UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE INFORMÁTICA
SANTIAGO - CHILE

REINGENIERÍA Y MODELO DE NEGOCIOS PARA UN PROYECTO DE LA FERIA DE SOFTWARE UTFSM - CASO DE ESTUDIO: SMATCH

CRISTIAN FELIPE VALLES PEREIRA

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL EN INFORMÁTICA

Profesor Guía: Sr. José Luis Martí Lara
Profesor Correferente: Sra. Cecilia Reyes Covarrubias

Enero - 2019

DEDICATORIA

Dedico este trabajo a mis padres y en especial a mi abuela, que fueron las personas que desde el inicio de mi vida confiaron en mí y me apoyaron en todo momento.

AGRADECIMIENTOS

Finalizar esta etapa me trae muchos recuerdos de todas las transiciones que he pasado para llegar hasta aquí. Me gustaría agradecer a todas esas personas que he conocido y me han apoyado en esta etapa de mi vida.

Primero quiero agradecer a mis padres Rosa y Cristian, a mi abuela Graciela (Chelita) y a mis hermanas Katy, Alanis, Violett, Pilar y Antonia; muchas gracias por creer en mí y apoyarme en todo momento. Sin ellos hubiese sido imposible llegar hasta donde estoy.

Quiero dar las gracias a mis amigos de la Universidad de Valparaíso, Camilo Ravelo, Ian Quiroga, Marcelo Verdugo, Elias Figueroa, Diego Gatica, Victoria Ara, entre muchas otras grandes personas que me ayudaron en mi primera instancia universitaria. Muchas gracias.

También quiero agradecer a mis hermanos y amigos de la UTFSM, cada uno de uds son personas increíbles que le doy gracias a la vida de haberlos conocido. A Chelo Treimun, Fabián Viani, Carl Angas, German Ortiz, Axel Símonsen, Pablo Ibarra, Tomás González, Gonzalo Moya, Jonathan Galassi, Franz Wompner, Rodrigo Naranjo, Criss, Carlos Jauregui, entre muchas otras personas que no me alcanzaría el texto para mencionarlos a todos pero saben quienes son. De todo corazón les agradezco todo el apoyo, apañe, noches de estudio, vaciles, emprendimientos, gracias a ustedes he crecido como persona. Quiero hacer una mención especial a mi amigo y hermano Diego Reyes, quien me acompañó por todo mi periodo universitario, sé que un amigo así no encontraré en otro lado, gracias por creer en mí y apoyarme cuando más lo necesitaba, gracias por participar en nuestros emprendimientos, no tengo palabras para decirte lo importante que fue tu compañía.

A mi polola Javiera Quezada que me apoyó en mis últimos años universitarios, fueron muy complejos. Gracias por tu amor y compañía, sin duda me has hecho cambiar y ser una persona diferente.

Quiero agradecer a todos los integrantes que componen Nursoft, son personas increíbles.

Por último mis agradecimientos a la profesora Cecilia Reyes, quien me ayudó desde el primer momento, gracias a ella logré desarrollarme en lo que más amo, los emprendimientos. También a mi profesor guía José Luis Martí, no he encontrado mejor docente que ud y fue un gran apoyo durante mi estadía en el campus San Joaquín.

RESUMEN

Resumen— En este trabajo se desarrolla un proceso de reconstrucción de un proyecto que nació en la XXII Feria de Software del año 2014 de la USM, el caso particular del proyecto Smatch. Para llevar a cabo este proceso se hace una reingeniería tanto de software como de modelo de negocios, luego se analiza la distancia que tiene el software para desarrollar a una nueva versión. Después se hace una propuesta de arquitectura y plan de proyecto para el nuevo software. Finalmente se construye una etapa del nuevo plan de proyecto y se hace una validación de interfaz para analizar si se cumplen los objetivos mínimos. En el caso de estudio particular (Smatch) se logra evidenciar las principales diferencias compuestas en los requerimientos no funcionales, por lo tanto se propone una arquitectura de software nuevo, pero manteniendo gran parte de los requerimientos funcionales. Se logra cumplir los objetivos gracias a la importancia de la metodología CANVAS y el agilistmo, el cual se basa la reconstrucción del software. El resultado de este documento refleja una metodología de reconstrucción válida para una *startup*.

Palabras Clave—Reingeniería, Modelo de Negocios, Startup, Metodología Ágil, CANVAS.

ABSTRACT

Abstract—In this work a process of reconstrucción of a software project is developed witch has gone through the process of the software fair of the USM, in particular the Smatch project. To carry out this process, a re-engineering of both software and business model is done, then the distance that the software has to reach the new one is analyzed. Then an architecture proposal and project plan for the new software is made. Finally, a stage of the new project plan is constructed and an interface validation is done to analyze if the minimum objectives are met. In the case of a particular study (Smatch), it is possible to demonstrate the main composite differences in the non-functional requirements, therefore a new software architecture is proposed, but maintaining a large part of the functional requirements. The objectives are achieved thanks to the importance of the CANVAS methodology and the agile methodology, which is based on the reconstruction of the software. The results of this document reflects a valid reconstruction methodology for a *startup*.

Keywords— Re-engineering, Business model, Startup, Agile Methodology, CANVAS..

GLOSARIO

- **Bibliotecas**: conjunto de implementaciones funcionales, codificadas en un lenguaje de programación, que ofrece una interfaz bien definida para la funcionalidad que se invoca.

- ***Framework***: es un conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular que sirve como referencia, para enfrentar y resolver nuevos problemas de índole similar.

- **Hardware**: se refiere a partes físicas, tangibles, de un sistema informático; sus componentes eléctricos, electromecánicos y mecánicos.

- **HTML**: es un lenguaje de marcado que se utiliza para el desarrollo de páginas de internet.

- **JSON**: es un formato de texto ligero para el intercambio de datos.

- **Sistemas monolítico**: son aquellos softwares que se encargan tanto de la administración de la base de datos y lógica, como de las vistas que entrega el sistema.

- **Reingeniería**: proceso de desarrollo en el cual se toma un sistema funcional que se desea mejorar, se lleva a requerimientos funcionales, y se vuelve a implementar, con el objetivo de que el nuevo sistema cumpla lo mismo que el anterior, pero no cometa sus mismos errores o problemas.

- ***Screencast***: es una grabación digital de la salida de la pantalla del computador, también conocida como la captura de pantalla de video, que a menudo contiene narración de audio.

- **Software**: conjuntos de programas y rutinas que permiten a la computadora realizar determinadas tareas.

- ***Startup***: es una organización humana con gran capacidad de cambio que desarrolla productos o servicios , de gran innovación, altamente deseados o requeridos por el mercado, donde su diseño y comercialización están orientados completamente al cliente.

- **UTFSM**: Siglas de Universidad Técnica Federico Santa María.

- **MVVM**: el patrón en inglés *model-view-viewmodel* es un patrón de arquitectura de software. Se caracteriza por tratar de desacoplar lo máximo posible la interfaz de usuario con la lógica de la aplicación.

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

CAPÍTULO 1: DEFINICIÓN DEL PROBLEMA

1.1. CONTEXTO

Cada año, la carrera de Ingeniería Civil en Informática de la Universidad Técnica Federico Santa María genera un evento de gran envergadura, la Feria de Software[1] de la Universidad Técnica Federico Santa María (UTFSM); en donde los alumnos desarrollan un proyecto con base tecnológica durante dos semestres universitarios (un año). Dicho software debe resolver un problema real, que recalque su innovación tecnológica. Los equipos que deben formarse son, por lo general, de 4 a 5 estudiantes de Ingeniería de Informática, el cual dividen sus tareas entre jefe de proyecto, desarrolladores y encargado comercial y/o marketing.

