario básico quiero poder ver todas las empresas de Montevideo ordenadas por rubro al que pertenecen.	13
31	Listado de empresas por nombre	Como usuario básico quiero poder ver todas las empresas de Montevideo ordenadas por nombre	3
32	Búsqueda (rubro o nombre)	Como usuario básico quiero poder buscar una empresa filtrando por el nombre de la misma o por el rubro al que se dedica.	13

33	Ver datos empresa	Como usuario básico quiero seleccionar una empresa de un listado y visualizar información sobre la misma. Nombre, rubro, información de contacto, etc.	3
34	Listar contactos de empresa	Como usuario básico quiero poder visualizar todos los usuarios de una empresa, cuyo número de teléfono pertenece a mi agenda de contactos telefónicos.	13

Figura 4 – Módulo de Servicios.

Anexo 8: Pruebas de Aceptación

La siguiente tabla muestra el grado de satisfacción asignado a cada *user story* del *product backlog*. Los posibles valores para la satisfacción de un *user story* pueden ser : 0 - 25 - 50 - 75 – 100.

ID	Módulo	Nombre	Satisfacción
1	Administración	Registro de una empresa	100
2	Administración	Editar empresa	100
3	Administración	Listar empresas	100
4	Administración	Ver estadísticas del sistema	-
5	Administración	Ver estadísticas de empresa	-
6	Administración	Ver estadísticas de usuario	-
7	Administración	ABM de administradores	100
8	Administración	Iniciar sesión usuario básico	100
9	Administración	Iniciar sesión súper administrador	100
10	Administración	Cerrar sesión	100
11	Administración	Registro de un usuario básico a una empresa	100
12	Administración	Asignar rol a usuario básico	100
13	Administración	Cambio de contraseña	25
14	Comunicación	Inicio de sesión	100
15	Comunicación	Cierre de sesión	50
16	Comunicación	Lista de secciones de la aplicación	100
17	Comunicación	Enviar solicitud de contacto	50

18	Comunicación	Aceptar solicitud de contacto	100
19	Comunicación	Listar usuario básico registrados	100
20	Comunicación	Enviar mensaje	100
21	Comunicación	Recibir mensaje	100
22	Comunicación	Listar contactos	100
23	Comunicación	Crear Grupo	50
24	Comunicación	Agregar usuario básico al grupo	100
25	Comunicación	Eliminar usuario básico del grupo	100
26	Comunicación	Abandonar Grupo	100
27	Comunicación	Eliminar grupo	50
28	Soporte	Comunicarse de la web al dispositivo	100
29	Soporte	Enviar mensaje del dispositivo a la web	100
30	Soporte	Listar conversación con usuarios web	100
31	Soporte	Listar solicitudes de usuarios web	25
32	Soporte	Aceptar solicitud web usuario	100
33	Soporte	Eliminar solicitudes aceptadas	100
34	Servicios	Listado de empresas por rubro	100
35	Servicios	Listado de empresas por nombre	100
36	Servicios	Búsqueda (rubro o nombre)	50
37	Servicios	Ver datos empresa	100
38	Servicios	Listar contactos de empresa	100

Figura 1 – Satisfacción del cliente por módulo.

Anexo 9: Pruebas de Usabilidad

9.1. Presentación

Aquí se presenta la evaluación de usabilidad entregada a los usuarios y los datos se les pidió a los mismos.

Datos.

Los datos requeridos para la evaluación son: nombre, edad, profesión, responsable, fecha, hora de inicio y hora fin. Se presentan de la siguiente manera.

Nombre:..... Edad:.....
Profesión:..... Responsable:.....
Fecha:...../...../..... Hora Inicio:.....:..... Hora Fin:.....:.....

Cuestionario

La tabla muestra los principios de Nielsen adaptados al proyecto. Como se puede apreciar el usuario puede ingresar uno de tres posibles afirmaciones para cada uno de los principios (Si, No o Parcialmente) y en caso de querer expresar alguna crítica y/o mejora existe la parte de “propuestas” para permitirle ingresarla.

	Respuestas			
Principio	**Si**	**No**	**Parcialmente**	**Propuestas**
El sistema debe informar el estado del sistema, brindando la información apropiada en un tiempo razonable.				
Experiencia amigable y fluida con el sistema a través un lenguaje claro y amigable.				
El sistema debe permitir deshacer una acción siempre y cuando la acción sea de tipo reversible.				

El usuario debe poder confiar en que el significado de una palabra dada es el mismo en todo el sistema.				
El sistema muestra un mensaje en pantalla cuando no se ingresa un valor correcto o el mismo es vacío.				
El sistema guía al usuario para realizar la acción que desea.				
Todo tipo de usuario debería poder utilizar la aplicación de forma fluida, desde los principiantes a los más expertos.				
Mantener al mínimo la cantidad de elementos en una pantalla para no confundir a los usuarios				
Los mensajes de error deben expresarse en un lenguaje claro, indicar exactamente el problema y ser constructivos				
El usuario cuenta con un manual para el uso del sistema.				

Figura 1 – Evaluación de los principios de Nielsen.

9.2. Resultados

9.2.1. Evaluaciones a usuarios con Android OS.

La siguiente evaluación fue tomada al alzar del conjunto total de evaluaciones realizadas a usuarios con Android OS.

Nombre: Juan Sorondo. **Edad:** 21.
Profesión: Estudiante y desarrollador.

Responsable: Rafael Etcheverry **Fecha:** 06 /08/2014. **Hora Inicio:** 14:00. **Hora Fin:** 14:25.

	Respuestas			
Principio	**Si**	**No**	**Parcialmente**	**Propuestas**
El sistema debe informar el estado del sistema, brindando la información apropiada en un tiempo razonable.	Si			
Experiencia amigable y fluida con el sistema a través un lenguaje claro y amigable.	Si			
El sistema debe permitir deshacer una acción siempre y cuando la acción sea de tipo reversible.			parcial	"Intenté borrar un chat pero no pude hacerlo. No se si no se puede a propósito."
El usuario debe poder confiar en que el significado de una palabra dada es el mismo en todo el sistema.	Si			
El sistema muestra un mensaje en pantalla cuando no se ingresa un valor correcto o el mismo es vacío.	Si			
El sistema guía al usuario para realizar la acción que desea.		No		"Cuando abrí la aplicación no sabía bien que hacer."
Todo tipo de usuario debería poder utilizar la aplicación de forma fluida, desde los principiantes a los más expertos.	Si			
Mantener al mínimo la cantidad de elementos en una pantalla para no confundir a los usuarios	Si			

Los mensajes de error deben expresarse en un lenguaje claro, indicar exactamente el problema y ser constructivos	Si			
El usuario cuenta con un manual para el uso del sistema.			Parcial	“Considero esta afirmación parcial porque le faltan explicar acciones que puede realizar el usuario.”

Figura 2 – Muestra de evaluación para plataforma Android OS.

9.2.1. Evaluaciones a usuarios con iOS.

La siguiente evaluación fue tomada al alzar del conjunto total de evaluaciones realizadas a usuarios con iOS.

Nombre: Diego Castiglioni. **Edad:** 26.
Profesión: Estudiante y desarrollador.
Responsable: José Luis Obes. **Fecha:** 07 /08/2014. **Hora Inicio:** 19:00. **Hora Fin:** 19:15.

	Respuestas			
Principio	**Si**	**No**	**Parcialmente**	**Propuestas**
El sistema debe informar el estado del sistema, brindando la información apropiada en un tiempo razonable.	Si			
Experiencia amigable y fluida con el sistema a través un lenguaje claro y amigable.	Si			
El sistema debe permitir deshacer una acción siempre y cuando la acción sea de tipo reversible.			Parcial	“No me deja eliminar un contacto.”

El usuario debe poder confiar en que el significado de una palabra dada es el mismo en todo el sistema.	Si			
El sistema muestra un mensaje en pantalla cuando no se ingresa un valor correcto o el mismo es vacío.	Si			
El sistema guía al usuario para realizar la acción que desea.	Si			
Todo tipo de usuario debería poder utilizar la aplicación de forma fluida, desde los principiantes a los más expertos.	Si			
Mantener al mínimo la cantidad de elementos en una pantalla para no confundir a los usuarios	Si			
Los mensajes de error deben expresarse en un lenguaje claro, indicar exactamente el problema y ser constructivos	Si			
El usuario cuenta con un manual para el uso del sistema.			Parcial	“Podría estar más completo.”

Figura 3 – Muestra de evaluación para iOS.

Anexo 10 : Hosting (Total Server Solutions)

Durante el desarrollo del proyecto, se tuvieron que realizar varias tareas de configuración de servidores, tanto para el servidor HTTP como para el servidor XMPP. En un principio, trabajando en un ambiente local, el manejo de los servidores se llevó a cabo de manera exitosa, todas las necesidades de la solución fueron cubiertas por el equipo. Sin embargo el ambiente local solo permite que dispositivos que están en la misma red local se comuniquen, lo cual por razones obvias no era suficiente en un ambiente de producción.

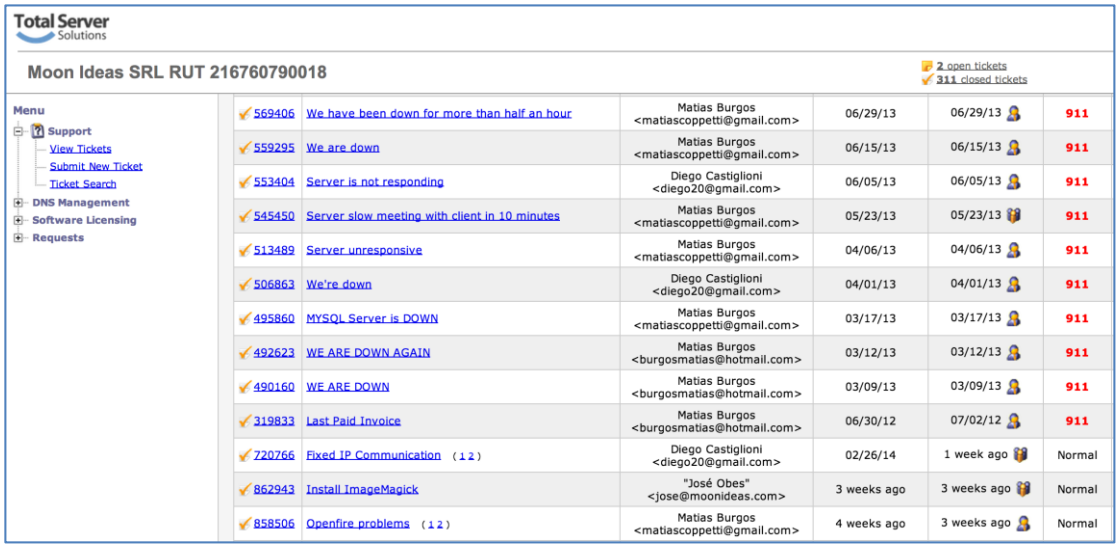
Para lograr la conectividad a través de internet, hubo que contratar un servicio de *hosting* dedicado. Se investigaron varias compañías que ofrecen servicios de *hosting* (Linode, Bluehost, GoDaddy, HostGator), pero al final, a sugerencia del cliente se optó por la compañía Total Server Solutions.

Las razón principal por la cual se seleccionó esta compañía es por el servicio de atención al cliente que brindan, ya que hoy en día, de una forma u otras todas las compañías de aprovisionamiento de *hosting* son capaces de brindar la potencia y las herramientas necesarias para mantener soluciones web con miles de usuarios. Sin embargo, no todas brindan facilidad de manejo, en cuanto a la instalación y configuración de los distintos sistemas, requiriendo en muchos casos experiencia en la utilización de SSH para poder realizar las tareas de manera remota; experiencia que no poseía ninguno de los miembros del equipo.

Total Server Solutions mantiene un equipo especializado, el cual se encarga de brindar soporte en todo momento del día, todos los días de la semana. El cliente simplemente debe solicitar ayuda y rápidamente es contactado por un representante que se encarga de realizar las tareas necesarias. Para realizar estas solicitudes se utiliza un sistema de *tickets*, mediante el cual se intercambian mensajes entre los clientes y el equipo de soporte de la compañía.

En este proyecto se utilizó este servicio en reiteradas ocasiones para varias tareas, entre las que se encuentra la instalación y configuración de los distintos servidores y la solución de los problemas con los cuales el equipo se encontró a lo largo del desarrollo.

A continuación se muestra evidencia fotográfica de la utilización del sistema de *tickets* de Total Server Solutions por parte del equipo. Las siguientes imágenes son de una conversación llevada a cabo entre uno de los integrantes del equipo y el *staff* de soporte:



Total Server Solutions

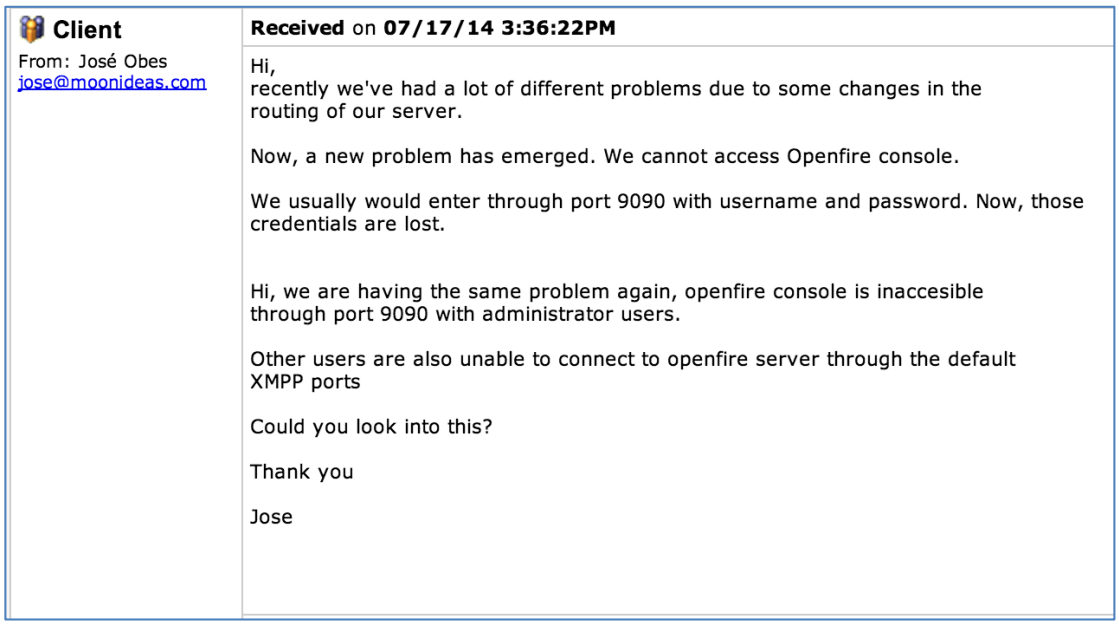
Moon Ideas SRL RUT 216760790018

2 open tickets
311 closed tickets

Menu
- Support
  - View Tickets
  - Submit New Ticket
  - Ticket Search
- DNS Management
- Software Licensing
- Requests

569406	We have been down for more than half an hour	Matias Burgos <matiascoppetti@gmail.com>	06/29/13	06/29/13	911
559295	We are down	Matias Burgos <matiascoppetti@gmail.com>	06/15/13	06/15/13	911
553404	Server is not responding	Diego Castiglioni <diego20@gmail.com>	06/05/13	06/05/13	911
545450	Server slow meeting with client in 10 minutes	Matias Burgos <matiascoppetti@gmail.com>	05/23/13	05/23/13	911
513489	Server unresponsive	Matias Burgos <matiascoppetti@gmail.com>	04/06/13	04/06/13	911
506863	We're down	Diego Castiglioni <diego20@gmail.com>	04/01/13	04/01/13	911
495860	MYSQL Server is DOWN	Matias Burgos <matiascoppetti@gmail.com>	03/17/13	03/17/13	911
492623	WE ARE DOWN AGAIN	Matias Burgos <burgosmatias@hotmail.com>	03/12/13	03/12/13	911
490160	WE ARE DOWN	Matias Burgos <burgosmatias@hotmail.com>	03/09/13	03/09/13	911
319833	Last Paid Invoice	Matias Burgos <burgosmatias@hotmail.com>	06/30/12	07/02/12	911
720766	Fixed IP Communication (1 2)	Diego Castiglioni <diego20@gmail.com>	02/26/14	1 week ago	Normal
862943	Install ImageMagick	"José Obes" <jose@moonideas.com>	3 weeks ago	3 weeks ago	Normal
858506	Openfire problems (1 2)	Matias Burgos <matiascoppetti@gmail.com>	4 weeks ago	3 weeks ago	Normal

Figura 1 – Listado de tickets creados por el equipo



Client
From: José Obes
jose@moonideas.com

Received on **07/17/14 3:36:22PM**

Hi,
recently we've had a lot of different problems due to some changes in the routing of our server.