A través de los años, este evento ha generado proyectos muy exitosos, donde algunos estudiantes han querido ir más allá y convertirse en una empresa real para vender su producto y/o servicio. El problema es que la Feria de *Software* crea un producto potente y robusto en cuanto a funcionalidades, pero no todas tienen su debida validación de mercado (como debería tener todo proyecto de *startup*). La falta de dicha validación puede generar funcionalidades innecesarias para el cliente y con errores en la programación. Esto hace que el software sea cada vez más complejo mantenerse en el tiempo y deja imposible seguir avanzando en él.

En el marco de la XXII versión de la Feria de *Software* (año 2014) se construye un proyecto denominado Smatch, el cual fue ganador de dicho evento con el premio *Microsoft* a la categoría de educación y entretenimiento. Además la idea fue participante de otros eventos, siendo finalista del evento *Geek Cam*p[2] y semifinalista del *Jump Chile*[3], ambos llevado a cabo en la Pontificia Universidad Católica de Chile.

Smatch es una plataforma que facilita la organización de eventos deportivos amateur y gestión de reservas de canchas. El software crea el nexo entre los jugadores y los recintos deportivos, a través de reservas en línea. Para mejorar la organización de partidos, el proyecto genera una comunidad de deportistas en línea, que los relaciona con sus mismos gustos deportivos. Por otro lado, la plataforma toma en cuenta a los recintos deportivos,

[1] http://www.feriadesoftware.cl
[2] http://incubauc.cl/geekcamp/
[3] https://www.jumpchile.com

creando un sistema de gestión de canchas, que permite organizar reservas tanto presenciales como en línea.

1.2. IDENTIFICACIÓN DEL PROBLEMA

Si bien la Feria de *Software* de la UTFSM es la primera gran instancia para los estudiantes de Ingeniería Civil en Informática para encontrarse con un proyecto real, al comparar con los proyectos que realmente se venden al mercado, estos presentan falencias importantes. La inexperiencia de los desarrolladores al momento de coordinar y gestionar un proyecto grande, sumado al proceso de evaluación que tiene esta asignatura, provoca los siguientes problemas:

- **Casos de uso sin valor real al cliente**: cada funcionalidad nueva muchas veces no es lo que realmente quiere el cliente, por lo que se crea software innecesario.

- **Demasiados casos de uso**: el proceso de evaluación de la asignatura se basa principalmente en que los proyectos tengan una buena cantidad de casos de uso. Muchas veces los alumnos prefieren inventar casos de uso solo para poder aprobar, generando un software que es innecesariamente robusto.

- **Inexperiencia de los alumnos**: esto provoca problemas tanto en la gestión del proyecto como en el software mismo.

Todos estos puntos no son realmente importantes para la Feria de Software en sí, ya que su propósito es acercar a los alumnos a un proyecto real. Sin embargo, proyectos con mucho potencial pierden oportunidades de negocio debido a estos problemas. Un proyecto que realmente desea generar valor al mercado, debe hacer mucho más. Además, se debe sumar la realidad del mercado; las personas que quieran retomar un proyecto que pasó por este proceso se ve enfrentado a un mercado que cambia rápidamente, pues las necesidades de los clientes van cambiando, existen nuevas tecnologías que podrían permitir mejorar lo que ya se había hecho, además de existir competencia que antes no estaba contemplada, etc.

1.3. SMATCH EN LA ACTUALIDAD

Por iniciativa de los integrantes del equipo de la Feria de *Software* de la UTFSM, en Marzo del 2015, se decide postergar el proyecto Smatch, debido a la alta carga académica dentro de la Universidad. El proyecto contempla casos de uso sin validación, como la generación de torneos/ligas, portal de noticias para que los recintos deportivos pudieran comunicarse

con los jugadores, por otro lado facilita la creación de equipos y partidos entre estos. Cabe mencionar el proyecto fue construido como plataforma web con el *framework* Ruby *on Rails* versión 4 en modo monolítico.

Han pasado tres años desde que fue pospuesto el proyecto, donde el segmento de clientes ha cambiado bastante y la tecnología cada vez se impregna mejor en ellos; además existen competencias la cual se están aprovechando de las tecnologías para ofrecer un servicio mucho más optimizado (comparado a la propuesta que entregó Smatch en su momento).

1.4. OBJETIVOS DE LA SOLUCIÓN

1.4.1. Objetivo general

Reconstruir tanto el software como el modelo de negocios de Smatch a través del proceso de reingeniería, para que pueda volver a ser rentable y que sea de valor para los clientes.

1.4.2. Objetivos específicos

- Replantear el modelo de negocios para crear una nueva propuesta adaptada a las necesidades del mercado actual.
- Aplicar reingeniería al software para verificar, filtrar y/o crear casos de uso acordes a la propuesta de valor, para tener un software acorde a la nueva propuesta de valor.
- Diseñar y construir una nueva arquitectura de software, permitiendo que el proyecto tenga herramientas para poder competir con el mercado.
- Construir y validar una etapa de la solución, para verificar si la propuesta entrega un mínimo valor.

CAPÍTULO 2: ESTADO DEL ARTE

2.1. TECNOLOGÍAS RECIENTES

Hoy en día existe un gran abanico de opciones al momento de escoger la tecnología para un *software*, los lenguajes de programación así como sus *frameworks* y bibliotecas están en un constante cambio, por lo que hace difícil escoger una. Hay ciertos factores que hay que considerar, tales como la popularidad, comunidad, escalabilidad, entre otros. En este apartado, se hace una investigación de los tipos de tecnologías que se están implementando hoy en día y como es la forma en el cual se debe escoger la mejor opción para los proyectos, en específico los enfocado en startups.

2.1.1. Lenguajes de programación actuales

La comunidad de programadores TIOBE[4] es un indicador de la popularidad en los lenguajes de programación, el cual es actualizado cada mes. El ranking se basa en la revisión en tiempo real de más de 300 millones de códigos de diversos software al día. Como se muestra en la Tabla 2.1, los lenguajes destacados son JAVA, C, Python, C++, .Net, PHP, Javascript, SQL, Objetive-C, Pascal y Ruby.

Ha medida que van pasando los años, más lenguajes de programación se van integrando, principalmente a las necesidades que va experimentando el mercado y los ingenieros. En los años 80[5] los lenguajes de programación eran muy básicos, con poca estructura y de ejecución secuencial, lenguajes como Cobol y Pascal eran muy populares. En el nuevo milenio la programación web se hizo más popular permitiendo desarrollar aplicaciones más complejas. Hace unos años, el software se separaba en aplicación web y de escritorio; luego con el lanzamiento de los smartphones se habla de aplicaciones móviles, hasta llegar hoy en día al Internet de las cosas, el cual propone cualquier dispositivo tecnológico puede ser programado. Lo importante es destacar que existen lenguajes de programación muy dedicados a las necesidades del software, por ejemplo, si se necesita un software en el ámbito científico, existen los lenguajes como Python, R y Matlab que apoyan esta área.

[4] (n.d.). TIOBE Programming Community Index. Recuperado el diciembre 4, 2018, de https://www.tiobe.com/tiobe-index/

[5] (2016, abril 20). La evolución de los Lenguajes de Programación - ParcelaDigital. Recuperado el diciembre 5, 2018, de https://parceladigital.com/2016/04/20/la-evolucion-de-los-lenguajes-de-programacion/

Si se necesita crear un *software* más complejo el cual requiere el manejo del uso de la memoria principal, se recomienda el uso del lenguaje C. Por lo tanto se concluye que es de suma importancia elegir el lenguaje de programación adecuado dependiendo de la necesidad del proyecto.