Now, a new problem has emerged. We cannot access Openfire console.

We usually would enter through port 9090 with username and password. Now, those credentials are lost.

Hi, we are having the same problem again, openfire console is inaccesible through port 9090 with administrator users.

Other users are also unable to connect to openfire server through the default XMPP ports

Could you look into this?

Thank you

Jose

Figura 2 – Solicitud de asistencia (equipo de proyecto)

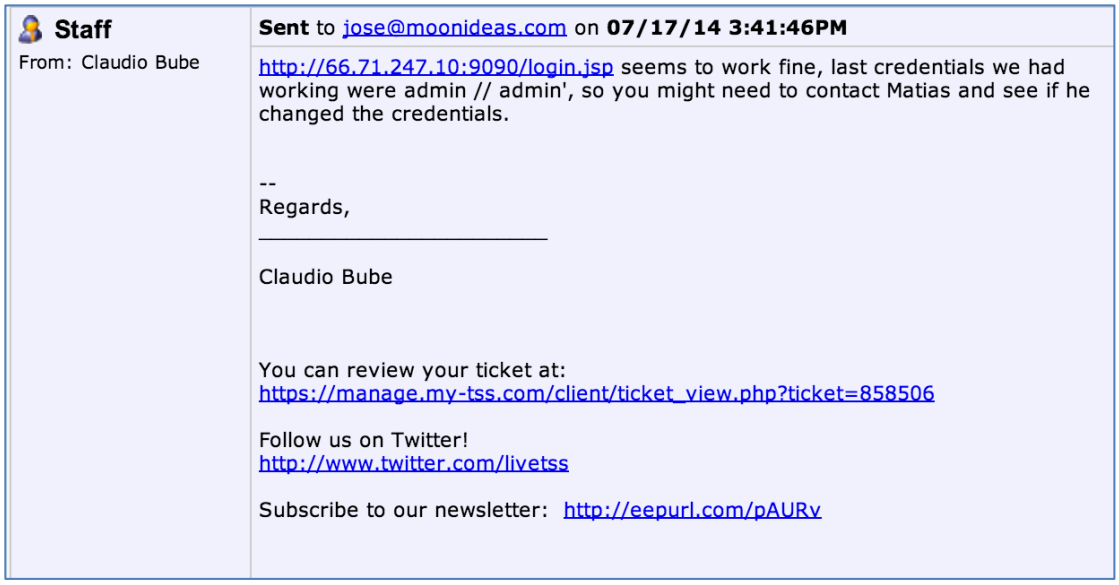


Figura 3 – Solicitud de asistencia (respuesta de equipo de soporte)

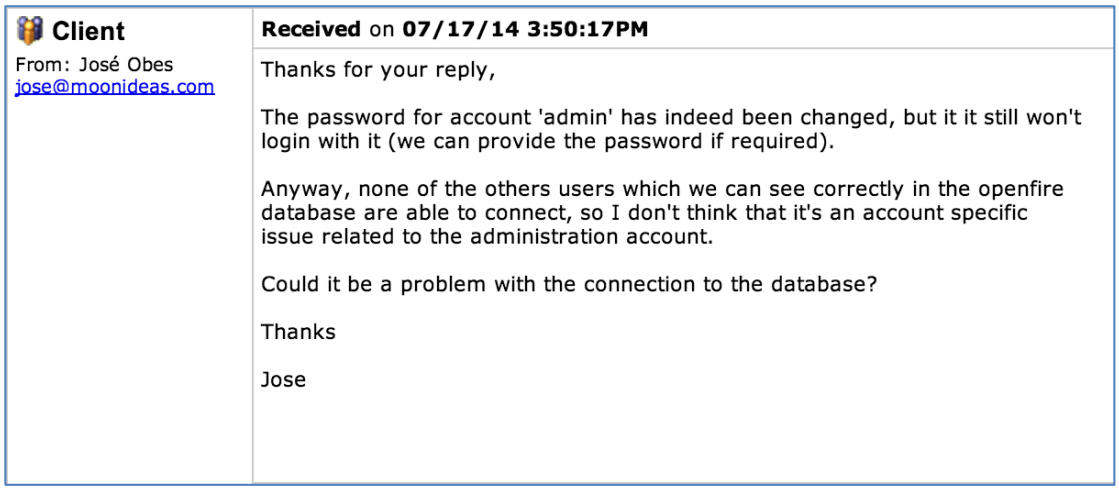


Figura 4 – Solicitud de asistencia (equipo de proyecto)

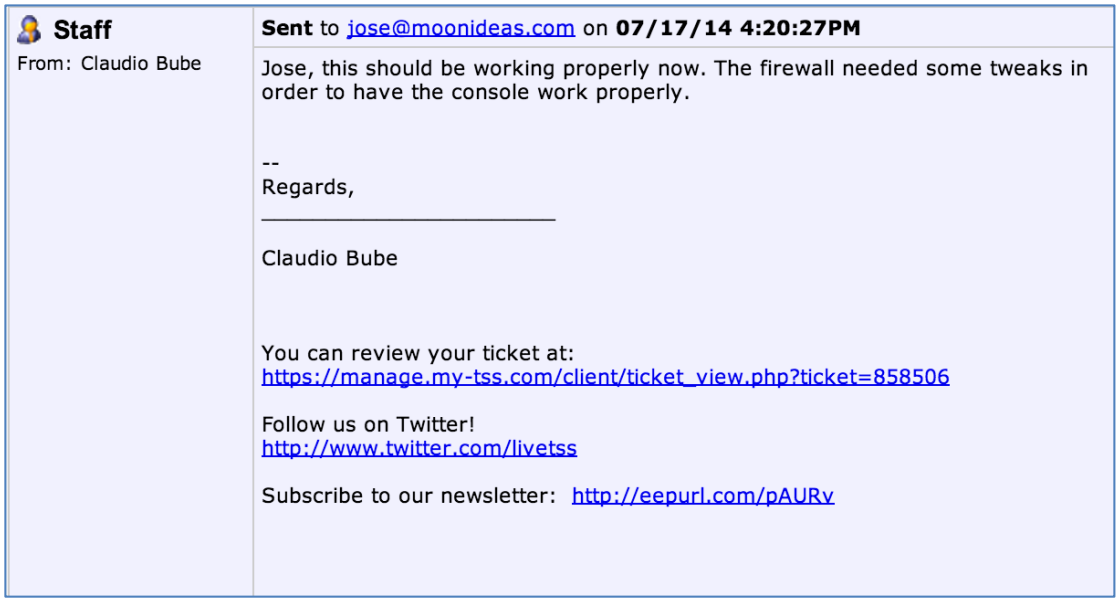


Figura 5 – Solicitud de asistencia (respuesta de equipo de soporte)

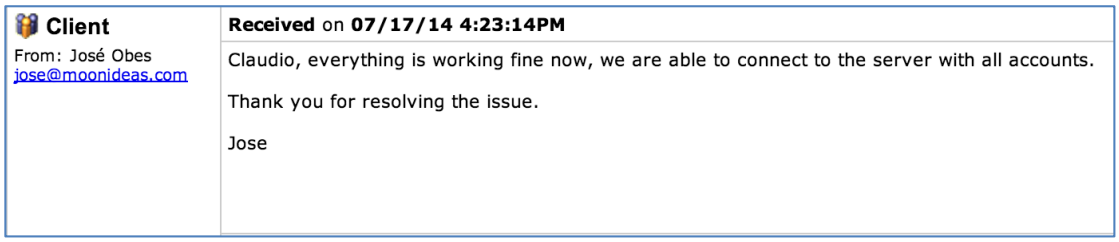


Figura 6 – Solicitud de asistencia (equipo de proyecto)

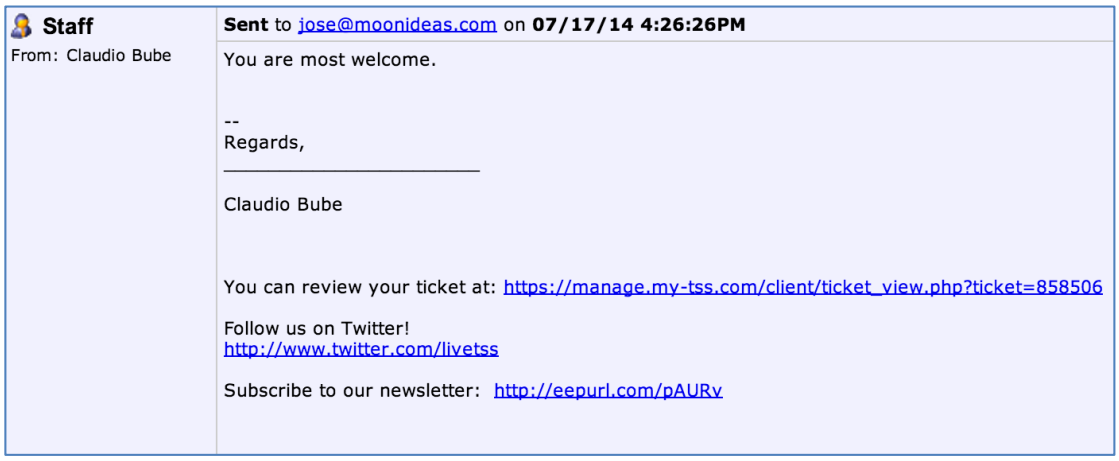


Figura 7 – Solicitud de asistencia (respuesta de equipo de soporte)

Anexo 11 : *Plugin* web: Detalle e instalación

11.1. El problema del *cross-domain*

Una componente especial que hubo que desarrollar para lograr la funcionalidad deseada en el producto fue el *plugin web*, el cual se instala en los sitios web de las distintas empresas que adquieren el sistema, para que sus clientes puedan realizar consultas directas desde los propios sitios web. A diferencia del resto de la solución, la cual está instalada bajo servidores o en ambientes propios del equipo de proyecto, el *plugin web* se instala en servidores ajenos.

El principal problema que trae el hecho de que el *plugin* esté en servidores ajenos, es que este necesita comunicarse con los servidores del equipo para solicitar información mediante AJAX, y esto es algo que todos los navegadores modernos prohíben estrictamente (no se puede realizar llamadas *cross-domain*, es decir, llamadas entre sitios web que están en distintos dominios).

Para resolver este problema, se tuvo que utilizar la tecnología CORS, la cual resulto ser una solución sencilla a nuestro problema. CORS (*Cross origin resource sharing*) es justamente lo que se necesitaba ya que provee un mecanismo que permite que muchos recursos (como archivos Javascript) que están en un sitio web soliciten información de otros dominios [16]. Para lograr esto, CORS agrega encabezados HTTP adicionales en los cuales viaja información acerca de que dominios tienen permiso para comunicarse con el servidor de origen. Los *web browsers* tiene la capacidad de interpretar dichos encabezados, y por lo tanto al saber el dominio desde donde sale la solicitud de contacto con el servidor, pueden determinar si se habilita la comunicación o no.

En nuestro caso, el servidor se programó de tal forma que sea sencillo agregar nuevos dominios a la lista de servicios con acceso. En esta lista aparecen los dominios de los sitios web de aquellas empresas que cuentan con la solución.

11.2. Instalación en un nuevo cliente

Gracias a la tecnología CORS, la instalación del *plugin web* en servidores ajenos resulto ser sencilla, al punto de que no es necesario que uno de los miembros del

equipo participe en la instalación, sino que cualquier empleado de una empresa cliente, encargado de administrar su sitio web lo puede hacer sin dificultad.

Por el momento, el *plugin* se le envía al usuario cuando este adquiere el producto. En el futuro también podrá ser descargado desde el sitio web de Moon Ideas. Su peso total es de 274 Kb y su contenido es el siguiente:

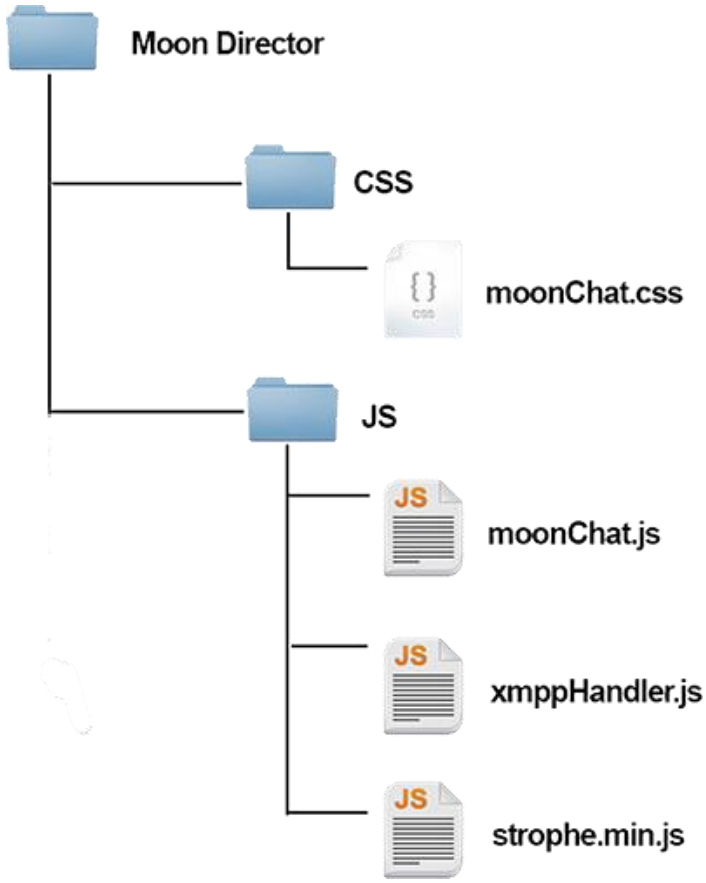


Figura 1 – Contenido del *plugin* web

Como se puede observar en la figura, el *plugin* está formado por tres archivos de Javascript (encargados de brindar la funcionalidad y generar el HTML necesario) y un archivo CSS (encargado de la presentación).

Los pasos que se deben tomar para la instalación completa son entonces:

- Adquirir la solución. Al adquirirla se debe especificar el dominio en el cual se va a instalar el *plugin*, para que este se agregue a la lista de dominios con acceso *cross domain*. Se le entregaran los archivos de contenido.

- Subir los archivos al servidor en donde se encuentra su sitio web, y hacer referencia a los tres archivos Javscript desde el encabezado del HTML (<head></head>). No es necesario hacer referencia al archivo CSS.

Una vez realizados los dos pasos podrá ver el *plugin* de chat en la parte inferior derecha del sitio web. Hay que destacar que se necesita tener JQuery para que este funcione.

Anexo 12: Detalle de protocolos utilizados

Debido a que el sistema construido tiene como eje central a la comunicación entre distintos dispositivos, uno de los temas centrales que se tuvieron que investigar fueron justamente los distintos mecanismos que sería necesario utilizar para realizar las distintas tareas de comunicación e intercambio de información. Para poder utilizar estos mecanismos se llegó a la conclusión de que se deberían utilizar tres protocolos de comunicación: HTTP/REST, XMPP y BOSH.

A continuación se detalla cada uno de estos protocolos, dando una breve descripción de los mismos y los usos que tuvieron en nuestro sistema:

12.1. HTTP y REST

Descripción

HTTP es un protocolo de capa de aplicación, el cual define como se comunican dos extremos (típicamente un cliente y un servidor) para intercambiar información y recursos. Es el estándar de comunicación en la internet.

REST es un estilo de arquitectura para designar aplicaciones con conectividad. La idea de REST es usar HTTP para realizar las llamadas entre los distintos dispositivos, en lugar de utilizar mecanismos más complicados como RPC o SOAP. Utiliza los comandos de HTTP (GET,POST,PUT,DELETE) para permitir que los dispositivos clientes realicen operaciones de creación, lectura, modificación y eliminación en los servidores.

Uso en el proyecto

En este proyecto se utilizó una API REST, es decir el servidor expone una serie de operaciones, a las cuales los clientes pueden acceder para obtener datos del propio servidor o modificarlos. Esta funcionalidad sirvió para comunicar tanto a los dispositivos móviles como a los servicios web (plugin y sitio de administración), con el servidor para intercambiar información.

Para el intercambio de la información en nuestro servicio REST se utilizó el formato JSON (Javascript Object Notation). JSON provee una manera de guardar información de un manera organizada y fácil de acceder. Las ventajas que encontramos al utilizar este formato es que nos permitió enviar los datos de un forma

liviana (prácticamente no agrega tamaño a los datos) y sencilla (el formato es leíble para seres humanos, y fue muy fácil interpretarlo desde los distintos lenguajes de programación utilizados).

12.2. XMPP

Descripción

XMPP (Extensible Markup Language) es un protocolo de comunicación para sistemas orientados a la mensajería basado en XML. Inicialmente llamado Jabber, el protocolo se creó para construir aplicaciones con funcionalidades de mensajería instantánea, información de presencia y mantenimiento de listas de contacto. Sin embargo, debido a su propiedad de ser extensible, se utilizó también para aplicaciones VoIP, video y transferencia de archivos, entre otras. A diferencia de otros protocolos, XMPP es un estándar abierto, por lo cual cualquiera puede agregar funcionalidad a la especificación original. Las extensiones más comunes son mantenidas Fundación de Estándares de XMPP [17] y están abiertas al público.

En la actualidad existen varias implementaciones de servidores XMPP y varias bibliotecas para crear aplicaciones clientes en todos los grandes lenguajes de programación. De esta forma, las aplicaciones cliente pueden enviar mensajes especializados llamados *stanza* al servidor, el cual sabe interpretarlos y realizar las acciones necesarias. Una *stanza* básica de envío de mensaje tiene el siguiente formato:

```
 <message
        to=’jabberId@example.com’
        from=”jabberId2@example.com”
        type=”chat">
        <body>Cuerpo del mensaje</body>
</message>
```

La lista completa de comandos que se pueden utilizar para este protocolo está publicada por la Fundación de Estándares de XMPP en su sitio web [17].

Uso en el proyecto

El uso de este protocolo fue fundamental en el proyecto, ya que brindó toda la funcionalidad central de mensajería instantánea. Como servidor se utilizó una implementación existente llamada Openfire (la creación de un servidor de este tipo es muy compleja y por si sola dedicaría un tiempo mayor al que se tiene para la creación de este proyecto). Para los clientes se crearon aplicaciones propias, utilizando las bibliotecas XMPPFramework para iOS, aSmack para Android y Strophe.js para Javascript. Se creó una versión extendida del protocolo, en la cual se agregaron tipos de mensajes propios, necesarios para la funcionalidad particular

requerida. Las aplicaciones clientes se diseñaron de manera que puedan interpretar este protocolo extendido.

12.3. BOSH

Descripción

BOSH (Bidirectional Streams Over Synchronous HTTP) es un protocolo de transporte que emula una conexión bidireccional entre dos entidades mediante múltiples solicitudes y respuestas HTTP síncronas. Sin necesidad de realizar polling. Esto permite que un cliente que no pueda mantener mantenga una conexión abierta con un servidor por un periodo extenso, pueda tener una alternativa que logra el mismo cometido.

Debido a que XMPP necesita mantener una conexión abierta, y los navegadores web no funcionan de esta forma (los navegadores webs hacen solicitudes a los servidores cuando son necesarias, sin mantener conexiones abiertas), es necesario utilizar BOSH para solucionar el problema. De hecho BOSH es la forma estándar de lograr una comunicación XMPP mediante HTTP [11] .

Uso en el proyecto

El uso de BOSH en el proyecto fue necesario e indispensable para los clientes web (*plugin web)* ya que al funcionar estos en navegadores web, como se explicó anteriormente, no es posible utilizar XMPP de manera directa. Utilizando este protocolo se pudo realizar la comunicación con el servidor XMPP y por lo tanto se logró la mensajería instantánea entre los clientes web y los clientes móviles.

Anexo 13: Selección de tecnologías

Tecnología	Función	Alternativas Consideradas	Razón de selección
PHP	Lenguaje de desarrollo para el lado servidor	Java,Python, Ruby, ASP.net, Perl	Experiencia de miembros del equipo y cliente con el lenguaje, gran comunidad, uso y ambientes de desarrollo gratuitos
Yii	Framework MVC para PHP	Cake PHP, Zend, CodeIgniter, Symfony	Experiencia de miembros del equipo
Apache	Servidor Web	IIS, Oracle Server, GlassFish	Experiencia miembros del equipo, fácil configuración, incluido en WAMP
MySql	Gestor de base de datos en el servidor web	PostgreSQL,Microsoft SQL Server, MariaDB	Experiencia de miembros del equipo, incluido en WAMP, gratuito
Javascript	Scripting web del lado cliente	CofeeScript, Dart,	Experiencia de miembros del equipo, tecnología estándar, funciona en todas las plataformas
JQuery	Presentación y animación web	Flash	Compatibilidad con todas las plataformas
Objective – C	Lenguaje de desarrollo para la plataforma iOS	-	Único lenguaje de desarrollo de aplicaciones nativas para iOS, soporte de Apple, Integración con Xcode, Experiencia de miembros de equipo

Java	Lenguaje de desarrollo para plataforma Android	-	Único lenguaje recomendado por Google
Openfire	Servidor XMPP	Prosody, Ejabberd, Tigase	Facilidad de configuración y manejo, interfaz gráfica de administración, cumple con todos los requerimientos del proyecto
NetBeans	IDE de desarrollo web (PHP, Javascript, JQery, HTML, CSS)	Eclipse, editores de texto, Dreamweaver, Aptana	Gratuito, Multiplataforma, experiencia de uso de miembros del equipo, soporte para los lenguajes y tecnologías web
Xcode	IDE de desarrollo para iOS	-	Herramienta creada por Apple específicamente para desarrollo iOS, integración con Objective-C y Cocoa Tocuh
ADT	IDE de desarrollo para Android	Android Studio	Herramienta recomendada por Google hasta el momento (Android Studio está en beta)

Figura 1 – Tecnologías seleccionadas.

Anexo 14: Herramientas de apoyo

Además de las herramientas utilizadas para el desarrollo de la solución, que se utilizaron varias herramientas para las distintas tareas de apoyo. A continuación se describen las herramientas utilizadas:

Version One

Tarea: Manejo de proyectos ágiles

Descripción: Version One es una herramienta que permite manejar proyectos que utiliza metodologías ágiles como scrum, Kanban y LEAN. Permite realizar todas las tareas necesarias a lo largo del ciclo de vida ágil de un proyecto. Es una herramienta web, accesible desde un navegador que permite que múltiples usuarios colaboren en conjunto para realizar tareas de planificación (creación de *product backlog*, *user stories*, *epics*, *themes*, *sprints*) y seguimiento (*storyboards*, *taskboards*, *testboards*, graficas de reporte). Cuenta con una versión gratuita limitada y una versión comercial.

Versions

Tarea: Manejo de versiones (cliente SVN)

Descripción: Versions es una herramienta de manejo de versiones (SVN) para la plataforma OSX. Permite acceder a repositorios SVN y realizar todas las operaciones necesarias a través de una interfaz gráfica intuitiva.

Visual SVN Server

Tarea: Manejo de versiones (servidor SVN)

Descripción: Visual SVN permite crear un servidor SVN, en el cual se pueden crear repositorios y cuentas de usuario para restringir el acceso a los repositorios. Permite que varios usuarios colaboren de manera conjunta ya sea en una red local o en un servidor en la red.

FileZilla

Tarea: Manejo de FTP

Descripción: FileZilla permite realizar tareas de FTP a través de una interfaz gráfica intuitiva. FTP es un protocolo que define como intercambiar archivos entre entidades. En este proyecto se utilizó específicamente para subir archivos al servidor web.

Adobe Photoshop

Tarea: Diseño Gráfico

Descripción: Photoshop es un software para la edición de graficos, el cual permite crear y manipular imágenes de distintos tipos de formatos, en una extensa cantidad de formas. Se utilizó en este proyecto para la creación de todas las imágenes necesarias para las distintas interfaces de usuario.

Adobe Illustrator

Tarea: Diseño gráfico

Descripción: Alternativa a Photoshop, con un enfoque distinto.

Remote desktop connection

Tarea: Conexiones de escritorio remotas

Descripción: Remote desktop connection es una aplicación creada por Microsoft para la plataforma OSX que permite acceder al escritorio de un ordenador de manera remota. Se utilizó en el proyecto para acceder y manipular las cuentas de usuario y los repositorios del servidor local de SVN.

SQLite Manager

Tarea: Administración de bases de datos SQLite

Descripción: SQLite Manager es una extensión del navegador Mozilla Firefox que permite interactuar con bases de datos de SQLite. Fue muy útil para el proyecto ya que tanto las bases de datos de la plataforma iOS como las de la plataforma Android utilizan SQLite.

CPanel

Tarea: Administración de sitios web

Descripción: CPanel es un panel de control que incluye una interfaz gráfica que permite administrar todos los aspectos de un sitio web o de un servidor (bases de datos, cuentas de email, DNS, entre otros aspectos). Se utilizó para manejar el servidor web y las aplicaciones que este contiene.

Google Drive

Tarea: Administración y sincronización de documentos

Descripción: Google drive es un sistema de almacenamiento y sincronización de archivos que permite que los usuarios almacenen archivos en la nube los compartan con otros usuarios y los manipulen de manera conjunta. Fue muy útil para mantener la documentación del proyecto, ya que permitió que los miembros del equipo trabajen de manera conjunta en un mismo documento, tengan acceso a toda la documentación estén donde estén y se aseguren de tener siempre un respaldo de todos los documentos.

Anexo 15 : Plan de riesgos

Análisis de riesgos

Valores de Impacto	Valores de Probabilidad de ocurrencia
0 - Ninguno	0.0 - No probable
1 - Marginal	0.2 - Poco probable
2 - Poco Importante	0.4 - Probable
3 - Importante (puede retrasar el proyecto)	0.6 - Muy probable
4 - Crítico (puede detener el proyecto)	0.8 - Altamente probable
5 - Catastrófico (fracaso del proyecto)	1.0 - Se convierte en problema

Figura 1 – Valores para el análisis de riesgos.

En el documento de Gestión de Riesgos, presentamos los diferentes riesgos que detectamos antes de comenzar el proyecto, clasificados por categorías. Previo al comienzo de cada *sprint*, una de las tareas que tiene el gerente del proyecto es relevar estos riesgos y detectar cuáles son los riesgos con mayor magnitud de impacto. Éste, se calcula multiplicando el valor de impacto asociado al riesgo por la probabilidad de que ese riesgo se materialice.

El valor va evolucionando con el proyecto y con las decisiones para mitigarlos, por eso la idea es hacerles un seguimiento y controlar los valores de cada riesgo para que luego, no se conviertan en un problema y puedan hacer que el proyecto fracase.

A continuación, se muestran las planillas de seguimiento de los Riesgos *sprint* a *sprint*. Para cada iteración, se presentan los Riesgos tenidos en cuenta y la prioridad de cada uno está dada la Magnitud asignada. La prioridad es alta, media o baja de acuerdo con las franjas presentadas en la escala.

Seguimiento de Riesgos

Puesta a punto

Riesgo	Estrategia	Acciones a realizar	Impacto	Probabilidad	Magnitud
R1 - Desvíos en el cronograma por falta de experiencia en la estimación de tareas y en gestión de proyectos	Prevenir	Sobreestimar el esfuerzo de las tareas al principio, luego comenzar a ajustar los valores con conocimientos de desempeño	3	0.6	1.8
R2 - Baja productividad por falta de motivación	Mitigar	Fomentar el buen ambiente de trabajo, ofrecer las tareas más amigables a quien se encuentra desmotivado	3	0.1	0.3
R3 - Insatisfacción del cliente con el producto	Prevenir	Validaciones por parte del *product owner* en las reuniones de retrospectiva de cada *sprint* y pruebas de aceptación al finalizar la construcción de cada módulo.	5	0.2	1.0
R4 - Insatisfacción de los usuarios finales con el producto	Asumir	Se deben registrar las inquietudes de los usuarios finales, para que sirvan como oportunidades de mejora y aspectos del producto que se deben mejorar.	5	0.2	1.0
R5 – Re-trabajo por comunicación ineficiente del equipo	Mitigar	Desarrollar procesos de comunicación mediante las herramientas tecnológicas. Explorar potencial de la herramienta de	3	0.6	1.8

		gestión seleccionada (VersionOne).			
R6 - Ineficiencia por falta de experiencia en la metodología *scrum*	Mitigar	Tratar de realizar las tareas aplicándose en la mayor medida posible a la metodología definida, para mejorar con la práctica. A su vez, la metodología debe ir mejorando en la medida que se detecten oportunidades de mejora.	3	0.8	2.4
R7 - No cumplir con el alcance del producto por retrasos en la construcción del *software*	Mitigar	Trabajar de a pares para agilizar tareas de construcción y que el equipo aprenda más. De todas formas se supone que se está en una curva de aprendizaje y que más adelante la productividad crecerá	5	0.2	1.0
R8 - Retrasos en la construcción de requerimientos complejos por falta de conocimientos avanzados en las tecnologías utilizadas	Asumir	Buscar información en internet o en bibliografía sobre diferentes formas de realizar las tareas y aplicarlas en el proyecto. También mediante pedido de ayuda dentro del equipo.	3	0	0
R9 - Retraso en tareas de gestión por ineficiencia en comunicación con actores externos al equipo	Mitigar	Mantener comunicación fluida tanto con el tutor como con el *product owner*, de forma de poder anticipar y planificar tareas para tener una mejor gestión.	3	0.7	2.1
R10 - Pérdida de interés en el proyecto por parte del cliente	Prevenir	Trabajar seriamente y mantener al cliente al tanto de todos los avances del proyecto, hacerlo partícipe mediante las validaciones, pruebas de aceptación y pidiendo opiniones personales en temas	5	0.1	0.5

		que corresponda.			
R11 - Pérdida de datos del proyecto (documentación, código, etc.)	Mitigar	Realizar respaldos periódicos y semanales	4	0.2	0.8
R12 - Incendio, robo o cualquier catástrofe con el servidor de aplicaciones y datos	Mitigar	Mantener respaldos descentralizados	5	0.1	0.5

Figura 2 – Análisis de riesgos para el *sprint* puesta a punto.

Sprint 1

Riesgo	Estrategia	Acciones a realizar	Impacto	Probabilidad	Magnitud
R1 - Desvíos en el cronograma por falta de experiencia en la estimación de tareas y en gestión de proyectos	Prevenir	Sobreestimar el esfuerzo de las tareas al principio, luego comenzar a ajustar los valores con conocimientos de desempeño	3	0.8	2.4

R2 - Baja productividad por falta de motivación	Mitigar	Fomentar el buen ambiente de trabajo, ofrecer las tareas más amigables a quien se encuentra desmotivado	3	0.1	0.3
R3 - Insatisfacción del cliente con el producto	Prevenir	Validaciones por parte del *product owner* en las reuniones de retrospectiva de cada *sprint* y pruebas de aceptación al finalizar la construcción de cada módulo.	5	0.2	1.0
R4 - Insatisfacción de los usuarios finales con el producto	Asumir	Se deben registrar las inquietudes de los usuarios finales, para que sirvan como oportunidades de mejora y aspectos del producto que se deben mejorar.	5	0.2	1.0
R5 – Re-trabajo por comunicación ineficiente del equipo	Mitigar	Desarrollar procesos de comunicación mediante las herramientas tecnológicas. Explorar potencial de la herramienta de gestión seleccionada (Version One).	3	0.6	1.8
R6 - Ineficiencia por falta de experiencia en la metodología *scrum*	Mitigar	Tratar de realizar las tareas aplicándose en la mayor medida posible a la metodología definida, para mejorar con la práctica. A su vez, la metodología debe ir mejorando en la medida que se detecten oportunidades de mejora.	3	0.8	2.4
R7 - No cumplir con el alcance del producto por retrasos en la construcción del software	Mitigar	Trabajar de a pares para agilizar tareas de construcción y que el equipo aprenda más. De todas formas se supone que se está en una curva de aprendizaje y que más adelante la productividad crecerá	5	0.2	1.0
R8 - Retrasos en la	Asumir	Buscar información en internet o en	3	0.2	0.6

construcción de requerimientos complejos por falta de conocimientos avanzados en las tecnologías utilizadas		bibliografía sobre diferentes formas de realizar las tareas y aplicarlas en el proyecto. También mediante pedido de ayuda dentro del equipo.			
R9 - Retraso en tareas de gestión por ineficiencia en comunicación con actores externos al equipo	Mitigar	Mantener comunicación fluida tanto con el tutor como con el *product owner*, de forma de poder anticipar y planificar tareas para tener una mejor gestión.	3	0.5	1.5
R10 - Pérdida de interés en el proyecto por parte del cliente	Prevenir	Trabajar seriamente y mantener al cliente al tanto de todos los avances del proyecto, hacerlo partícipe mediante las validaciones, pruebas de aceptación y pidiendo opiniones personales en temas que corresponda.	5	0.1	0.5
R11 - Pérdida de datos del proyecto (documentación, código, etc.)	Mitigar	Realizar respaldos periódicos y semanales	4	0.2	0.8
R12 - Incendio, robo o cualquier catástrofe con el servidor de aplicaciones y datos	Mitigar	Mantener respaldos descentralizados	5	0.1	0.5

Figura 3 – Análisis de riesgos para el *sprint* 1.