Dec 2018	Dec 2017	Change	Programming Language	Ratings	Change
1	1		Java	15.932%	+2.66%
2	2		C	14.282%	+4.12%
3	4	↑	Python	8.376%	+4.60%
4	3	↓	C++	7.562%	+2.84%
5	7	↑	Visual Basic .NET	7.127%	+4.66%
6	5	↓	C#	3.455%	+0.63%
7	6	↓	JavaScript	3.063%	+0.59%
8	9	↑	PHP	2.442%	+0.85%
9	-	⇈	SQL	2.184%	+2.18%
10	12	↑	Objective-C	1.477%	-0.02%
11	16	⇈	Delphi/Object Pascal	1.396%	+0.00%
12	13	↑	Assembly language	1.371%	-0.10%
13	10	↓	MATLAB	1.283%	-0.29%
14	11	↓	Swift	1.220%	-0.35%
15	17	↑	Go	1.189%	-0.20%
16	8	⇊	R	1.111%	-0.80%
17	15	↓	Ruby	1.109%	-0.32%
18	14	⇊	Perl	1.013%	-0.42%
19	20	↑	Visual Basic	0.979%	-0.37%
20	19	↓	PL/SQL	0.844%	-0.52%

Tabla 2.1: Ranking Tiobe de los lenguajes de programación. Fuente: “Tabla Tiobe de Diciembre del 2018”.

2.1.1.1 Ruby

Ruby[6] es un lenguaje de programación dinámico, open source, enfocado en la simplicidad y la productividad. Tiene una sintaxis elegante, natural al leer y fácil de escribir.

[6] https://www.ruby-lang.org/en/

Este lenguaje es de propósito general, es decir, con Ruby se pueden desarrollar todo tipo de aplicaciones diferentes, tales como de servicio web, clientes de correo electrónico, procesamiento de datos en backend, aplicaciones de red, etc.

¿Porqué utilizar Ruby?

- **Los recursos de Ruby son abundantes**: cuando se trabaja en un lenguaje de programación, los recursos disponibles son elementos imprescindibles. Además de una amplia documentación oficial, hay una gran cantidad de recursos disponibles para el desarrollador, incluyendo libros, *screencasts*, videos, cursos online, clases locales, *bootcamps* de desarrollo, foros, etc.
- **Ruby tiene una comunidad activa**: existe una gran cantidad de desarrolladores que trabajan activamente, lo que permite tener una fuente de ayuda ante problemas recurrentes.
- **Cuenta con más de 60.000 bibliotecas y varios *frameworks* a elegir**: con frameworks y bibliotecas como Ruby on Rails y Chief, se puede incorporar código existente en proyectos propios. *RubyGems* fue creado específicamente para Ruby, el cual ayuda a administrar los numerosos frameworks y bibliotecas construidas en el lenguaje. Todas estas fuentes se traducen en un ciclo de desarrollo más fácil y rápido para los proyectos.

2.1.1.2 Javascript

Javascript es un lenguaje de programación interpretado, dialecto del estándar *ECMASCRIPT*[7]. Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico. Se utiliza principalmente en su forma del lado del cliente, interpretado como parte de un navegador web permitiendo mejoras en la interfaz de usuario y páginas webs dinámicas.

Tradicionalmente se venía utilizando en páginas web *HTML*[8] para realizar operaciones y solo en el marco de la aplicación cliente, sin acceso a funciones del servidor. Hoy en día es también utilizado para enviar y recibir información del servidor junto con la ayuda de

[7] ECMAScript define un lenguaje de tipos dinámicos ligeramente inspirado en Java y otros lenguajes del estilo de C. Soporta algunas características de la programación orientada a objetos mediante objetos basados en prototipos y pseudoclases..

[8] **HTML**, sigla en inglés de ***HyperText Markup Language*** (lenguaje de marcas de hipertexto), hace referencia al lenguaje de marcado para la elaboración de páginas web.

otras tecnologías como *AJAX*[9]. Javascript se interpreta en el agente de usuario al mismo tiempo que las sentencias se van descargando junto con el código HTML.

Javascript está en todas partes, se ha convertido en un lenguaje idóneo para aprender, dispone de muchas y variadas aplicaciones, además de aportar sencillez para las personas que comienzan. En la actualidad, Javascript ha sobrepasado el ámbito de los clientes web, para situarse en casi cualquier parte del software.

- **Javascript Web:** este es el entorno el cual apareció en primer lugar. Su ejecución se centra en el ámbito Web, el cual permite a los desarrolladores aportar interactividad, manipular documento o ventana del navegador, realizar cálculos, etc.
- **Javascript en el backend**: este entorno nació desde la extracción del motor de ejecución de Javascript, que hasta entonces sólo se disponía en el ámbito del navegador; gracias a esto se puede usar este lenguaje para cualquier otro propósito, incluso fuera del cliente web.
- **Javascript como lenguaje de aplicaciones de dispositivos**: Hoy en día es posible utilizar Javascript como lenguaje para la creación de aplicaciones de dispositivos, tales como móviles, tablets, TV, etc. Este enfoque da la posibilidad de usar Javascript como lenguaje de desarrollo de aplicaciones nativas, que no requieren de un entorno web para funcionar.

Cabe destacar que en la Sección 2.1.1 Javascript aparece como uno de los 7 lenguajes de programación más utilizados por los desarrolladores en todos el mundo.

Hoy en día es muy difícil desprenderse de este lenguaje, se quiera o no, está en todas partes, ya que está fuertemente vinculado a al internet. Por lo tanto es muy probable se que ocupe esta tecnología.

[9] **AJAX**, acrónimo de ***A**synchronous **J**avaScript **A**nd **X**ML* (JavaScript asíncrono y XML), es una técnica de desarrollo web para crear aplicaciones interactivas o RIA (*Rich Internet Applications*).

2.1.2. Frameworks actuales

En general, un marco de trabajo, o *framework*[10], es una estructura real o conceptual destinada a servir soporte o guía para la construcción de algo que expande la estructura en algo útil.

Desde el punto de vista en la informática, un *framework* es a menudo una estructura en capas que indica qué tipo de programas pueden o deben ser construidos y cómo se interrelacionan. Algunos marcos de trabajo de sistemas informáticos también incluyen programas reales, especifican interfaces de programación u ofrecen herramientas de programación para usar los marcos. Un framework puede servir para un conjunto de funciones dentro de un sistema y cómo se interrelacionan; las capas de un sistema operativo; las capas de un subsistema de aplicación; cómo debería normalizarse la comunicación en algún nivel de una red; etcétera. Un marco de trabajo es generalmente más completo que un protocolo y más prescriptivo que una estructura.

Al momento de iniciar un proyecto en el cual se desea desarrollar un sistema, debemos tener en cuenta varios factores que van a determinar el crecimiento del mismo, como por ejemplo su estructura, y que por lo general toma parte del tiempo que puede ser enfocado a avance del desarrollo de los requerimientos del sistema. Acá es donde entra el juego la importancia del *framework*, ya que estos proporcionan de una estructura base que te permite agilizar estos procesos y orientar el tiempo y esfuerzo en lógica del software.

Hoy en día es de suma importancia utilizar *frameworks*, ya que se permite el desarrollo de software de alto rendimiento, proporciona a los desarrolladores varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación compleja así como de su mantenimiento futuro.

2.1.2.1. Ruby on Rails

Ruby on rails (ROR) es un *framework* de aplicaciones web, de código abierto escrito en el lenguaje de programación Ruby, siendo el paradigma del patrón Modelo Vista Controlador (MVC). Trata de combinar la simplicidad con la posibilidad de desarrollar aplicaciones en el

[10] (n.d.). ¿Qué es Framework? - Definición en WhatIs.com - SearchDataCenter. Recuperado el diciembre 5, 2018, de https://searchdatacenter.techtarget.com/es/definicion/Framework

mundo real escribiendo menos código que otros *frameworks* y con un mínimo de configuración. El lenguaje de programación Ruby permite la metaprogramación, de la cual Rails hace uso, lo que resulta en una sintaxis que muchos usuarios encuentran muy legible. Rails se distribuye a través de *RubyGems*, que es el formato oficial del paquete y canal de distribución de bibliotecas y aplicaciones Ruby.

Ruby on rails como API

Las Interfaces de Programación de Aplicaciones (API's por sus siglas en inglés) especifican cómo componentes de *software* deberían interactuar entre ellos. Se pueden ver como transacciones que permiten la comunicación entre componentes sin necesidad de conocer las implementaciones internas de éstos, para así poder acceder a información y servicios que permitan la construcción rápida de aplicaciones. Específicamente una API para las aplicaciones móviles es un conjunto de URLs[11] que permiten la interacción de la aplicación con un servidor web.

2.1.2.2. React JS

React es una biblioteca escrita en Javascript, desarrollada por Facebook para facilitar la creación de componentes interactivos, reutilizables, para interfaces de usuario. Se utiliza en Facebook para la producción de componentes, e instagram está escrito completamente en React. Uno de los puntos más destacados, es que no sólo se utiliza para el lado del cliente, sino que también puede representar en el servidor y trabajar juntos.

React está construido en torno a hacer funciones, que toman las actualizaciones de estado de la página y se traducen en una representación virtual de la página resultante. Siempre que React es informado de un cambio de estado, vuelve a ejecutar esas funciones para determinar una nueva representación virtual de la página, por consiguiente, se traduce automáticamente ese resultado en los cambios del DOM[12] necesarios para reflejar la nueva representación de la página.

[11] URL son las siglas en inglés de uniform resource locator (en español, localizador uniforme de recursos), que sirve para nombrar recursos en Internet.

[12] El Modelo de Objetos del Documento (DOM) es una interfaz de programación de aplicaciones (API) para documentos HTML y XML.

React Native

Es un subconjunto de React que permite desarrollar aplicaciones nativas en *Android* e *IOS* usando Javascript. Lo ventajoso es que hereda el mismo comportamiento y estructura de React, lo que aporta flexibilidad y aprovechamiento de código.

2.2 REINGENIERÍA

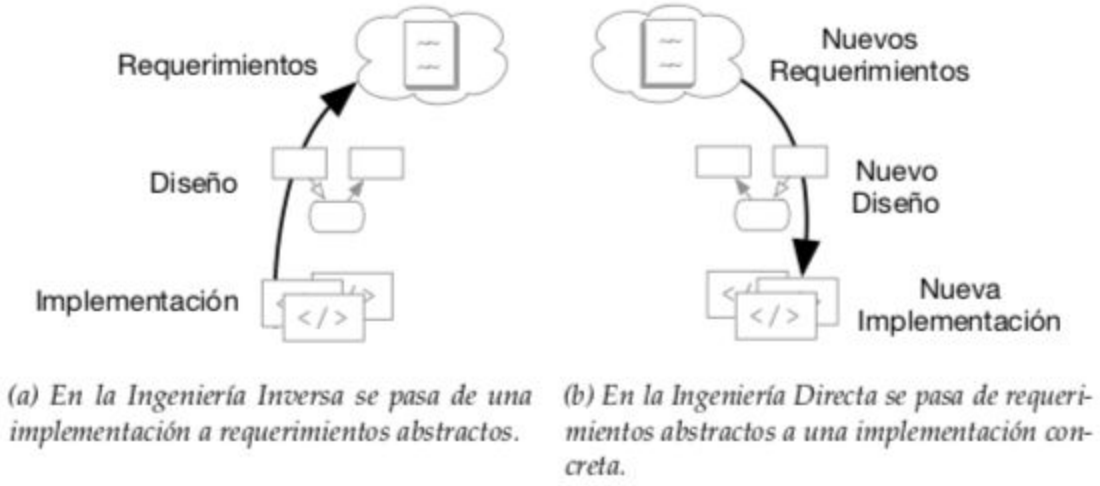
La Reingeniería de software[13] es la examinación, análisis y alteración de un sistema de software existente para reconstruirlo en una nueva forma, y la implementación subsecuente de esa nueva forma. El proceso típico de reingeniería se compone de una combinación de otros procesos como lo es la ingeniería inversa, re-documentación, reestructuración, traslación e ingeniería directa. La meta de la reingeniería es entender el software existente (especificación, diseño, implementación) y entonces re-implementarlo para mejorar las funcionalidades del sistema.

Generalmente, este proceso se debe utilizar cuando el software es muy complejo de mantener o se torna difícil de agregar nuevas funcionalidades. Sin embargo, un software debería usar reingeniería cuando se encuentra con uno o más de estos siguientes problemas:

- Falta de documentación actualizada.
- Falta de pruebas automatizadas .
- Existe poco funcionamiento del funcionamiento del sistema.
- Mal rendimiento.
- Lento paso de funcionalidad a producción.
- Mucho tiempo requerido para realizar cambios simples.
- Bugs constantes.
- Tiempo de compilación elevados.
- Cuando el código en sí es funcional, sin embargo puede llevar problemas de mantenibilidad a futuro.

El proceso abarca principalmente dos subconjuntos, la ingeniería inversa y la ingeniería directa, como se muestra en la Figura 2.1.

[13] (n.d.). Software Re-engineering - Semantic Scholar. Recuperado el diciembre 5, 2018, de https://pdfs.semanticscholar.org/d6b2/89019f9fef924fd3300d1094f29c8bc079f9.pdf



(a) En la Ingeniería Inversa se pasa de una implementación a requerimientos abstractos.

(b) En la Ingeniería Directa se pasa de requerimientos abstractos a una implementación concreta.

Figura 2.1: Los dos pasos de la reingeniería: Ingeniería inversa e ingeniería directa. Fuente: elaboración propia.

A continuación, se explican los pasos de la reingeniería.

2.2.1. Ingeniería inversa

La ingeniería inversa es el proceso de analizar un sistema para identificar sus componentes y las relaciones entre ellos para crear representaciones del sistema en otra forma o en un nivel más alto de abstracción. Es decir, es la acción de pasar de una aplicación concreta a una definición más abstracta al funcionamiento de esta.

2.2.2. Ingeniería directa

Como su nombre lo indica, la ingeniería directa se basa en el proceso contrario a la ingeniería inversa. En este caso se busca ir desde los requerimientos abstractos hacia una implementación concreta. De manera más formal se puede explicar como: El proceso[14]

[14] (2013, noviembre 27). Object-Oriented Reengineering Patterns - SCG. Recuperado el diciembre 5, 2018, de http://scg.unibe.ch/download/oorp/OORP.pdf

tradicional de ir de abstracciones de alto nivel y diseños lógicos hacia la implementación física de un sistema.