Sprint 2

Riesgo	Estrategia	Acciones a realizar	Impacto	Probabilidad	Magnitud
R1 - Desvíos en el cronograma por falta de experiencia en la estimación de tareas y en gestión de proyectos	Prevenir	Sobreestimar el esfuerzo de las tareas al principio, luego comenzar a ajustar los valores con conocimientos de desempeño	3	0.8	2.4
R2 - Baja productividad por falta de motivación	Mitigar	Fomentar el buen ambiente de trabajo, ofrecer las tareas más amigables a quien se encuentra desmotivado	3	0.1	0.3
R3 - Insatisfacción del cliente con el producto	Prevenir	Validaciones por parte del *product owner* en las reuniones de retrospectiva de cada *sprint* y pruebas de aceptación al finalizar la construcción de cada módulo.	5	0.2	1.0
R4 - Insatisfacción de los usuarios finales con el producto	Asumir	Se deben registrar las inquietudes de los usuarios finales, para que sirvan como oportunidades de mejora y aspectos del producto que se deben mejorar.	5	0.2	1.0
R5 – Re-trabajo por comunicación ineficiente del equipo	Mitigar	Desarrollar procesos de comunicación mediante las herramientas tecnológicas. Explorar potencial de la herramienta de gestión seleccionada (Version One).	3	0.6	1.8
R6 - Ineficiencia por falta de experiencia en la metodología	Mitigar	Tratar de realizar las tareas aplicándose en la mayor medida posible a la	3	0.9	2.7

scrum		metodología definida, para mejorar con la práctica. A su vez, la metodología debe ir mejorando en la medida que se detecten oportunidades de mejora.			
R7 - No cumplir con el alcance del producto por retrasos en la construcción del software	Mitigar	Trabajar de a pares para agilizar tareas de construcción y que el equipo aprenda más. De todas formas se supone que se está en una curva de aprendizaje y que más adelante la productividad crecerá.	5	0.3	1.5
R8 - Retrasos en la construcción de requerimientos complejos por falta de conocimientos avanzados en las tecnologías utilizadas	Asumir	Buscar información en internet o en bibliografía sobre diferentes formas de realizar las tareas y aplicarlas en el proyecto. También mediante pedido de ayuda dentro del equipo.	3	0.6	1.8
R9 - Retraso en tareas de gestión por ineficiencia en comunicación con actores externos al equipo	Mitigar	Mantener comunicación fluida tanto con el tutor como con el *product owner*, de forma de poder anticipar y planificar tareas para tener una mejor gestión.	3	0.4	1.2
R10 - Pérdida de interés en el proyecto por parte del cliente	Prevenir	Trabajar seriamente y mantener al cliente al tanto de todos los avances del proyecto, hacerlo partícipe mediante las validaciones, pruebas de aceptación y pidiendo opiniones personales en temas que corresponda.	5	0.1	0.5
R11 - Pérdida de datos del proyecto (documentación,	Mitigar	Realizar respaldos periódicos y semanales.	4	0.2	0.8

código, etc.)					
R12 - Incendio, robo o cualquier catástrofe con el servidor de aplicaciones y datos	Mitigar	Mantener respaldos descentralizados.	5	0.1	0.5
R13 - Retraso por detectar fallas en forma tardía por no utilizar casos, ni datos de prueba para realizar los diferentes tipos de pruebas	Asumir	Sobreestimar el esfuerzo de las tareas al principio, luego comenzar a ajustar los valores con conocimientos de desempeño.	3	0.3	0.9

Figura 4 – Análisis de riesgos para el *sprint* 2.

Sprint 3

Riesgo	Estrategia	Acciones a realizar	Impacto	Probabilidad	Magnitud
R1 - Desvíos en el cronograma por falta de experiencia en la estimación de tareas y en gestión de proyectos	Prevenir	Sobreestimar el esfuerzo de las tareas al principio, luego comenzar a ajustar los valores con conocimientos de desempeño.	3	0.8	2.4
R2 - Baja productividad por falta de motivación	Mitigar	Fomentar el buen ambiente de trabajo, ofrecer las tareas más amigables a quien se encuentra desmotivado.	3	0.1	0.3

R3 - Insatisfacción del cliente con el producto	Prevenir	Validaciones por parte del *product owner* en las reuniones de retrospectiva de cada *sprint* y pruebas de aceptación al finalizar la construcción de cada módulo.	5	0.2	1.0
R4 - Insatisfacción de los usuarios finales con el producto	Asumir	Se deben registrar las inquietudes de los usuarios finales, para que sirvan como oportunidades de mejora y aspectos del producto que se deben mejorar.	5	0.2	1.0
R5 – Re-trabajo por comunicación ineficiente del equipo	Mitigar	Desarrollar procesos de comunicación mediante las herramientas tecnológicas. Explorar potencial de la herramienta de gestión seleccionada (Version One).	3	0.6	1.8
R6 - Ineficiencia por falta de experiencia en la metodología *Scrum*	Mitigar	Tratar de realizar las tareas aplicándose en la mayor medida posible a la metodología definida, para mejorar con la práctica. A su vez, la metodología debe ir mejorando en la medida que se detecten oportunidades de mejora.	3	1.0	3.0
R7 - No cumplir con el alcance del producto por retrasos en la construcción del software	Mitigar	Trabajar de a pares para agilizar tareas de construcción y que el equipo aprenda más. De todas formas se supone que se está en una curva de aprendizaje y que más adelante la productividad crecerá.	5	0.3	1.5
R8 - Retrasos en la construcción de requerimientos complejos por falta de conocimientos	Asumir	Buscar información en internet o en bibliografía sobre diferentes formas de realizar las tareas y aplicarlas en el proyecto. También mediante pedido de	3	0.8	2.4

avanzados en las tecnologías utilizadas		ayuda dentro del equipo.			
R9 - Retraso en tareas de gestión por ineficiencia en comunicación con actores externos al equipo	Mitigar	Mantener comunicación fluida tanto con el tutor como con el *product owner*, de forma de poder anticipar y planificar tareas para tener una mejor gestión.	3	0.3	0.9
R10 - Pérdida de interés en el proyecto por parte del cliente	Prevenir	Trabajar seriamente y mantener al cliente al tanto de todos los avances del proyecto, hacerlo partícipe mediante las validaciones, pruebas de aceptación y pidiendo opiniones personales en temas que corresponda.	5	0.1	0.5
R11 - Pérdida de datos del proyecto (documentación, código, etc.)	Mitigar	Realizar respaldos periódicos y semanales.	4	0.2	0.8
R12 - Incendio, robo o cualquier catástrofe con el servidor de aplicaciones y datos	Mitigar	Mantener respaldos descentralizados.	5	0.1	0.5
R13 - Retraso por detectar fallas en forma tardía por no utilizar casos, ni datos de prueba para realizar los diferentes tipos de pruebas	Asumir	Realizar pruebas y documentación de hallazgos según proceso de pruebas definido. Al ser un equipo chico, el código es conocido por todos y de encontrar fallas a la hora de integrar o realizar	3	0.4	1.2

		pruebas del sistema, la localización no sería tan compleja.			

Figura 5 – Análisis de riesgos para el *sprint* 3.

Sprint 4

Riesgo	Estrategia	Acciones a realizar	Impacto	Probabilidad	Magnitud
R1 - Desvíos en el cronograma por falta de experiencia en la estimación de tareas y en gestión de proyectos	Prevenir	Sobreestimar el esfuerzo de las tareas al principio, luego comenzar a ajustar los valores con conocimientos de desempeño.	3	0.8	2.4
R2 - Baja productividad por falta de motivación	Mitigar	Fomentar el buen ambiente de trabajo, ofrecer las tareas más amigables a quien se encuentra desmotivado.	3	0.1	0.3
R3 - Insatisfacción del cliente con el producto	Prevenir	Validaciones por parte del *product owner* en las reuniones de retrospectiva de cada *sprint* y pruebas de aceptación al finalizar la construcción de cada módulo.	5	0.2	1.0
R4 - Insatisfacción de los usuarios finales con el producto	Asumir	Se deben registrar las inquietudes de los usuarios finales, para que sirvan como oportunidades de mejora y aspectos del producto que se deben mejorar.	5	0.2	1.0
R5 – Re-trabajo por comunicación ineficiente del equipo	Mitigar	Desarrollar procesos de comunicación mediante las herramientas tecnológicas. Explorar potencial de la herramienta de gestión seleccionada (Version One).	3	0.5	1.5
R6 - Ineficiencia por falta de experiencia en la metodología	Mitigar	Tratar de realizar las tareas aplicándose en la mayor medida posible a la	3	1.0	3.0

Scrum		metodología definida, para mejorar con la práctica. A su vez, la metodología debe ir mejorando en la medida que se detecten oportunidades de mejora.			
R7 - No cumplir con el alcance del producto por retrasos en la construcción del software	Mitigar	Trabajar de a pares para agilizar tareas de construcción y que el equipo aprenda más. De todas formas se supone que se está en una curva de aprendizaje y que más adelante la productividad crecerá.	5	0.4	2.0
R8 - Retrasos en la construcción de requerimientos complejos por falta de conocimientos avanzados en las tecnologías utilizadas	Mitigar	Buscar información en internet o en bibliografía sobre diferentes formas de realizar las tareas y aplicarlas en el proyecto. También mediante pedido de ayuda dentro del equipo.	3	0.9	2.7
R9 - Retraso en tareas de gestión por ineficiencia en comunicación con actores externos al equipo	Mitigar	Mantener comunicación fluida tanto con el tutor como con el *product owner*, de forma de poder anticipar y planificar tareas para tener una mejor gestión.	3	0.3	0.9
R10 - Pérdida de interés en el proyecto por parte del cliente	Prevenir	Trabajar seriamente y mantener al cliente al tanto de todos los avances del proyecto, hacerlo partícipe mediante las validaciones, pruebas de aceptación y pidiendo opiniones personales en temas que corresponda.	5	0.1	0.5
R11 - Pérdida de datos del proyecto (documentación,	Mitigar	Realizar respaldos periódicos y semanales.	4	0.2	0.8

código, etc.)					
R12 - Incendio, robo o cualquier catástrofe con el servidor de aplicaciones y datos	Mitigar	Mantener respaldos descentralizados.	5	0.1	0.5
R13 - Retraso por detectar fallas en forma tardía por no utilizar casos, ni datos de prueba para realizar los diferentes tipos de pruebas	Asumir	Realizar pruebas y documentación de hallazgos según proceso de pruebas definido. Al ser un equipo chico, el código es conocido por todos y de encontrar fallas a la hora de integrar o realizar pruebas del sistema, la localización no sería tan compleja.	3	0.6	1.8
R14 - Retrasos en la construcción de funcionalidades por tareas no previstas/estimadas o cambios surgidos a mitad de *sprint*	Mitigar	Si la tarea o cambio requiere un esfuerzo menor a 8 horas, el cambio se incluye dentro del mismo *sprint*. En caso que el esfuerzo requerido sea mayor, se aborta el *sprint* y se planifica uno nuevo.	3	0.5	1.5

Figura 6 – Análisis de riesgos para el *sprint* 4.

Sprint 5

Riesgo	Estrategia	Acciones a realizar	Impacto	Probabilidad	Magnitud
R1 - Desvíos en el cronograma por falta de experiencia en la estimación de tareas y en gestión de proyectos	Prevenir	Sobreestimar el esfuerzo de las tareas al principio, luego comenzar a ajustar los valores con conocimientos de desempeño.	3	0.8	2.4
R2 - Baja productividad por falta de motivación	Mitigar	Fomentar el buen ambiente de trabajo, ofrecer las tareas más amigables a quien se encuentra desmotivado.	3	0.1	0.3
R3 - Insatisfacción del cliente con el producto	Prevenir	Validaciones por parte del *product owner* en las reuniones de retrospectiva de cada *sprint* y pruebas de aceptación al finalizar la construcción de cada módulo.	5	0.2	1.0
R4 - Insatisfacción de los usuarios finales con el producto	Asumir	Se deben registrar las inquietudes de los usuarios finales, para que sirvan como oportunidades de mejora y aspectos del producto que se deben mejorar.	5	0.2	1.0
R5 – Re-trabajo por comunicación ineficiente del equipo	Mitigar	Desarrollar procesos de comunicación mediante las herramientas tecnológicas. Explorar potencial de la herramienta de gestión seleccionada (Version One).	3	0.4	1.2

R6 - Ineficiencia por falta de experiencia en la metodología *Scrum*	Mitigar	Tratar de realizar las tareas aplicándose en la mayor medida posible a la metodología definida, para mejorar con la práctica. A su vez, la metodología debe ir mejorando en la medida que se detecten oportunidades de mejora.	3	0.7	2.1
R7 - No cumplir con el alcance del producto por retrasos en la construcción del software	Mitigar	Trabajar de a pares para agilizar tareas de construcción y que el equipo aprenda más. De todas formas se supone que se está en una curva de aprendizaje y que más adelante la productividad crecerá.	5	0.4	2.0
R8 - Retrasos en la construcción de requerimientos complejos por falta de conocimientos avanzados en las tecnologías utilizadas	Asumir	Buscar información en internet o en bibliografía sobre diferentes formas de realizar las tareas y aplicarlas en el proyecto. También mediante pedido de ayuda dentro del equipo.	3	0.9	2.7
R9 - Retraso en tareas de gestión por ineficiencia en comunicación con actores externos al equipo	Mitigar	Mantener comunicación fluida tanto con el tutor como con el *product owner*, de forma de poder anticipar y planificar tareas para tener una mejor gestión.	3	0.3	0.9
R10 - Pérdida de interés en el proyecto por parte del cliente	Prevenir	Trabajar seriamente y mantener al cliente al tanto de todos los avances del proyecto, hacerlo partícipe mediante las validaciones, pruebas de aceptación y pidiendo opiniones personales en temas que corresponda.	5	0.1	0.5

R11 - Pérdida de datos del proyecto (documentación, código, etc.)	Mitigar	Realizar respaldos periódicos y semanales.	4	0.2	0.8
R12 - Incendio, robo o cualquier catástrofe con el servidor de aplicaciones y datos	Mitigar	Mantener respaldos descentralizados.	5	0.1	0.5
R13 - Retraso por detectar fallas en forma tardía por no utilizar casos, ni datos de prueba para realizar los diferentes tipos de pruebas	Asumir	Realizar pruebas y documentación de hallazgos según proceso de pruebas definido. Al ser un equipo chico, el código es conocido por todos y de encontrar fallas a la hora de integrar o realizar pruebas del sistema, la localización no sería tan compleja.	3	0.7	2.1
R14 - Retrasos en la construcción de producto por tareas no previstas/estimadas o cambios surgidos a mitad de *sprint*	Mitigar	Si la tarea o cambio requiere un esfuerzo menor a 8 horas, el cambio se incluye dentro del mismo *sprint*. En caso que el esfuerzo requerido sea mayor, se aborta el *sprint* y se planifica uno nuevo.	3	1.0	3.0
R15 - Retrasos en construcción de producto por contar con demasiadas correcciones para hacer.	Mitigar	Mantener al tanto al *product owner* de que es lo que se está haciendo. Pedirle opinión y aprobación de lo realizado.	3	0.5	1.5

Pruebas de aceptación con *product owner* demasiado esporádicas.					

Figura 7 – Análisis de riesgos para el *sprint* 5.