2.2 METODOLOGÍAS PARA EL DESARROLLO DE *STARTUPS*

Una *startup* es una institución humana diseñada para para crear un nuevo producto o servicio bajo condiciones de incertidumbre extrema. Es precisamente esa incertidumbre extrema lo que hace que una *startup* (con o sin fines de lucro) no se pueda gestionar con los mismos métodos y estándares que utilizan las empresas consolidadas. Tampoco las nociones de éxito o fracaso son los mismos en ambos ámbitos, porque una *startup* necesita del fracaso y el aprendizaje continuos como mecanismos para evaluar sus hipótesis de partida. En este apartado, se revisan metodologías de desarrollo para una correcta implementación del un software enfocado a emprendimientos.

2.2.1. *Lean Startup*

El método *Lean Startup* es un conjunto de prácticas pensadas para ayudar a los emprendedores a incrementar las probabilidades de una *startup* con éxito. No es una fórmula matemática infalible, sino una filosofía empresarial innovadora que ayuda a los emprendedores a escapar de las trampas del pensamiento empresarial tradicional.

esta metodología hace alusión a que una *startup* contiene un motor de crecimiento, el cual, cada nueva versión de un producto, cada nueva característica y cada nuevo programa de marketing es un intento de mejorar este motor de crecimiento. Sin embargo, no todos estos cambios acaban siendo mejoras. Gran parte del tiempo en la vida de una *startup* transcurre poniendo a punto el motor a través de mejoras en los productos, el marketing o las operaciones.

Este método está diseñado para enseñar a conducir una *startup* a través de la experimentación. En lugar de hacer planes complejos basados en muchas asunciones, se pueden hacer ajustes constantes a través del *circuito de feedback* de CREAR-MEDIR-APRENDER, el cual, es el núcleo de toda la metodología. A través de este proceso de dirección, un proyecto puede como saber si hacer un giro drástico llamado *pivote* o si se debe perseverar en la trayectoria actual.

La Figura 2.2 muestra un resumen de la metodología *Lean Startup* , el cual contiene tres componentes llamados CREAR, MEDIR y APRENDER.

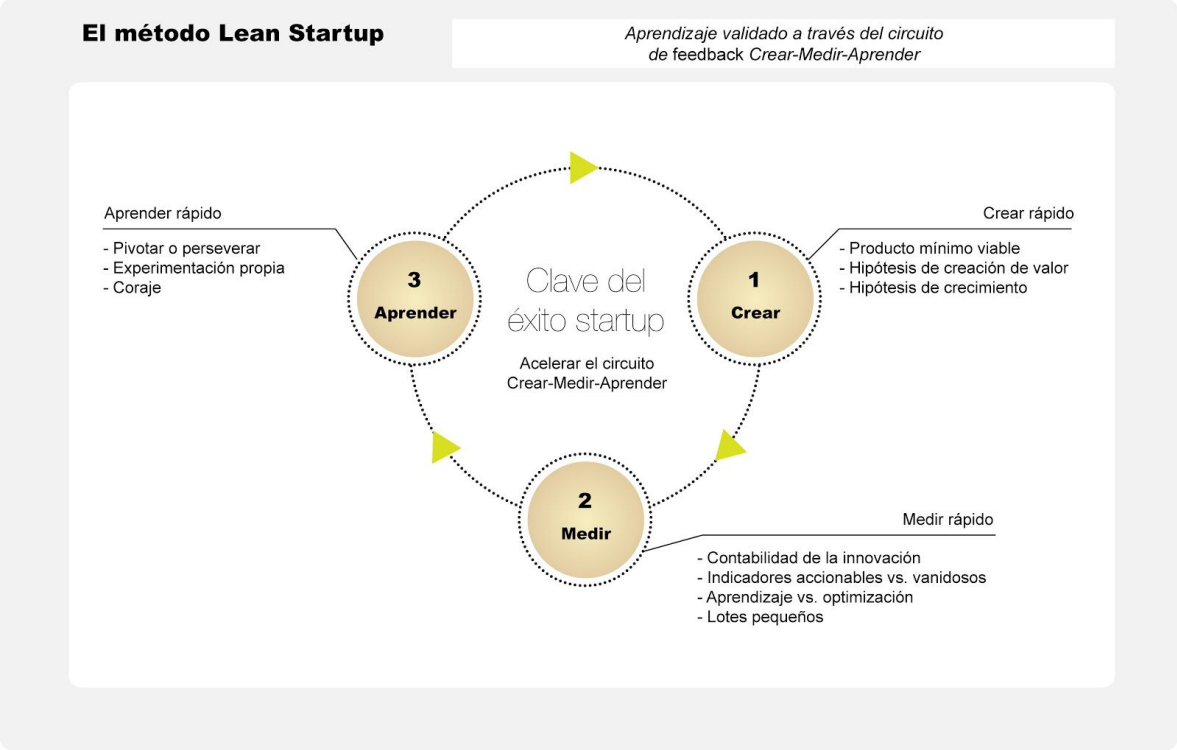


Figura 2.2: Pasos de la metodología *Lean Startup*. Fuente: "Escuela de negocios Iebschool".

2.2.2. Canvas para modelos de negocios

A muchos emprendedores que tienen una idea les resulta difícil plasmarla en papel y desarrollarla. El método CANVAS puede ayudar, de manera sencilla y gráfica, a diseñar y analizar un modelo de negocios.

El lienzo de modelos de negocios o CANVAS consiste en poner sobre un lienzo o cuadro nueve elementos esenciales de las empresas y testar estos elementos hasta encontrar un modelo sustentable en VALOR para crear un negocio exitoso. Estos nueve módulos cubren las cuatro áreas principales de un negocio: clientes, oferta, infraestructuras y viabilidad económica.

A continuación se describen las nueve secciones que componen el modelo canvas.

Segmentos de mercado

En este módulo se definen los diferentes grupos de personas o entidades a los que se dirige una empresa. Los grupos de clientes pertenecen a segmentos diferentes si:

- Los clientes que tienen necesidades diferentes.
- Los clientes tienen distintos canales de distribución.
- El tipo de relación que se tiene con un cliente.
- Las rentas de los clientes son distintas.
- Los clientes pagan por una oferta distinta.

Existen varios segmentos del mercado, a continuación se presentan algunos ejemplos:

- **Mercado de masas**: los modelos de negocio que se centran en el público general no distinguen segmentos de mercado. Tanto las propuestas de valor como los canales de distribución y las relaciones con los clientes se centran en un gran grupo de clientes que tienen necesidades y problemas similares.
- **Nicho de mercado**: los modelos de negocio orientados a nichos de mercado atienden a segmentos específicos y especializados. Las propuestas de valor, los canales de distribución y las relaciones con los clientes se adaptan a los requisitos específicos de una fracción del mercado.
- **Mercado segmentado**: algunos modelos de negocio distinguen varios segmentos de mercado con necesidades y problemas ligeramente diferentes.
- **Mercado diversificado**: una empresa que tenga un modelo de negocio diversificado atiende a dos segmentos de mercado que no están relacionados y que presentan necesidades y problemas muy diferentes.
- **Plataformas multilaterales**: algunas empresas se dirigen a dos o más segmentos de mercado independientes. Una empresa de tarjetas de crédito, por ejemplo, necesita una gran base de clientes y una gran base de comercios que acepten sus tarjetas.

Propuesta de valor

La propuesta de valor es el factor que hace que un cliente se decante por una u otra empresa, es la que soluciona un problema o satisface una necesidad. También se puede definir como una serie de ventajas que se ofrecen a los clientes. Principalmente se separan por dos tipos:

- Ingresos por transacciones de pagos puntuales.

- Ingresos recurrentes de pagos periódicos.