Sprint 6

Riesgo	Estrategia	Acciones a realizar	Impacto	Probabilidad	Magnitud
R1 - Desvíos en el cronograma por falta de experiencia en la estimación de tareas y en gestión de proyectos	Prevenir	Sobreestimar el esfuerzo de las tareas al principio, luego comenzar a ajustar los valores con conocimientos de desempeño.	3	0.8	2.4
R2 - Baja productividad por falta de motivación	Mitigar	Fomentar el buen ambiente de trabajo, ofrecer las tareas más amigables a quien se encuentra desmotivado.	3	0.1	0.3
R3 - Insatisfacción del cliente con el producto	Prevenir	Validaciones por parte del *product owner* en las reuniones de retrospectiva de cada *sprint* y pruebas de aceptación al finalizar la construcción de cada módulo.	5	0.3	1.5
R4 - Insatisfacción de los usuarios finales con el producto	Asumir	Se deben registrar las inquietudes de los usuarios finales, para que sirvan como oportunidades de mejora y	5	0.5	2.5

		aspectos del producto que se deben mejorar.			
R5 – Re-trabajo por comunicación ineficiente del equipo	Mitigar	Desarrollar procesos de comunicación mediante las herramientas tecnológicas. Explorar potencial de la herramienta de gestión seleccionada (Version One).	3	0.4	1.2
R6 - Ineficiencia por falta de experiencia en la metodología *Scrum*	Mitigar	Tratar de realizar las tareas aplicándose en la mayor medida posible a la metodología definida, para mejorar con la práctica. A su vez, la metodología debe ir mejorando en la medida que se detecten oportunidades de mejora.	3	0.6	1.8
R7 - No cumplir con el alcance del producto por retrasos en la construcción del software	Mitigar	Trabajar de a pares para agilizar tareas de construcción y que el equipo aprenda más. Pedir licencia en el trabajo para tener mayor capacidad de esfuerzo.	5	0.6	3.0
R8 - Retrasos en la construcción de requerimientos complejos por falta de conocimientos avanzados en las tecnologías utilizadas	Mitigar	Buscar información internet sobre diferentes formas de realizar las tareas, asesorarse con gente conocida y experimentada. También aplicar programación de a pares.	3	1.0	3.0
R9 - Retraso en tareas de gestión por ineficiencia en comunicación con actores	Mitigar	Mantener comunicación fluida tanto con el tutor como con el *product owner*, de forma de poder anticipar y planificar	3	0.2	0.6

externos al equipo		tareas para tener una mejor gestión.			
R10 - Pérdida de interés en el proyecto por parte del cliente	Prevenir	Trabajar seriamente y mantener al cliente al tanto de todos los avances del proyecto, hacerlo partícipe mediante las validaciones, pruebas de aceptación y pidiendo opiniones personales en temas que corresponda.	5	0.1	0.5
R11 - Pérdida de datos del proyecto (documentación, código, etc.)	Mitigar	Realizar respaldos periódicos y semanales.	4	0.2	0.8
R12 - Incendio, robo o cualquier catástrofe con el servidor de aplicaciones y datos	Mitigar	Mantener respaldos descentralizados.	5	0.1	0.5
R13 - Retraso por detectar fallas en forma tardía por no utilizar casos, ni datos de prueba para realizar los diferentes tipos de pruebas	Asumir	Realizar pruebas y documentación de hallazgos según proceso de pruebas definido. Al ser un equipo chico, el código es conocido por todos y de encontrar fallas a la hora de integrar o realizar pruebas del sistema, la localización no sería tan compleja.	3	0.8	2.4
R14 - Retrasos en la construcción de funcionalidades por tareas no previstas/estimadas o cambios surgidos a mitad de *sprint*	Mitigar	Si la tarea o cambio requiere un esfuerzo menor a 8 horas, el cambio se incluye dentro del mismo *sprint*. En caso que el esfuerzo requerido sea mayor, se aborta el *sprint* y se planifica uno nuevo.	3	0.5	1.5

R15 - Retrasos en construcción de producto por contar con demasiadas correcciones para hacer. Pruebas de aceptación con *product owner* demasiado esporádicas.	Mitigar	Mantener al tanto al *product owner* de que es lo que se está haciendo.	3	0.5	1.5
R16 - Falta de tiempo para corregir la aplicación en caso de que las pruebas alfa generen incidentes.	Asumir	Acordar con el cliente un listado de incidentes prioritarios que se corrijan para la entrega del proyecto, los incidentes excluidos de esta lista se corregirán en versiones posteriores del sistema.	5	0.5	2.5

Figura 8 – Análisis de riesgos para el *sprint* 6.

Sprint 7

Riesgo	Estrategia	Acciones a realizar	Impacto	Probabilidad	Magnitud
R1 - Desvíos en el cronograma por falta de experiencia en la estimación de tareas y en gestión de proyectos	Prevenir	Sobreestimar el esfuerzo de las tareas al principio, luego comenzar a ajustar los valores con conocimientos de desempeño.	3	0.8	2.4
R2 - Baja productividad por	Mitigar	Fomentar el buen ambiente de trabajo,	3	0.1	0.3

falta de motivación		ofrecer las tareas más amigables a quien se encuentra desmotivado.			
R3 - Insatisfacción del cliente con el producto	Prevenir	Validaciones por parte del *product owner* en las reuniones de retrospectiva de cada *sprint* y pruebas de aceptación al finalizar la construcción de cada módulo. Mantener contacto fluido para disipar dudas.	5	0.4	2.0
R4 - Insatisfacción de los usuarios finales con el producto	Asumir	Se deben registrar las inquietudes de los usuarios finales, para que sirvan como oportunidades de mejora y aspectos del producto que se deben mejorar.	5	0.5	2.5
R5 – Re-trabajo por comunicación ineficiente del equipo	Mitigar	Desarrollar procesos de comunicación mediante las herramientas que nos brinda la tecnología hoy en día. Explorar potencialidades de la herramienta de gestión seleccionada (Version One).	3	0.3	0.9
R6 - Ineficiencia por falta de experiencia en la metodología *Scrum*	Mitigar	Tratar de realizar las tareas aplicándose en la mayor medida posible a la metodología definida, para mejorar con la práctica. A su vez la metodología debe ir mejorando en la medida que se detecten oportunidades de mejora.	3	0.6	1.8
R7 - No cumplir con el alcance del producto por retrasos en la	Mitigar	Trabajar de a pares para agilizar tareas de construcción y que el equipo	5	0.8	4.0

construcción del software		aprenda más. Pedir licencia en el trabajo para tener mayor capacidad de esfuerzo.			
R8 - Retrasos en la construcción de requerimientos complejos por falta de conocimientos avanzados en las tecnologías utilizadas	Mitigar	Buscar información internet sobre diferentes formas de realizar las tareas, asesorarse con gente conocida y experimentada. También aplicar programación de a pares.	3	1.0	3.0
R9 - Retraso en tareas de gestión por ineficiencia en comunicación con actores externos al equipo	Mitigar	Mantener comunicación fluida tanto con el tutor como con el *product owner*, de forma de poder anticipar y planificar tareas para tener una mejor gestión.	3	0.3	0.9
R10 - Pérdida de interés en el proyecto por parte del cliente	Prevenir	Trabajar seriamente y mantener al cliente al tanto de todos los avances del proyecto, hacerlo partícipe mediante las validaciones, pruebas de aceptación y pidiendo opiniones personales en temas que corresponda.	5	0.1	0.5
R11 - Pérdida de datos del proyecto (documentación, código, etc.)	Mitigar	Realizar respaldos periódicos y semanales	4	0.2	0.8
R12 - Incendio, robo o cualquier catástrofe con el servidor de aplicaciones y datos	Mitigar	Mantener respaldos descentralizados.	5	0.1	0.5
R13 - Retraso por	Asumir	Realizar pruebas y documentación de	3	0.8	2.4

detectarfallas en forma tardía por no utilizar casos, ni datos de prueba para realizar los diferentes tipos de pruebas		hallazgos según proceso de pruebas definido. Al ser un equipo chico, el código es conocido por todos y de encontrar fallas a la hora de integrar o realizar pruebas del sistema, la localización no sería tan compleja.			
R14 - Retrasos en la construcción de funcionalidades por tareas no previstas/estimadas o cambios surgidos a mitad de *sprint*	Mitigar	Si la tarea o cambio requiere un esfuerzo menor a 8 horas, el cambio se incluye dentro del mismo *sprint*. En caso que el esfuerzo requerido sea mayor, se aborta el *sprint* y se planifica uno nuevo.	3	0.5	1.5
R15 - Retrasos en construcción de producto por contar con demasiadas correcciones para hacer. Pruebas de aceptación con *product owner* demasiado esporádicas.	Mitigar	Mantener al tanto al *product owner* de que es lo que se está haciendo.	3	0.6	1.8
R16 - Falta de tiempo para corregir la aplicación en caso de que las pruebas alfa generen incidentes.	Asumir	Acordar con el cliente un listado de incidentes prioritarios que se corrijan para la entrega del proyecto, los incidentes excluidos de esta lista se corregirán en versiones posteriores del sistema.	5	0.5	2.5

Figura 9 – Análisis de riesgos para el *sprint* 7.

Sprint 8

Riesgo	Estrategia	Acciones a realizar	Impacto	Probabilidad	Magnitud
R1 - Desvíos en el cronograma por falta de experiencia en la estimación de tareas y en gestión de proyectos	Mitigar	Modificar la forma de estimar, hasta ahora no habíamos tenido en cuenta el valor de nuestro SP promedio. Para tratar de mitigar este problema de acá al fin del proyecto vamos a estimar tomando en cuenta el valor SP del último *sprint*. En este caso vamos para arrancar vamos a hacer un promedio con los valores de todos los *sprints* anteriores.	3	0.9	2.7
R2 - Baja productividad por falta de motivación	Mitigar	Fomentar el buen ambiente de trabajo, ofrecer las tareas más amigables a quien se encuentra desmotivado.	3	0.1	0.3
R3 - Insatisfacción del cliente con el producto	Prevenir	Validaciones por parte del *product owner* en las reuniones de retrospectiva de cada *sprint* y pruebas de aceptación al finalizar la construcción de cada módulo. Mantener contacto fluido para disipar dudas.	5	0.4	2.0
R4 - Insatisfacción de los usuarios finales con el producto	Asumir	Se deben registrar las inquietudes de los usuarios finales, para que sirvan como oportunidades de mejora y aspectos del producto que se deben mejorar.	5	0.5	2.5

R5 – Re-trabajo por comunicación ineficiente del equipo	Mitigar	Desarrollar procesos de comunicación mediante las herramientas que nos brinda la tecnología hoy en día. Explorar potencialidades de la herramienta de gestión seleccionada (Version One).	3	0.3	0.9
R6 - Ineficiencia por falta de experiencia en la metodología *Scrum*	Mitigar	Tratar de realizar las tareas aplicándose en la mayor medida posible a la metodología definida, para mejorar con la práctica. A su vez la metodología debe ir mejorando en la medida que se detecten oportunidades de mejora.	3	0.6	1.8
R7 - No cumplir con el alcance del producto por retrasos en la construcción del software	Mitigar	Trabajar de a pares para agilizar tareas de construcción y que el equipo aprenda más. Pedir licencia en el trabajo para tener mayor capacidad de esfuerzo.	5	0.8	4.0
R8 - Retrasos en la construcción de requerimientos complejos por falta de conocimientos avanzados en las tecnologías utilizadas	Asumir	Buscar información internet sobre diferentes formas de realizar las tareas, asesorarse con gente conocida y experimentada y aplicarlas en el proyecto. También aplicar programación de a pares.	3	1.0	3.0
R9 - Retraso en tareas de gestión por ineficiencia en comunicación con actores externos al equipo	Mitigar	Mantener comunicación fluida tanto con el tutor como con el *product owner*, de forma de poder anticipar y planificar tareas para tener una mejor gestión.	3	0.3	0.9

R10 - Pérdida de interés en el proyecto por parte del cliente	Prevenir	Trabajar seriamente y mantener al cliente al tanto de todos los avances del proyecto, hacerlo partícipe mediante las validaciones, pruebas de aceptación y pidiendo opiniones personales en temas que corresponda.	5	0.1	0.5
R11 - Pérdida de datos del proyecto (documentación, código, etc.)	Mitigar	Realizar respaldos periódicos y semanales.	4	0.2	0.8
R12 - Incendio, robo o cualquier catástrofe con el servidor de aplicaciones y datos	Mitigar	Mantener respaldos descentralizados.	5	0.1	0.5
R13 - Retraso por detectar fallas en forma tardía por no utilizar casos, ni datos de prueba para realizar los diferentes tipos de pruebas	Asumir	Realizar pruebas y documentación de hallazgos según proceso de pruebas definido. Al ser un equipo chico, el código es conocido por todos y de encontrar fallas a la hora de integrar o realizar pruebas del sistema, la localización no sería tan compleja.	3	0.8	2.4
R14 - Retrasos en la construcción de funcionalidades por tareas no previstas/estimadas o cambios surgidos a mitad de *sprint*	Mitigar	Si la tarea o cambio requiere un esfuerzo menor a 8 horas, el cambio se incluye dentro del mismo *sprint*. En caso que el esfuerzo requerido sea mayor, se aborta el *sprint* y se planifica uno nuevo.	3	0.5	1.5
R15 - Retrasos en	Mitigar	Mantener al tanto al *product owner* de	3	0.7	2.1

construcción de producto por contar con demasiadas correcciones para hacer. Pruebas de aceptación con *product owner* demasiado esporádicas.		que es lo que se está haciendo.			
R16 - Falta de tiempo para corregir la aplicación en caso de que las pruebas alfa generen incidentes.	Asumir	Registrar los incidentes registrados por los usuarios, analizarlos y ponerlos como mejoras necesarias para el obtener la primera versión del producto.	5	1.0	5.0

Figura 10 – Análisis de riesgos para el *sprint* 8.

Sprint 9

Riesgo	Estrategia	Acciones a realizar	Impacto	Probabilidad	Magnitud
R1 - Desvíos en el cronograma por falta de experiencia en la estimación de tareas y en gestión de proyectos	Mitigar	Modificar la forma de estimar, hasta ahora no habíamos tenido en cuenta el valor de nuestro SP promedio. Para tratar de mitigar este problema de acá al fin del proyecto vamos a estimar tomando en cuenta el valor SP del último *sprint*. En este caso vamos para arrancar vamos a hacer un promedio con los valores de todos los *sprints* anteriores.	3	0.8	2.4

R2 - Baja productividad por falta de motivación	Mitigar	Fomentar el buen ambiente de trabajo, ofrecer las tareas más amigables a quien se encuentra desmotivado.	3	0.1	0.3
R3 - Insatisfacción del cliente con el producto	Prevenir	Validaciones por parte del *product owner* en las reuniones de retrospectiva de cada *sprint* y pruebas de aceptación al finalizar la construcción de cada módulo. Mantener contacto fluido para disipar dudas.	5	0.3	1.5
R4 - Insatisfacción de los usuarios finales con el producto	Asumir	Se deben registrar las inquietudes de los usuarios finales, para que sirvan como oportunidades de mejora y aspectos del producto que se deben mejorar.	5	0.4	2.0
R5 – Re-trabajo por comunicación ineficiente del equipo	Mitigar	Desarrollar procesos de comunicación mediante las herramientas que nos brinda la tecnología hoy en día. Explorar potencialidades de la herramienta de gestión seleccionada (Version One).	3	0.3	0.9
R6 - Ineficiencia por falta de experiencia en la metodología *Scrum*	Mitigar	Tratar de realizar las tareas aplicándose en la mayor medida posible a la metodología definida, para mejorar con la práctica. A su vez la metodología debe ir mejorando en la medida que se detecten oportunidades de mejora.	3	0.5	1.5
R7 - No cumplir con el alcance	Mitigar	Trabajar de a pares para agilizar tareas	5	0.5	2.5

del producto por retrasos en la construcción del software		de construcción y que el equipo aprenda más. Pedir licencia en el trabajo para tener mayor capacidad de esfuerzo.			
R8 - Retrasos en la construcción de requerimientos complejos por falta de conocimientos avanzados en las tecnologías utilizadas	Asumir	Buscar información internet sobre diferentes formas de realizar las tareas, asesorarse con gente conocida y experimentada y aplicarlas en el proyecto. También aplicar programación de a pares.	3	0.8	2.4
R9 - Retraso en tareas de gestión por ineficiencia en comunicación con actores externos al equipo	Mitigar	Mantener comunicación fluida tanto con el tutor como con el *product owner*, de forma de poder anticipar y planificar tareas para tener una mejor gestión.	3	0.3	0.9
R10 - Pérdida de interés en el proyecto por parte del cliente	Prevenir	Trabajar seriamente y mantener al cliente al tanto de todos los avances del proyecto, hacerlo partícipe mediante las validaciones, pruebas de aceptación y pidiendo opiniones personales en temas que corresponda.	5	0.1	0.5
R11 - Pérdida de datos del proyecto (documentación, código, etc.)	Mitigar	Realizar respaldos periódicos y semanales.	4	0.2	0.8
R12 - Incendio, robo o cualquier catástrofe con el servidor de aplicaciones y datos	Mitigar	Mantener respaldos descentralizados.	5	0.1	0.5

R13 - Retraso por detectar fallas en forma tardía por no utilizar casos, ni datos de prueba para realizar los diferentes tipos de pruebas	Asumir	Realizar pruebas y documentación de hallazgos según proceso de pruebas definido. Al ser un equipo chico, el código es conocido por todos y de encontrar fallas a la hora de integrar o realizar pruebas del sistema, la localización no sería tan compleja.	3	0.8	2.4
R14 - Retrasos en la construcción de funcionalidades por tareas no previstas/estimadas o cambios surgidos a mitad de *sprint*	Mitigar	Si la tarea o cambio requiere un esfuerzo menor a 8 horas, el cambio se incluye dentro del mismo *sprint*. En caso que el esfuerzo requerido sea mayor, se aborta el *sprint* y se planifica uno nuevo.	3	0.5	1.5
R15 - Retrasos en construcción de producto por contar con demasiadas correcciones para hacer. Pruebas de aceptación con *product owner* demasiado esporádicas.	Mitigar	Mantener al tanto al *product owner* de que es lo que se está haciendo.	3	0.6	1.8
R16 - Falta de tiempo para corregir la aplicación en caso de que las pruebas alfa generen incidentes.	Asumir	Registrar los incidentes registrados por los usuarios, analizarlos y ponerlos como mejoras necesarias para el obtener la primera versión del producto.	5	0.7	3.5

Figura 11 – Análisis de riesgos para el *sprint* 9.

Sprint 10

Riesgo	Estrategia	Acciones a realizar	Impacto	Probabilidad	Magnitud
R1 - Desvíos en el cronograma por falta de experiencia en la estimación de tareas y en gestión de proyectos	Mitigar	Modificar la forma de estimar, hasta ahora no habíamos tenido en cuenta el valor de nuestro SP promedio. Para tratar de mitigar este problema de acá al fin del proyecto vamos a estimar tomando en cuenta el valor SP del último *sprint*. En este caso vamos para arrancar vamos a hacer un promedio con los valores de todos los *sprints* anteriores.	3	0.7	2.1
R2 - Baja productividad por falta de motivación	Mitigar	Fomentar el buen ambiente de trabajo, ofrecer las tareas más amigables a quien se encuentra desmotivado.	3	0.1	0.3
R3 - Insatisfacción del cliente con el producto	Prevenir	Validaciones por parte del *product owner* en las reuniones de retrospectiva de cada *sprint* y pruebas de aceptación al finalizar la construcción de cada módulo.	5	0.4	2.0
R4 - Insatisfacción de los usuarios finales con el producto	Asumir	Se deben registrar las inquietudes de los usuarios finales, para que sirvan como oportunidades de mejora y aspectos del producto que se deben mejorar.	5	0.4	2.0

R5 – Re-trabajo por comunicación ineficiente del equipo	Mitigar	Desarrollar procesos de comunicación mediante las herramientas tecnológicas. Explorar potencial de la herramienta de gestión seleccionada (*Version One*).	3	0.2	0.6
R6 - Ineficiencia por falta de experiencia en la metodología *Scrum*	Mitigar	Tratar de realizar las tareas aplicándose en la mayor medida posible a la metodología definida, para mejorar con la práctica. A su vez, la metodología debe ir mejorando en la medida que se detecten oportunidades de mejora.	3	0.5	1.5
R7 - No cumplir con el alcance del producto por retrasos en la construcción del software	Mitigar	Trabajar de a pares para agilizar tareas de construcción y que el equipo aprenda más. Pedir licencia en el trabajo para tener mayor capacidad de esfuerzo.	5	0.5	2.5
R8 - Retrasos en la construcción de requerimientos complejos por falta de conocimientos avanzados en las tecnologías utilizadas	Asumir	Buscar información internet sobre diferentes formas de realizar las tareas, asesorarse con gente conocida y experimentada y aplicarlas en el proyecto. También aplicar programación de a pares.	3	0.6	1.8
R9 - Retraso en tareas de gestión por ineficiencia en comunicación con actores externos al equipo	Mitigar	Mantener comunicación fluida tanto con el tutor como con el *product owner*, de forma de poder anticipar y planificar tareas para tener una mejor gestión.	3	0.3	0.9
R10 - Pérdida de interés en el	Prevenir	Trabajar seriamente y mantener al	5	0.1	0.5

proyecto por parte del cliente		cliente al tanto de todos los avances del proyecto, hacerlo partícipe mediante las validaciones, pruebas de aceptación y pidiendo opiniones personales en temas que corresponda.			
R11 - Pérdida de datos del proyecto (documentación, código, etc.)	Mitigar	Realizar respaldos periódicos y semanales	4	0.2	0.8
R12 - Incendio, robo o cualquier catástrofe con el servidor de aplicaciones y datos	Mitigar	Mantener respaldos descentralizados.	5	0.1	0.5
R13 - Retraso por detectar fallas en forma tardía por no utilizar casos, ni datos de prueba para realizar los diferentes tipos de pruebas	Asumir	Realizar pruebas y documentación de hallazgos según proceso de pruebas definido. Al ser un equipo chico, el código es conocido por todos y de encontrar fallas a la hora de integrar o realizar pruebas del sistema, la localización no sería tan compleja.	3	0.8	2.4
R14 - Retrasos en la construcción de funcionalidades por tareas no previstas/estimadas o cambios surgidos a mitad de *sprint*	Mitigar	Si la tarea o cambio requiere un esfuerzo menor a 8 horas, el cambio se incluye dentro del mismo *sprint*. En caso que el esfuerzo requerido sea mayor, se aborta el *sprint* y se planifica uno nuevo.	3	0.5	1.5
R15 - Retrasos en construcción de producto por	Mitigar	Mantener al tanto al *product owner* de que es lo que se está haciendo.	3	0.6	1.8

contar con demasiadas correcciones para hacer. Pruebas de aceptación con *product owner* demasiado esporádicas.					
R16 - Falta de tiempo para corregir la aplicación en caso de que las pruebas alfa generen incidentes.	Asumir	Registrar los incidentes registrados por los usuarios, analizarlos y ponerlos como mejoras necesarias para el obtener la primera versión del producto.	5	0.6	3.0

Figura 12 – Análisis de riesgos para el *sprint* 10.

Anexo 16 : Plan de pruebas

Introducción

En esta sección del documento, se busca brindar información más detallada sobre la realización y la gestión de las pruebas a lo largo del proyecto. A continuación, se mostrarán detalles referenciados en las diferentes secciones del cuerpo del documento.

Diagrama de flujo del proceso de pruebas funcionales

El proceso de pruebas, fue uno de los resultados obtenidos luego del *sprint* de “Puesta a punto”. El diagrama de flujo, se fue modificando de acuerdo con la evolución del proceso de pruebas. Éste, no sufrió mayores cambios a lo largo del proyecto, sino que se fueron incorporando pequeños detalles a la forma de trabajo.
El diagrama especificado a continuación, fue una referencia de alto nivel durante el proyecto, una guía sobre el mínimo de tareas que se debían realizar para llevar a cabo el proceso pruebas.

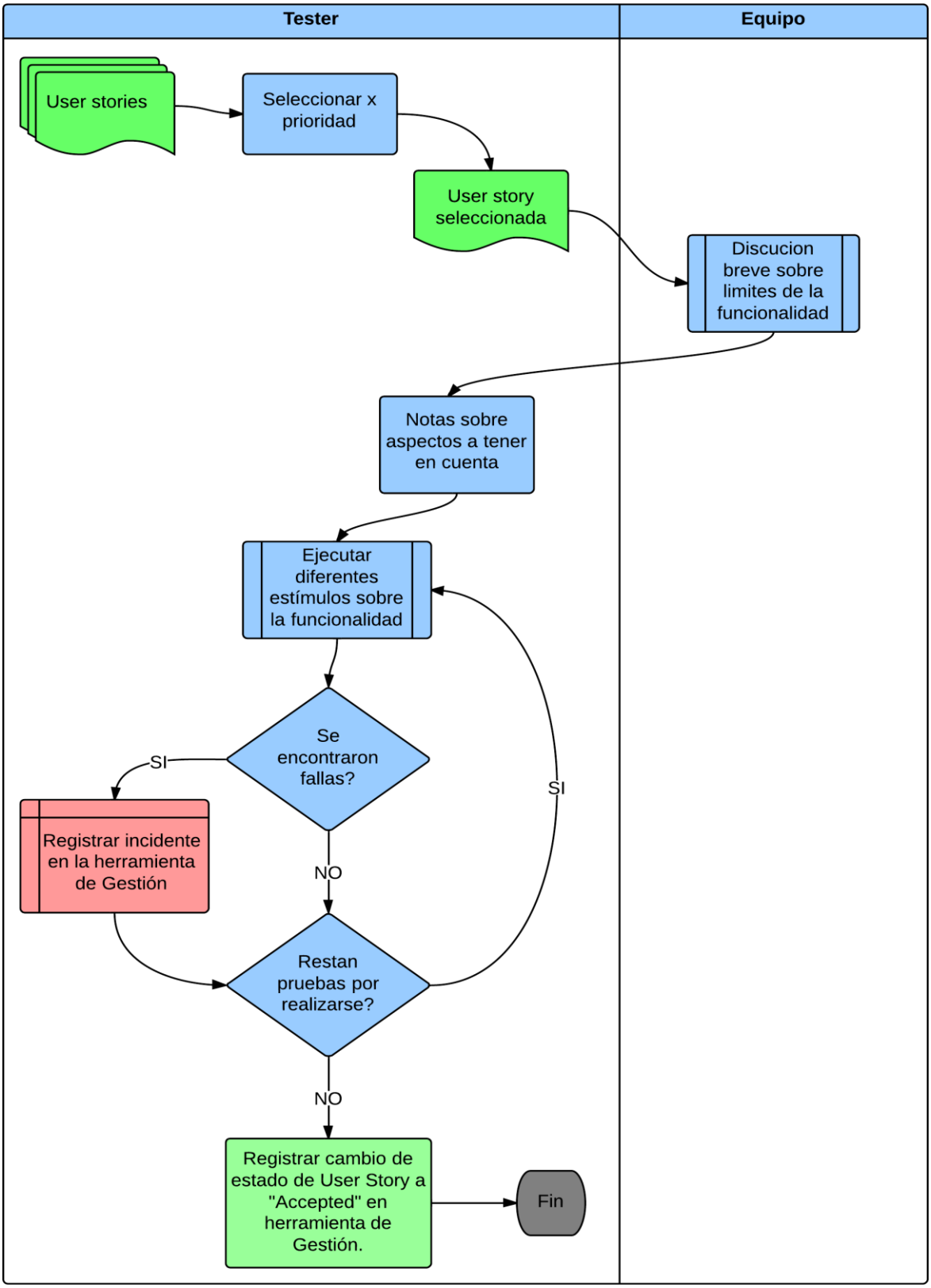


Figura 1 - Diagrama de flujo del proceso de pruebas.

Tabla de seguimiento del registro de fallas a lo largo del proyecto

Durante el proyecto, se utilizó una tabla para realizar el seguimiento de las fallas encontradas en las pruebas. El objetivo, era controlar para cada nivel de criticidad cuantas fallas se habían registrado, y del total de las mismas, cuantas se habían reparado. Cabe destacar, que la reparación de las fallas se realizaba tomando primero las de mayor criticidad. Ésta política, se definió con el fin de no contar con fallas pendientes de reparación con criticidad alta o media.

	Criticidad baja	Criticidad media	Criticidad alta	Fallas totales	Reparados	Total acumulado	Reparados acumulado	Porcentaje reparados
***Sprint* 1**	0	0	0	0	0	0	0	0.0
***Sprint* 2**	6	1	0	7	2	7	2	28.6
***Sprint* 3**	7	2	1	10	6	17	8	47.1
***Sprint* 4**	3	1	0	4	12	21	20	95.2
***Sprint* 5**	0	0	0	0	1	21	21	100.0
***Sprint* 6**	4	3	0	7	5	28	26	92.9
***Sprint* 7**	3	1	2	6	2	34	28	82.4
***Sprint* 8**	4	2	0	6	5	40	33	82.5
***Sprint* 9**	3	1	1	5	8	45	41	91.1
***Sprint* 10**	1	2	0	3	7	48	48	100.0

Tabla 1: planilla de seguimiento de fallas, *sprint* a *sprint*.

Anexo 17 : Evolución de los *sprints*

17.1. *Sprint* "Puesta a punto"

Objetivos del *sprint*

- Generar documentos cuyo contenido nos pueda servir tanto a lo largo del proyecto, como en la fase final de documentación.
- Realizar pruebas de concepto sobre las diferentes plataformas con las que vamos a trabajar a lo largo del proyecto, además generar código que sirva para integrar con la solución del proyecto.
- Comenzar a conocernos a la hora de trabajar y tener una idea del rendimiento y productividad de cada uno en cada actividad, para comenzar a tener una idea a la hora de estimar.

Conclusiones

Se generaron documentos útiles que nos van a ayudar a aplicarnos a la metodología para empezar a incorporarla, los documentos son:

- *Product backlog*
- Plan de SQA
- Plan de riesgos
- Plan de pruebas
- Plan de SCM
- Proceso definido
- Requerimientos no funcionales
- Pruebas de concepto

- Primera versión del Diseño de Arquitectura

La gran mayoría de estos documentos fueron evolucionando a lo largo del proyecto, pero el hecho de tenerlos definidos desde el primer momento nos ayudó a automatizar algunas actividades que se nos presentaron más adelante en el proyecto.

Métricas

Horas dedicadas:

El siguiente gráfico, pertenece a la dedicación horaria de toda la fase de puesta a punto, es decir, desde octubre a diciembre. En el mismo se presenta, la división del porcentaje de horas dedicadas a realizar pruebas de concepto e investigación, y de las dedicadas a definir planes y procesos para la gestión de las diferentes áreas. Éste porcentaje, aparece categorizado como Gestión y Documentación. Cabe destacar, que el gráfico se realizó en base a un total de 387 horas dedicadas.

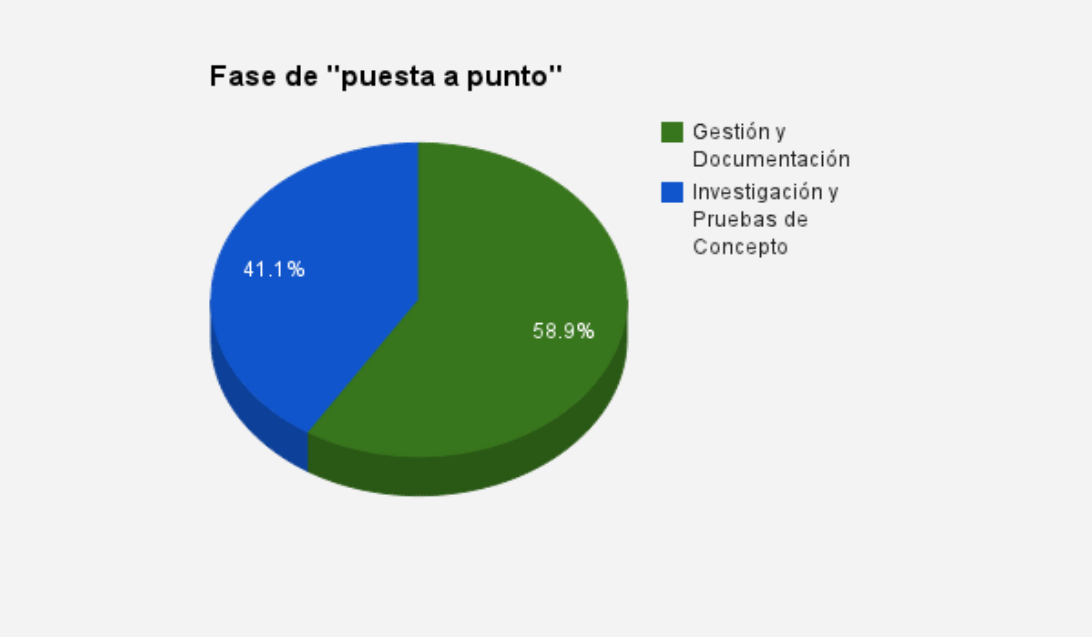


Figura 1 - Dedicación horaria dividida en dos categorías.