Canales

Los denominados canales de distribución explican cómo la empresa se comunica con los distintos segmentos del mercado, para llegar a ellos y proporcionarles una propuesta de valor. Las preguntas recurrentes son ¿qué canales prefieren actualmente el segmento de clientes?¿Cuáles tiene mayor resultado? ¿Cuáles son más rentables? ¿Cómo se integran en las actividades diarias de los clientes?. Generalmente se clasifican por dos tipos:

- **Directos**: puede ser un equipo comercial o un sitio web.
- **Indirectos**: generalmente son tiendas establecidas el cual se hacen distribuciones al por mayor/menor, o ventas mayoristas.
- **Información**: ¿cómo se da a conocer el servicio?
- **Evaluación**: ¿cómo ayudar a que evalúen la propuesta de valor?
- **Compra**: ¿cómo los clientes compran el servicio?
- **Entrega**: ¿cómo se entrega la propuesta de valor a los clientes?
- **Postventa**: ¿cómo será el servicio post-venta de atención que se ofrecerá?

Relación con los clientes

Se define el tipo de relación que se establecen con cada segmento del mercado. Generalmente son relación tipo personal y/o automatizado. Estos se dividen en:

- **Asistencia personal**: es decir, la interacción humana.
- **Asistencia personal exclusiva**: interacción humana exclusiva a un cliente determinado.
- **Autoservicio**: solo se limita a entregar los medios para que el cliente se atienda el mismo.
- **Servicios automáticos**: de igual manera que autoservicio, agregando tecnología que facilite su trabajo.
- **Comunidades**: se basan en que se facilite el contacto entre comunidades de modo que genere un valor.

- **Creación colectiva**: esto se traduce a entregar un medio el cual usuarios/clientes vayan creando un producto/servicio que genere valor. Ej: calificaciones de productos en tiendas online.

Fuentes de ingresos

La fuente de ingresos es el flujo de caja que genera la empresa en los diferentes segmentos de mercado. Principalmente se diferencian en transacciones de pagos puntuales, es decir, la venta de un producto directo; y por otro lado existen los ingresos recurrentes de pagos periódicos, como su nombre lo dice, se basan en rentas en la que los clientes deben pagar periódicamente. Los tipos de fuentes de ingresos son:

- **Venta de activos**: venta de un producto físico.
- **Cuota por uso**: mientras más se usa un servicio determinado, más paga el cliente.
- **Cuota de suscripción**: acceso ininterrumpido a un producto o servicio.
- **Préstamo/alquiler/leasing**: se basa en una concesión temporal, a cambio de una tarifa.
- **Concesión de licencias**: la concesión de utilizar una propiedad intelectual.
- **Gastos de corretaje**: servicios de intermediación realizados en nombre de dos o más personas.
- **Publicidad**: como su nombre lo dice, es el resultado de las cuotas por publicidad de un producto o servicio.

Recurso clave

Se definen básicamente como los activos más importantes para que un modelo de negocios funcione. Se clasifican en 4 tipos, estos son:

- **Físicos**: edificios, vehículos, máquinas, sistemas.
- **Intelectuales**: información privada, patentes, etc.
- **Humanos**: científicos expertos, desarrolladores, entre otros.
- **Económicos**: En el caso de que se requiera explícitamente dinero en efectivo.

Actividades clave

Las actividades claves se definen como las acciones más importantes que debe aprender la empresa para que su modelo de negocios funcione. Se separan en:

- **Producción**: esta actividad está relacionada con el diseño, la fabricación y la entrega de un producto en grandes cantidades o con una calidad superior. Este tipo de actividad está relacionada principalmente en las empresas de fabricación.
- **Resolución de problemas**: este tipo de actividades implican la búsqueda de soluciones nuevas a los problemas individuales a cada cliente, por ejemplo las empresas consultoras.
- **Plataforma/red**: son los modelos de negocios diseñados con una plataforma/software como recurso clave.

Asociaciones clave

En esta sección se describe la red de proveedores y socios que contribuyen al funcionamiento de un modelo de negocios. Las asociaciones clave son de suma importancia, debido a que las empresas necesitan crear alianzas para optimizar el modelo de negocios, reducir los riesgos o adquirir recursos. Existen 4 tipos, estas son:

- **Alianzas estratégicas** entre empresas no competidoras.
- **Competición**: asociaciones estratégicas entre empresas competidoras.
- **Joint Ventures**: son empresas en conjunto para crear nuevos negocios.
- **Relación cliente-proveedor**: para garantizar la fiabilidad de los suministros.

Estructura de costes

La estructura de costes son todos los costes que implica la puesta en marcha de un modelo de negocios. Existen ciertos enfoques, donde se pone énfasis a un coste o valor determinado; se pueden diferenciar:

- **Según costes**: recortar costos como sea posible, para propuestas de valor de bajo precio.
- **Según valo**r: en este caso no importan los costos, solo importa generar una propuesta de valor *premium*.

Es importante destacar que en la estructura de costos se deben definir:

- **Costes fijos**: son todos los costos el cual no varía el volumen de transacciones o ventas.
- **Costes variables**: son los costos el cual varía según el volumen de transacciones o ventas.

2.2.3. Metodología Ágil

Las metodologías ágiles son aquellas que permiten adaptar la forma de trabajo a las condiciones del proyecto, consiguiendo flexibilidad e inmediatez en la respuesta para amoldar el proyecto y su desarrollo a las circunstancias específicas del entorno.

Esta metodología mejora la satisfacción al cliente dado que se involucra y compromete a lo largo de todo el proyecto. En cada etapa se informará al cliente de los logros y procesos del mismo, con la visión de involucrarse directamente para sumar su experiencia y conocimiento, y así, optimizar las características del producto final.

Metodologías Ágiles más utilizadas

- **Extreme Programming (XP)**: esta herramienta es muy útil sobre todo para startups o empresas que están en proceso de consolidación, puesto que su principal objetivo es ayudar en las relaciones entre los empleados y clientes. La clave del éxito es potenciar las relaciones personales, a través, del trabajo en equipo, fomentando la comunicación y eliminando los tiempos muertos.

- **SCRUM**: se caracteriza por ser la “metodología del caos” que se basa en una estructura de desarrollo incremental, esto es, cualquier ciclo de desarrollo del producto y/o servicio se desgrana en “pequeños proyectos” divididos en distintas etapas: análisis, desarrollo y testing. En la etapa de desarrollo se encuentran las interacciones del proceso o Sprint, es decir, entregas regulares y parciales del producto final. Esta metodología permite abordar proyectos complejos que exigen una flexibilidad y una rapidez esencial a la hora de ejecutar los resultados. La estrategia irá orientada a gestionar y normalizar los errores que se puedan producir en desarrollos demasiado largos, a través de, reuniones frecuentes para asegurar el cumplimiento de los objetivos establecidos.

- ***Agile Inception***: Está orientada a la definición de los objetivos generales de las empresas. Su meta es clarificar cuestiones como el tipo de cliente objetivo, las propuestas de valor añadido, las formas de venta. Suele girar en torno al método de “elevator pitch“, que consiste en pequeñas reuniones entre los socios y el equipo de trabajo en las que las intervenciones no pueden superar los 5 minutos.

- **Kanban**: La estrategia Kanban conocida como ‘Tarjeta Visual” muy útil para los responsables de proyectos. Esta consiste en la elaboración de un cuadro o

diagrama en el que se reflejan tres columnas de tareas; pendientes, en proceso o terminadas. Este cuadro debe estar al alcance de todos los miembros del equipo, evitando así la repetición de tareas o la posibilidad de que se olvide alguna de ellas. Por tanto, ayuda a mejorar la productividad y eficiencia del equipo de trabajo.

2.3 INGENIERÍA DE *SOFTWARE*

2.3.1. Definición

La ingeniería de *software* es una disciplina formada por un conjunto de métodos, herramientas y técnicas que se utilizan en el desarrollo de los programas informáticos. El ingeniero de *software* se encarga de toda la gestión del proyecto para que éste se pueda desarrollar en un plazo determinado y con el presupuesto previsto.

Por lo tanto, la ingeniería de *software* incluye el análisis previo de la situación, diseño del proyecto, desarrollo del software, las pruebas necesarias para confirmar su correcto funcionamiento y la implementación del sistema.

2.2.2. Levantamiento de requerimientos

Un requerimiento es una característica que el sistema “debe” tener o es una restricción que el sistema “debe” satisfacer para ser aceptada por el cliente. Por lo tanto un levantamiento de requerimientos es la especificación del sistema en términos que el cliente entienda, de forma que se constituya en el contrato entre el cliente y los desarrolladores.

2.2.2.1 Requerimientos funcionales

Describen la interacción entre el sistema y su ambiente, independiente de su implementación. El ambiente incluye al usuario y cualquier otro sistema externo que interactúa con el sistema.

2.2.2.2 Requerimientos no funcionales

Son restricciones sobre el espacio de posibles soluciones. A diferencia de los requisitos funcionales el cual definen qué debe hacer un sistema, los requisitos no funcionales definen cómo debe ser un sistema.

Dentro de los requerimientos no funcionales están:

- **Disponibilidad**: disposición del sistema para prestar servicios correctamente.
- **Extensibilidad**: define la facultad de flexibilidad para el cambio que posee un programa o aplicación determinada.
- **Mantenibilidad**: posibilidad de realizar modificaciones o reparaciones a un proceso sin afectar la continuidad del servicio.
- **Portabilidad**: capacidad de transferencia de un producto de software de un entorno a otro. El entorno puede ser de tipo organizacional, *hardware* o *software*.
- **Velocidad de desarrollo**: es la velocidad que mide la facilidad de agregar funcionalidades al software.
- **Resiliencia**: es la capacidad que tiene un *software* de adaptación en ciertos momentos de riesgos y problemas tanto en el negocio como en el software mismo.
- **Tiempo de respuesta**: como su nombre lo indica, es el tiempo de respuesta que tiene el software a la interacción del usuario.
- **Reutilización**: es la habilidad para poder reutilizar código de programación en distintas funcionalidades.
- **Robustez**: un *software* robusto es aquel que puede ejecutar diversos procesos de manera simultánea sin generar fallos.
- **Escalabilidad**: se refiere al aumento de la capacidad de trabajo o de tamaño de un sistema sin comprometer el funcionamiento y calidad normales del mismo.
- **Seguridad**: son las capacidades que debe tener un sistema para cumplir atributos en la seguridad de los datos, seguridad lógica, control de acceso de información, autenticidad, privacidad, entre otros.
- **Usabilidad**: es el esfuerzo que necesita hacer un usuario para aprender, usar, ingresar datos e implementar los resultados obtenidos de un software.

2.2.2.3 Casos de uso

Describen el modo en que un actor interactúa con el sistema (descripción de un rol en lenguaje natural), el cual, narran el comportamiento dinámico del sistema desde el punto de vista concreto (el del actor). En general se pueden expresar tanto requerimientos funcionales como no funcionales. En general, se tiene varias formas de representación, sin embargo, en este documento se utilizará el “Diagrama de casos de uso”.

El Diagrama de casos de uso es una forma diagramada de comportamiento UML[15] mejorado, define una notación gráfica para representar casos de uso.

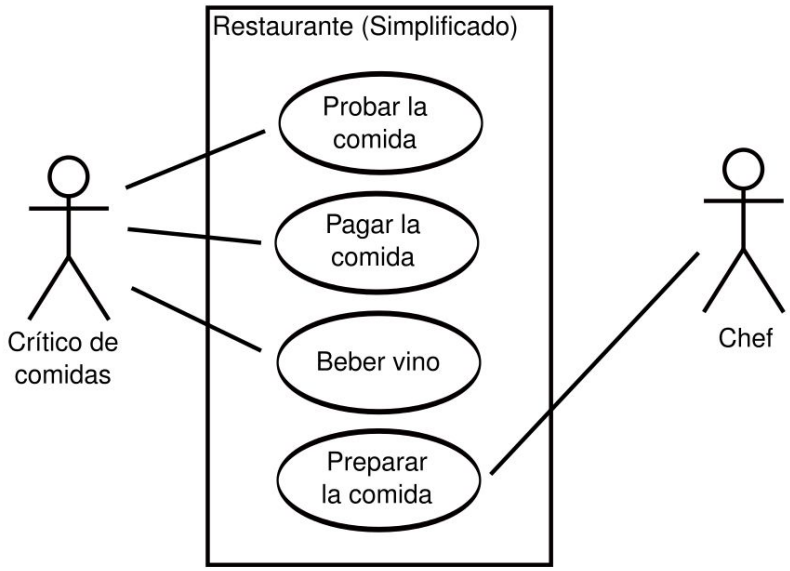


Figura 2.3: Ejemplo de un diagrama de casos de uso. Fuente: “elaboración propia”.

La Figura 2.3 muestra un ejemplo de diagrama de casos de uso, en este caso los actores que interactúan con el sistema son “Crítico de comidas” y “Chef”. El rectángulo representa el sistema “Restaurante” y los óvalos representan las acciones del sistema, en este caso “Probar la comida”, “Pagar la comida”, “Beber vino”, “Preparar la comida”.

2.4 HEURÍSTICAS DE NIELSEN

Hoy en día existen muchos tipos de test y pruebas para evaluar la usabilidad y asegurar que los diseños ofrezcan una buena experiencia de usuario. Una de las más conocidas y utilizadas son las 10 reglas heurísticas del apodado “padre” de la usabilidad, Jakob Nielsen.

[15] UML son las siglas de “Unified Modeling Language” o “Lenguaje Unificado de Modelado”. Se trata de un estándar que se ha adoptado a nivel internacional por numerosos organismos y empresas para crear esquemas, diagramas y documentación relativa a los desarrollos de software (programas informáticos).

A continuación se hacen un resumen de estas 10 heurísticas:

1. **Visibilidad**: explica al usuario cuál es el estado del sistema en cada momento, y mantenle informado de lo que está pasando.

2. **Relación con la realidad**: utiliza un lenguaje familiar y apropiado para los usuarios a los que te diriges, y organiza la información con un orden natural y lógico.

3. **Control y libertad**: ofrece funciones de rehacer y deshacer que permitan al usuario tener el control de sus interacciones con libertad.

4. **Consistencia y estándares**: establecer una convenciones lógicas y mantenerlas siempre (mismo lenguaje, mismo flujo de navegación...)

5. **Prevención de errores**: ayuda a los usuarios a evitar equivocarse antes de que cometan el error.

6. **Reconocimiento**: haz visible todo lo que sea posible, no esperes que los usuarios recuerden o memoricen información, muéstrala si es necesaria en el proceso, las instrucciones deben estar a la vista cuando sea necesario.

7. **Flexibilidad:** permite que el sistema pueda adaptarse a los usuarios frecuentes, diseña la realización de tareas avanzadas de manera fluida y eficiente.

8. **Estética y minimalismo**: muestra sólo lo necesario y relevante en cada situación, no distraigas al usuario con información extra poco relevante.

9. **Recuperarse de los errores**: ayuda a los usuarios a reconocer y corregir sus errores, indica siempre el problema concreto que está ocurriendo y sugiere soluciones constructivas.