17.2. *Sprint* 1

Objetivos del *sprint*

Se comienza con la construcción del producto mediante el desarrollo del módulo de Administración, para eso se estima realizar las siguientes *user stories*:

- Registrar Empresa.
- Editar Empresa.
- Listar Empresas.
- ABM de administradores.
- Inicio de sesión de usuario básico.
- Inicio de sesión de súper-administrador.
- Cerrar sesión
- Registro de un usuario básico a una Empresa.

Resultados

Algunas *user stories* no se pudieron completar en el marco de las dos semanas de trabajo, por lo cual en la semana posterior, en los días que el equipo no se reunió los desarrolladores aprovecharon esas horas para finalizarlas.

Cabe destacar, que para este *sprint* no se contó con uno de los integrantes del equipo, lo cual tiene un impacto sobre las horas dedicas al proyecto. En el siguiente se debería contemplar su incorporación a la hora de estimar el esfuerzo.

Desde el punto de vista de los riesgos, lo más importante fue la desviación en la estimación de esfuerzo por la falta de experiencia de los integrantes del equipo en la gestión de proyectos. El equipo trabajó coordinadamente y no tuvo complicaciones para realizar las tareas que se le asignaron, esto se debió a que ambos integrantes

son compañeros de trabajo, donde además trabajan con las tecnologías involucradas en el desarrollo del módulo de Administración.

Métricas

Métrica	Objetivo	Resultado
Velocidad	Cantidad de *story points* completados en el *sprint*	51 *story points*
Horas dedicadas a construcción del producto	Cantidad de horas dedicadas a diseño, desarrollo y pruebas en un *sprint*	60 horas
Horas dedicas a gestión y documentación	Cantidad de horas dedicadas a actividades de gestión y documentación en un *sprint*	0 horas
Desvío	Diferencia entre horas estimadas y la dedicación real	0 horas
Defectos del equipo por *sprint*	Cantidad de defectos encontrados para las *user stories* desarrolladas en el *sprint*	No hubo actividades de pruebas en este *sprint*

Figura 2 - Métricas del *sprint* 1.

17.3. *Sprint* 2

Objetivos del *sprint*

Se busca finalizar la construcción del módulo de Administración mediante la construcción de la siguiente *user story*:

- Cambio de contraseña

Comenzar la construcción del módulo de Comunicación mediante el desarrollo de las siguientes *user stories:*

- Iniciar sesión
- Listar usuarios básicos registrados
- Aceptar solicitud de contacto

Resultados

Se logró finalizar la mayoría de las *user stories* involucradas, faltó finalizar Aceptar solicitud de contacto, que pertenece al módulo de Comunicación. Al igual que en el *sprint* anterior, algunas *user stories* se finalizaron en la semana posterior a la finalización de esta iteración.

Aumentó la productividad en cuanto a *story points* realizados por el equipo, sobre todo teniendo en cuenta que la dedicación en horas de construcción de producto no fue tanto mayor. Además, el *product owner* sugirió algunas modificaciones por lo cual también hubo re-trabajo, estas horas de trabajo se llevaron a cabo en la semana posterior a la finalización del *sprint.*

Se encontraron dificultades para el desarrollo de algunas *user stories* en el módulo de Comunicación. Esto se debe a que los desarrolladores del equipo no están familiarizados con las tecnologías iOS y Android, a pesar de haber realizado investigación y pruebas de concepto durante la puesta a punto del proyecto.

Métricas

Métrica	Objetivo	Resultado
Velocidad	Cantidad de *story points* completados en el *sprint*	58 *story points*
Horas dedicadas a construcción del producto	Cantidad de horas dedicadas a diseño, desarrollo y pruebas en un *sprint*	66 horas
Horas dedicas a gestión y documentación	Cantidad de horas dedicadas a actividades de gestión y documentación en un *sprint*	27 horas
Desviación	Diferencia entre horas	-6 horas

	estimadas y la dedicación real	
Defectos del equipo por *sprint*	Cantidad de defectos encontrados para las *user stories* desarrolladas en el *sprint*	7 fallas

Figura 3 - Métricas del *sprint* 2

17.4. *Sprint* 3

Objetivos del *sprint*

Avanzar con la construcción del módulo de Comunicación, mediante la finalización de Aceptar solicitud de contacto, que no se pudo finalizar en el *sprint* anterior y la construcción de las siguientes *user stories*:

- Listar secciones de aplicación
- Enviar solicitud de contacto
- Enviar mensaje

Resultados

Se finalizó la construcción de las *user stories* involucradas. Durante el transcurso del *sprint* surgió la necesidad de realizar algunas tareas de mejora, las cuales fueron realizadas luego de finalizar las tareas estimadas. Estas nuevas tareas debieron finalizarse la semana posterior a la finalización y no se incluyeron como *story points* del *product backlog* sino que se tomaron como re-trabajo.

Sobre el trayecto final de este *sprint*, se le informó al tutor de esta situación pero ya era tarde para tomar medidas. Él sugirió al equipo, que ante situaciones de estas características era preferible evaluar el esfuerzo de lo que se debía realizar. En caso de que el esfuerzo fuera menor, se podía continuar con el *sprint* pero en caso de que fuera un esfuerzo considerable, era conveniente abortar y re-planificar el mismo.

En cuanto a los riesgos, comienza a crecer la probabilidad de sufrir retrasos por falta de experiencia en el uso de las tecnologías involucradas. Además se detectó el riesgo de encontrarse con tareas no planificadas o desconocidas al comienzo de un *sprint*. Por último, el equipo encuentra dificultadas con la metodología de trabajo distribuida, por lo cual se tomaran medidas al respecto.

Métricas

Métrica	Objetivo	Resultado
Velocidad	Cantidad de *story points* completados en el *sprint*	47 *story points*
Horas dedicadas a construcción del producto	Cantidad de horas dedicadas a diseño, desarrollo y pruebas en un *sprint*	75 horas
Horas dedicas a gestión y documentación	Cantidad de horas dedicadas a actividades de gestión y documentación en un *sprint*	45 horas
Desviación	Diferencia entre horas estimadas y la dedicación real para la construcción del producto	5 horas
Defectos del equipo por *sprint*	Cantidad de defectos encontrados para las *user stories* desarrolladas en el *sprint*	10 fallas

Figura 4 - Métricas del *sprint* 3.

Las métricas registradas para la dedicación horaria en tareas de gestión y documentación llaman la atención por cantidad. Dentro de ese conjunto de horas, se incluyen horas dedicadas a tareas de gestión, tareas de documentación, reuniones de equipo y con terceros, y horas dedicadas a investigación sobre cómo realizar determinadas actividades vinculadas a la gestión.

Como se mencionó anteriormente, este sprint fue un hito en el proyecto, y el equipo aún estaba tratando de solucionar inconvenientes en materia de gestión, por esa razón no se pudo realizar un desglose detallado de las horas dedicadas a las actividades involucradas.

17.5. *Sprint* 4

Objetivos del *sprint*

Se continúa con la construcción del módulo de Comunicación mediante las siguientes *user stories*:

- Recibir mensaje.

Resultados

Se finalizó la *user story* involucrada. Cabe destacar que en este *sprint*, el equipo no contó con uno de sus integrantes, esto se refleja tanto en las horas dedicadas al proyecto como en los *story points* implementados.

En cuanto a los riesgos, la falta de experiencia en las tecnologías utilizadas sigue siendo un riesgo dado la complejidad de las tareas que se deben llevar a cabo. El equipo comenzó a reunirse con más frecuencia, eso tuvo un impacto positivo en la coordinación de las tareas y en la comunicación. Se decidió definir un lugar fijo para utilizar como oficina del equipo, se definió un horario de trabajo y todos los integrantes coordinaron su horario de trabajo para poder brindarle mayor dedicación al proyecto.

Métricas

Métrica	Objetivo	Resultado
Velocidad	Cantidad de *story points* completados en el *sprint*	34 *story points*
Horas dedicadas a construcción del producto	Cantidad de horas dedicadas a diseño, desarrollo y pruebas en un *sprint*	46 horas
Horas dedicas a gestión y documentación	Cantidad de horas dedicadas a actividades de gestión y documentación en un *sprint*	30 horas

Desviación	Diferencia entre horas estimadas y la dedicación real para la construcción del producto	-4 horas
Defectos del equipo por *sprint*	Cantidad de defectos encontrados para las *user stories* desarrolladas en el *sprint*	4 fallas

Figura 5 - Métricas del *sprint* 4.

17.6. *Sprint* 5

Objetivos del *sprint*

Continúa la construcción del módulo de Comunicación, se estima realizar las siguientes *user stories*:

- Crear grupo.
- Agregar usuario básico a un grupo
- Eliminar usuario básico de un grupo
- Abandonar grupo
- Eliminar grupo

Resultados

Luego de comenzado el *sprint*, se decidió abortarlo por cambios importantes en la arquitectura, se estimó el esfuerzo que demandarían las tareas y resultó ser demasiado trabajo para incluir en este *sprint*.

Se decidió modificar la forma de comunicación entre los dispositivos, resolviendo que los mensajes van a dejar de enviarse por http y el nuevo formato será xmpp. Este nuevo formato es más eficiente y además provee más facilidades para el intercambio de información adicional en cada mensaje.

Métricas

Métrica	Objetivo	Resultado
Velocidad	Cantidad de story points completados en el *sprint*	0
Horas dedicadas a construcción del producto	Cantidad de horas dedicadas a diseño, desarrollo y pruebas en un *sprint*	16 horas
Horas dedicas a gestión y documentación	Cantidad de horas dedicadas a actividades de gestión y documentación en un *sprint*	8 horas
Desviación	Diferencia entre horas estimadas y la dedicación real para la construcción del producto	*Sprint abortado*
Defectos del equipo por *sprint*	Cantidad de defectos encontrados para las user stories desarrolladas en el *sprint*	0

Figura 6 - Métricas del *sprint* 5.

17.7. *Sprint* 6

Objetivos del *sprint*

Se comienza con la construcción del producto y se estima realizar las siguientes *user stories*:

- Pruebas de concepto XMPP
- Integrar XMPP con aplicación
- Crear grupo

Resultados

Se logró modificar la tecnología de comunicación, la cual quedó integrada a la solución. En cuanto a la construcción de grupos, no se logró avanzar todo lo deseado, con lo cual el resto de la funcionalidad debe continuarse construyendo en el próximo *sprint*.

Luego de la revisión, el revisor advirtió que el equipo debía aumentar el esfuerzo en tareas de gestión. Por ese motivo, el tutor del equipo sugirió comenzar a escribir capítulos de áreas en las cuales se debía mejorar, eso facilitaría la comprensión de los aspectos a mejorar. Para cumplir con esos objetivos, el equipo comenzó a documentar los capítulos de Calidad, Gestión y por último el *abstract* del proyecto.
Al finalizar el *sprint*, se logró finalizar la primera versión del *abstract* y se comenzó a escribir tanto el capítulo de Gestión como el de Calidad.

En cuanto a los riesgos, la coordinación del equipo ha mejorado notoriamente gracias a que se comienza a notar la automatización de las tareas y el uso de la metodología. El avance en cuanto a la construcción del producto continua siendo insatisfactorio, eso hace hizo que el riesgo de no cumplir con el alcance del producto comience a ser muy elevado. Para controlar esos inconvenientes, todos los integrantes del equipo resolvieron pedir licencia en sus respectivos trabajos para dedicarse al proyecto en forma completa.

A partir del siguiente *sprint*, se comenzará a estimar a partir de la productividad que se ha tenido en *sprints* anteriores. Se realizará un promedio del valor del *story point*/hora de los *sprints* realizados, y a partir de la estimación de horas a dedicar, se obtendrá la capacidad de construcción del equipo en cuanto a *story points.*

Métricas

Métrica	Objetivo	Resultado
Velocidad	Cantidad de *story points* completados en el *sprint*	25 *story points*
Horas dedicadas a construcción del producto	Cantidad de horas dedicadas a diseño, desarrollo y pruebas en un *sprint*	83 horas
Horas dedicas a gestión y	Cantidad de horas dedicadas a actividades de gestión y	43 horas

documentación	documentación en un *sprint*	
Desviación	Diferencia entre horas estimadas y la dedicación real para la construcción del producto.	-2 horas
Defectos del equipo por *sprint*	Cantidad de defectos encontrados para las *user stories* desarrolladas en el *sprint*	7 fallas

Figura 7 - Métricas del *sprint* 6.

17.8. *Sprint* 7

Objetivos del *sprint*

En primer lugar, comenzar a estimar basándose en la productividad (*story point*/hora) de los *sprints* anteriores. Y segundo, seguir desarrollando el módulo de Comunicación, finalizar Crear grupo y construir las siguientes *user stories*:

- Agregar usuario básico a un grupo
- Eliminar usuario básico de un grupo
- Abandonar grupo
- Eliminar grupo

Resultados

Se logró finalizar la *user story* Crear Grupo, pero el resto de las tareas se finalizaron únicamente para la plataforma iOS. Motivo por el cual, en el siguiente *sprint* se asignaran las *user stories* que quedaron por finalizar.

El equipo encuentra inconvenientes para resolver tareas complejas en las tecnologías utilizadas. En este caso, el equipo brindó la mayor dedicación horaria hasta el momento, ya que durante la segunda semana sus integrantes se dedicaron

a tiempo completo. Sin embargo, la productividad fue la más baja de todos los *sprints* realizados.

En caso de logar finalizar el módulo de Comunicación en la siguiente iteración, el riesgo de no cumplir con el alcance del producto disminuirá.

Métricas

Métrica	Objetivo	Resultado
Velocidad	Cantidad de *story points* completados en el *sprint*	68 *story points*
Horas dedicadas a construcción del producto	Cantidad de horas dedicadas a diseño, desarrollo y pruebas en un *sprint*	126 horas
Horas dedicas a gestión y documentación	Cantidad de horas dedicadas a actividades de gestión y documentación en un *sprint*	50 horas
Desviación	Diferencia entre horas estimadas y la dedicación real para la construcción del producto	6 horas
Defectos del equipo por *sprint*	Cantidad de defectos encontrados para las *user stories* desarrolladas en el *sprint*	6 fallas

Figura 8 - Métricas del *sprint* 7.

17.9. *Sprint* 8

Objetivos del *sprint*

Finalizar la construcción del módulo de Comunicación mediante el cierre de las siguientes *user stories*:

- Agregar usuario básico a un grupo
- Abandonar grupo
- Eliminar grupo
- Aspectos de configuración para plataforma iOS

Comenzar la construcción del módulo de Soporte mediante la construcción de la siguiente *user story:*

- Enviar mensaje desde la web a dispositivo

Resultados

Se finalizó la construcción del módulo de Comunicación, no se realizaron algunas pruebas de integración que se verán retrasadas por la tercera revisión del proyecto. Además no se logró finalizar Enviar mensaje desde la web a dispositivo, con lo cual quedó asignado para el siguiente *sprint*.

El riesgo de sufrir retrasos por falta de experiencia en las tecnologías disminuyó por dos razones, primero, porque el proyecto le ha brindado experiencia a los desarrolladores y les permitió familiarizarse con las mismas. Y en segundo lugar, porque mucho del código desarrollado se reutilizará en el módulo de Soporte.

Métricas

Métrica	Objetivo	Resultado
Velocidad	Cantidad de *story points* completados en el *sprint*	34 *story points*
Horas dedicadas a construcción del producto	Cantidad de horas dedicadas a diseño, desarrollo y pruebas en un *sprint*	117 horas
Horas dedicas a gestión y documentación	Cantidad de horas dedicadas a actividades de gestión y documentación en	50 horas

	un sprint	
Desviación	Diferencia entre horas estimadas y la dedicación real para la construcción del producto	-3 horas
Defectos del equipo por *sprint*	Cantidad de defectos encontrados para las *user stories* desarrolladas en el *sprint*	6 fallas

Figura 9 - Métricas del *sprint* 8.

17.10. *Sprint* 9

Objetivos del *sprint*

Finalizar la construcción de Enviar mensaje desde la web a dispositivo e implementar las siguientes *user stories*:

- Enviar mensaje desde dispositivo a la web
- Listar conversaciones con usuarios web
- Listar solicitudes de usuarios web
- Aceptar solicitud de usuarios web
- Eliminar solicitudes aceptadas

Resultados

Se finalizó la construcción del módulo de Soporte.
En la semana entrante se realizarán las pruebas alfa en el cliente, y posteriormente la evaluación de sobre las pruebas de usabilidad del producto, con lo cual el riesgo más importante pasa por conocer la satisfacción de los usuarios con la aplicación.

Métricas

Métrica	Objetivo	Resultado
Velocidad	Cantidad de *story points* completados en el *sprint*	62 *story points*
Horas dedicadas a construcción del producto	Cantidad de horas dedicadas a diseño, desarrollo y pruebas en un *sprint*	83 horas
Horas dedicas a gestión y documentación	Cantidad de horas dedicadas a actividades de gestión y documentación en un *sprint*	63 horas
Desviación	Diferencia entre horas estimadas y la dedicación real para la construcción del producto	3 horas
Defectos del equipo por *sprint*	Cantidad de defectos encontrados para las *user stories* desarrolladas en el *sprint*	5 fallas

Figura 10 - Métricas del *sprint* 9.

17.11. *Sprint* 10

Objetivos del *sprint*

Comenzar la construcción del módulo de Servicios y realizar todas las *user stories* posibles, ya que este es el último *sprint*. Las *user stories* involucradas son:

- Listar Empresas por rubro
- Ver datos de Empresa
- Listar contactos de Empresa
- Listar empresas por nombre
- Búsqueda (por nombre o rubro)

Resultados

No se logró completar el total de las funcionalidades. Quedaron pendientes: Listar empresas por nombre y Búsqueda (por nombre o rubro).

No se pudo dedicar el total de horas que se habían estimado para la construcción del producto. El equipo, estaba enfocado en la documentación final del proyecto dado que restaban pocos días para la entrega de la misma.

Métricas

Métrica	Objetivo	Resultado
Velocidad	Cantidad de *story points* completados en el *sprint*	32 *story points*
Horas dedicadas a construcción del producto	Cantidad de horas dedicadas a diseño, desarrollo y pruebas en un *sprint*	43 horas
Horas dedicas a gestión y documentación	Cantidad de horas dedicadas a actividades de gestión y documentación en un *sprint*	124 horas
Desviación	Diferencia entre horas estimadas y la dedicación real para la construcción del producto	-16 horas
Defectos del equipo por *sprint*	Cantidad de defectos encontrados para las *user stories* desarrolladas en el *sprint*	3 fallas

Figura 11 - Métricas del *sprint* 10.

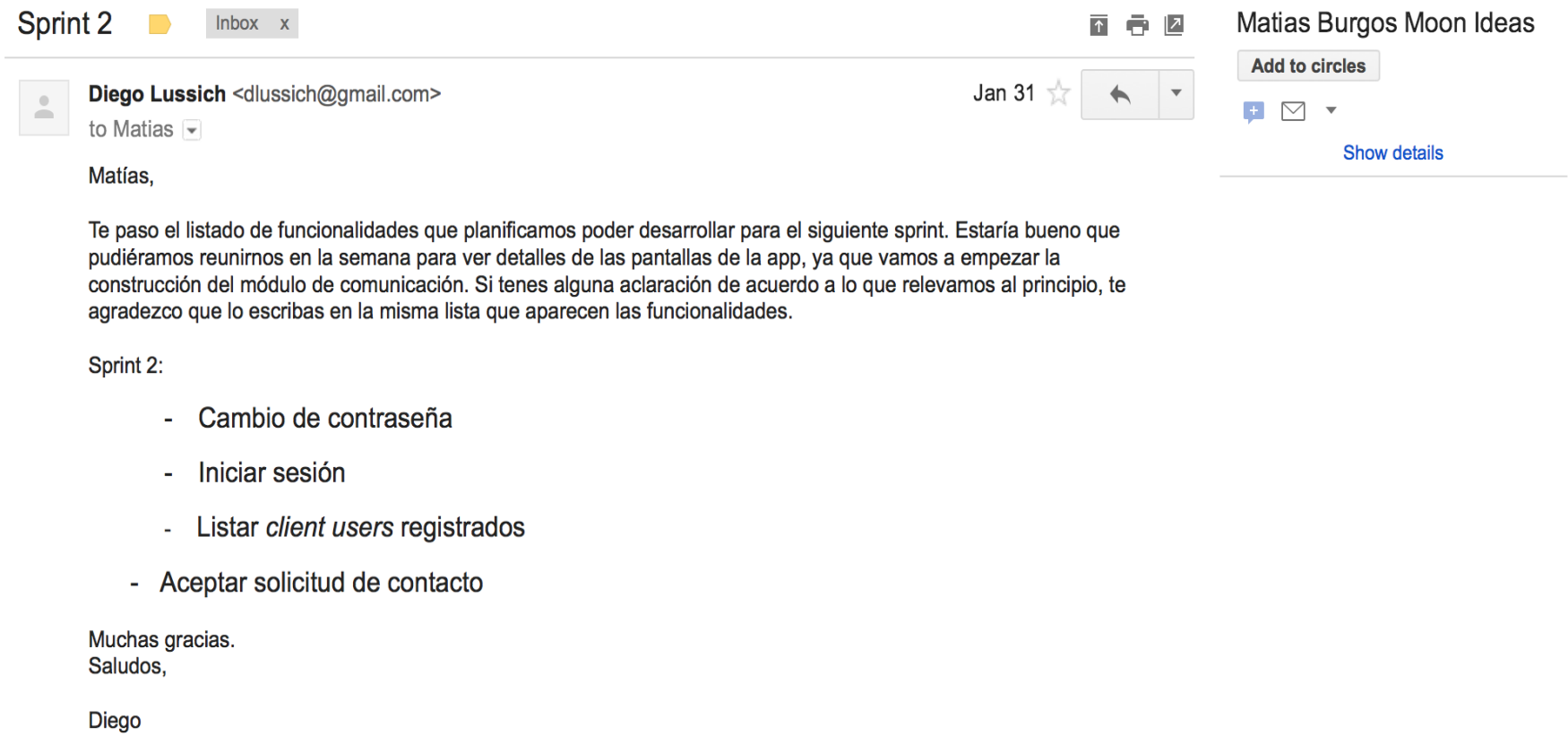
Anexo 18: Evidencia de comunicación con *product owner*

Introducción

En esta sección, se desea evidenciar las comunicaciones establecidas con el *product owner* mediante correo electrónico. Este tipo de comunicaciones, se llevaron a cabo en diferentes instancias de los *sprints* a lo largo del proyecto, y sirvieron para mantener actualizado al cliente.

En las siguientes imágenes, se muestran ejemplos de correos enviados al *product owner* luego de finalizar una reunión de planificación, con su respuesta correspondiente. Y además, un ejemplo de correo enviado luego de una reunión de revisión con la respuesta recibida. Ésta evidencia se debe a que, como mencionamos en el capítulo de Gerencia, durante el proyecto era complejo coordinar reuniones con el referente en fechas precisas.

Correo luego de reunión de planificación



Sprint 2 Inbox x

Diego Lussich <dlussich@gmail.com> Jan 31

to Matias

Matías,

Te paso el listado de funcionalidades que planificamos poder desarrollar para el siguiente sprint. Estaría bueno que pudiéramos reunirnos en la semana para ver detalles de las pantallas de la app, ya que vamos a empezar la construcción del módulo de comunicación. Si tenes alguna aclaración de acuerdo a lo que relevamos al principio, te agradezco que lo escribas en la misma lista que aparecen las funcionalidades.

Sprint 2:

- Cambio de contraseña
- Iniciar sesión
- Listar *client users* registrados
- Aceptar solicitud de contacto

Muchas gracias.
Saludos,

Diego

Matias Burgos Moon Ideas

Add to circles

Show details

Figura 1 - Correo enviado a *product owner* luego de reunión de planificación.

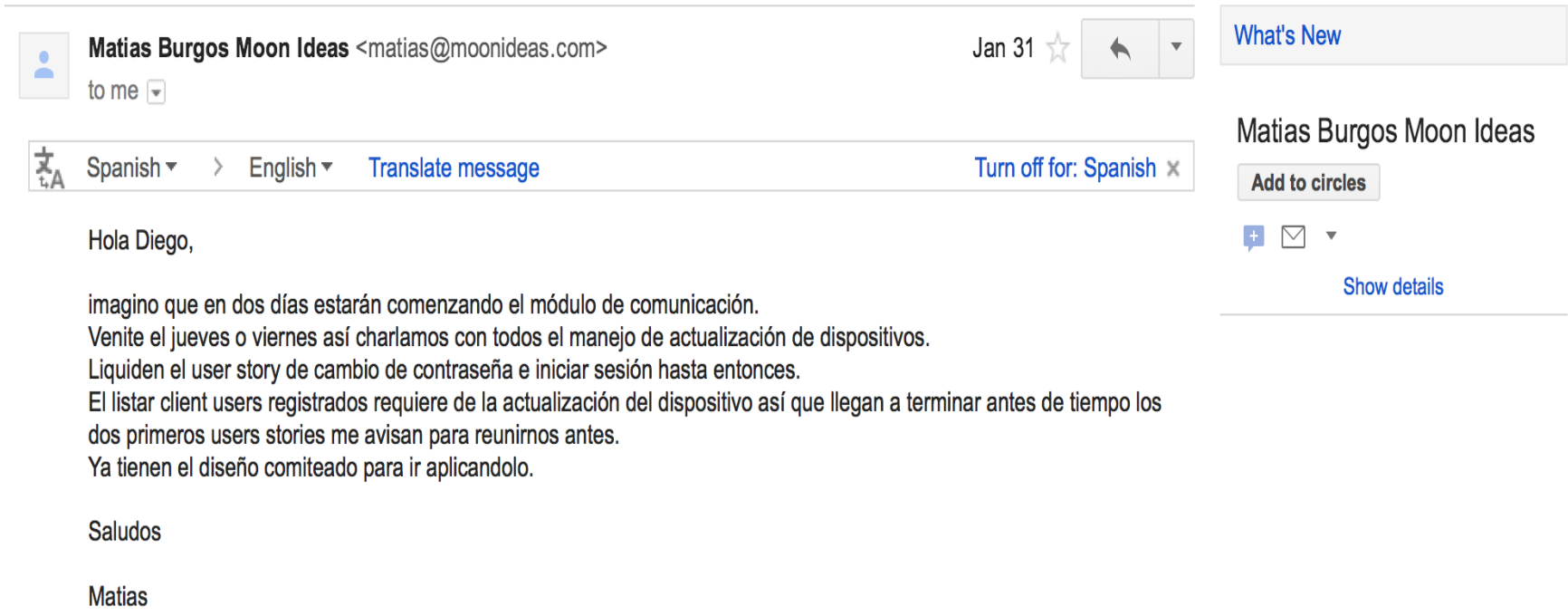


Figura 2 - Respuesta del *product owner* luego de la reunión de planificación.

Correo luego de la finalización de un *sprint*

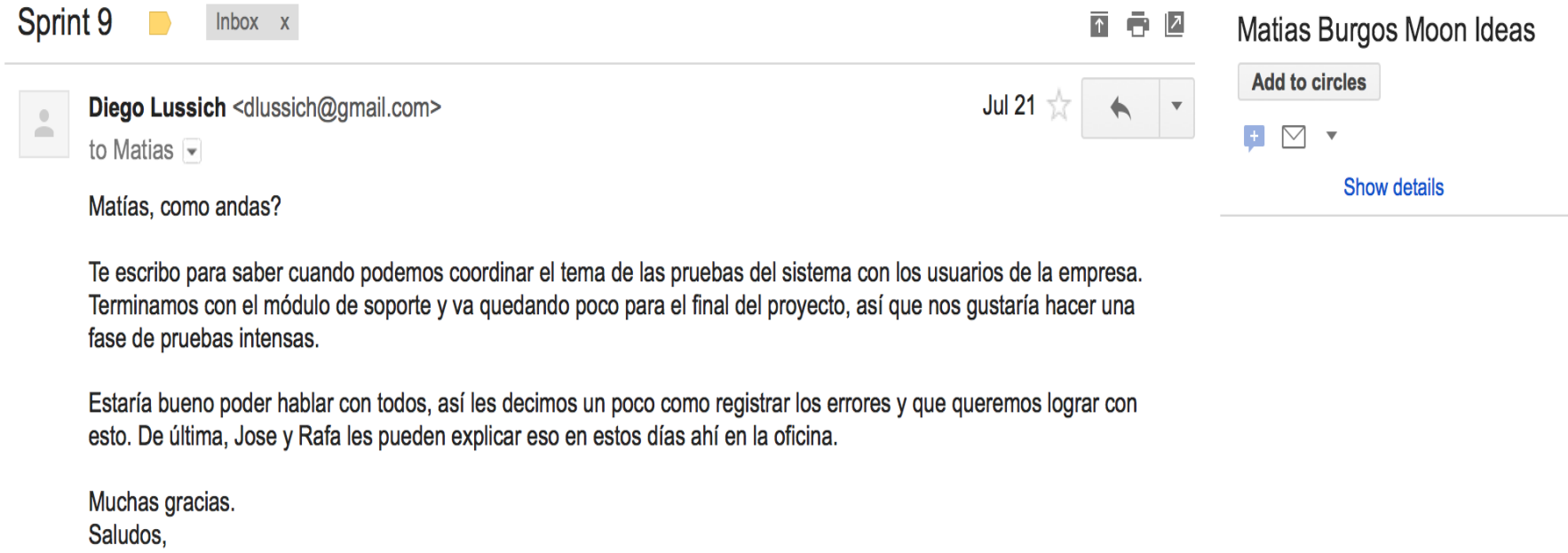


Figura 3 - Correo enviado al *product owner* luego de finalizar el *sprint* 9.

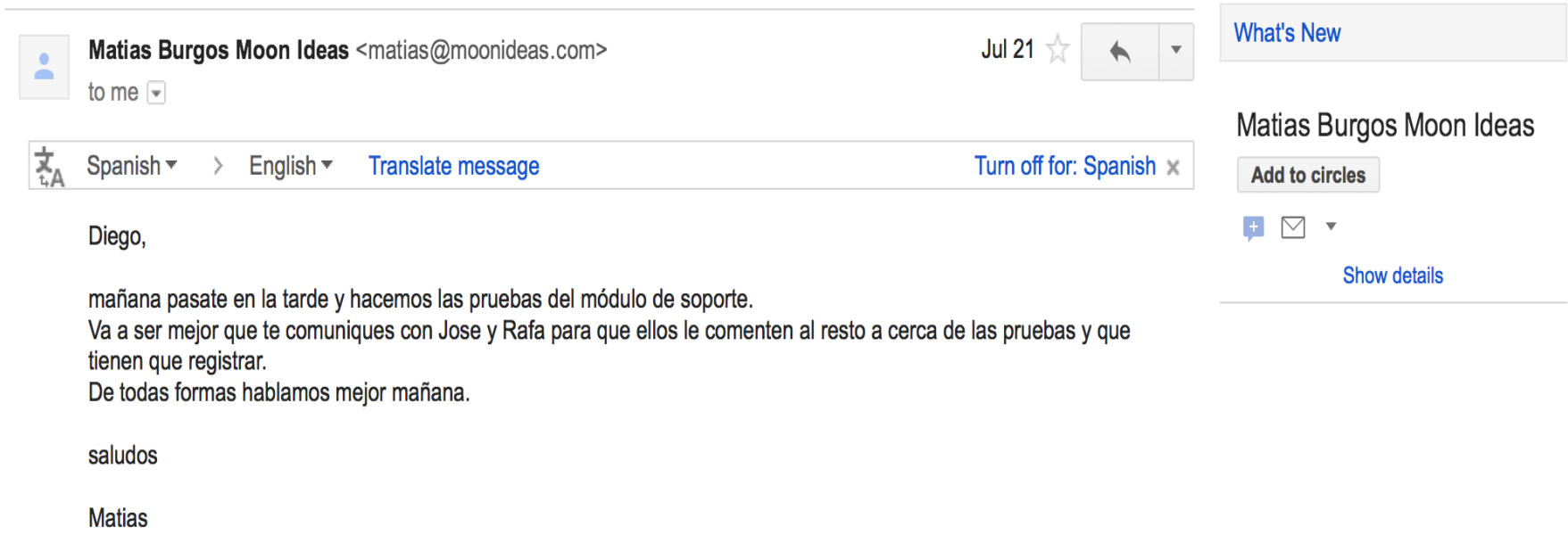


Figura 4 - Respuesta del *product owner* luego de finalizar el *sprint* 9.