10. **Ayuda y documentación**: la información de ayuda debe ser breve, concisa, fácil de buscar y enfocada a las tareas del usuario.

CAPÍTULO 3: DISEÑO DE LA SOLUCIÓN

3.1. CONTEXTO

En el Capítulo 1 se presentan los grandes problemas que conlleva un software que ha pasado por la Feria de *Software*, luego en el Capítulo 2 se investigaron alternativas y metodologías para la construcción de la solución. Este capítulo se centra en la reconstrucción del proyecto, usando los conocimientos adquiridos de reingeniería y metodologías para la construcción de startups. Es importante mencionar que todo el diseño de la solución se basa en metodologías ágiles para startups.

Como se explica en apartados anteriores, el objetivo principal de este documento es la reconstrucción de un proyecto el cual haya usado el proceso de la Feria de *Software*. Actualmente los ingenieros de software solucionan este problema aplicando directamente la reingeniería, sin embargo para una startup no es suficiente este proceso, ya que no se consideran variables importantes como el cambio de requerimientos por parte de los clientes, el cambio y/o descubrimiento de nuevas necesidades del mercado, propuesta de valor diferente, entre otros elementos que repercutan en el modelo de negocios y por consiguiente al proyecto en cuestión.

En este apartado se propone una solución basado en tres pilares fundamentales:

- El modelo de negocios: se busca replanteamiento del modelo de negocios utilizando principalmente la metodología CANVAS.

- Reingeniería: una vez adquirido los conocimientos del propósito del proyecto, se descomponen los requerimientos funcionales y no funcionales del proyecto; luego se analiza las diferencias de estos requerimientos entre ambos proyectos, finalmente se busca conocer la distancia (es decir, que tan lejos está el proyecto anterior con respecto a la nueva propuesta) y proponer la mejor forma de llegar a ella.

- Arquitectura de software: utilizando dos procesos anteriores, este apartado propone una arquitectura de software que cumpla con los requerimientos funcionales y no funcionales. Se debe llegar con una propuesta de desarrollo clara para que en el siguiente capítulo comience la construcción de esta.

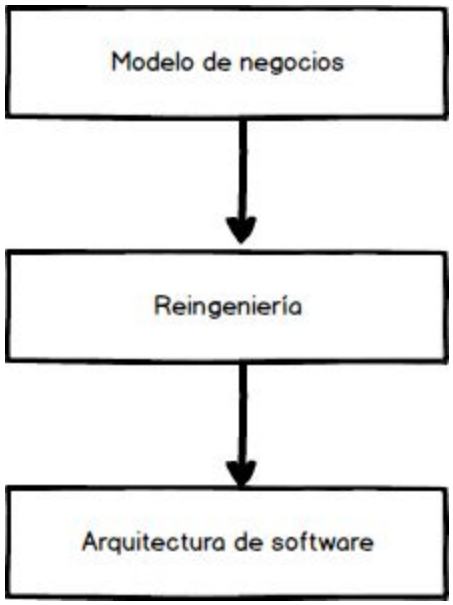


Figura 3.1: Pasos para la solución propuesta. Fuente: “elaboración propia”.

3.2. MODELO DE NEGOCIOS

Esta sección propone un modelo de negocios basados en el mercado actual; se busca conocer una nueva propuesta de negocios basados en los conocimientos adquiridos en el anterior proyecto. La metodología a utilizar es CANVAS, ya que es lo más cercano y simple para una Startup.

3.1.1. Idea del proyecto Smatch

El proyecto viene a resolver las problemáticas que existen en los eventos deportivos. Armar un partido en donde participan muchas personas es una tarea que toma bastante tiempo y paciencia por parte de los organizadores, por ejemplo, en un partido de fútbol se requiere que 12 personas se pongan de acuerdo a escoger el lugar, la hora en que todos puedan, etc. Muchas veces existen partidos que se cancelan debido a que personas se bajan del partido y no pueden encontrar reemplazantes.

Es por esto que nace Smatch, una plataforma que se encarga de todo el proceso deportivo, es decir, tanto de la organización como la reserva de las canchas. Esto es una

gran oportunidad para los recintos deportivos, el cual se puede optar porque sus clientes tengan una experiencia enriquecedora no solo al momento de jugar sino en el de organizar.

El proyecto considera dos ejes principales. El primero es solucionar la problemática de la organización de partidos, a través de funcionalidades que permitan la facilidad de ello, como lo es, armar una comunidad de deportistas a través de la aplicación, el cual se puedan comunicar para armar partidos entre ellos o también para encontrar posibles jugadores. Por otro lado, se requiere de armar un sistema de gestión para los recintos deportivos, así puedan tener ellos mismos una experiencia grata al momento de organizar su reservas, tanto presencial, como en línea.

Desde el punto de vista tecnológico, se debe crear una aplicación móvil para la organización de partidos y un sitio web de gestión de reservas para los recintos deportivos.

3.2.2. CANVAS

3.2.2.1. Segmento de mercado

El proyecto considera tanto usuarios como clientes de forma distinta, donde los primeros (usuarios) son las personas que usarán la plataforma, aportando un valor importante en el negocio del proyecto, mientras que los clientes son el perfil el cual se basa nuestro modelo de ingresos, el cual nos entrega valor monetario. A continuación se muestran los perfiles de cada uno de ellos:

- **Usuarios-1:** estos usuarios son jugadores de futbolito, entre 18 y 25 años, que tengan una actividad deportiva regular (al menos una vez por semana), de clase socioeconómica media-alta y con tendencia/disposición a jugar en muchas canchas distintas. Este perfil son los que ayudarán a completar los cupos de jugadores faltantes.

- **Usuarios-2:** los segundos usuarios son bastante parecidos al primero, sin embargo, se diferencian en que su rango etario está entre los 25 y 45 años, es decir, se consideran personas en la etapa más adulta, con trabajos estables. Este perfil se espera que tengan disposición deportiva ocasional (al menos una vez cada dos semanas), pero que contiene un grupo más cerrado de amigos (generalmente grupos de Whatsapp), el cual contienen una cantidad limitada y generalmente justa para armar los partidos. Otra característica importante es el poco tiempo que

tienen para dedicarle a la organización de un partido y que están dispuestos a pagarles a otros jugadores para que puedan completar los cupos faltantes.

- **Clientes-1:** los clientes, son los que las empresas de recintos deportivos de futbolito emergentes, el cual son poco conocidos y/o están en ubicaciones difíciles de encontrar; su principal utilidad son las reservas de canchas. Este perfil generalmente no tiene plataforma web ni los recursos para tener un medio de difusión masivo. Tampoco tienen el pago en línea. La principal necesidad de este tipo de clientes son la de posicionarse en el mercado.

- **Clientes-2**: el segundo tipo de cliente son las empresas de futbolito con más de 10 canchas a su disposición, donde tienen grandes inversiones en logística en las reservas de canchas, además deben gestionar ligas y/o torneos.

3.2.2.2. Propuesta de valor

Para el proyecto, se definen dos diferentes propuestas de valor, separadas por usuario y cliente.

- Los usuarios (jugadores de futbolito) tendrán la posibilidad de armar partidos de manera sencilla, sin requerir de todos los jugadores para completar sus partidos. El principal dolor que tienen los organizadores de partidos es el riesgo de que se caigan los partidos y la cantidad de tiempo que toma organizar. El proyecto toma como propuesta de valor estos enfoques.

- Los clientes (recintos deportivos) tendrán mayor oportunidad de reserva, ofreciendo un canal que se ocupa de toda la logística de organización y reserva. Por lo tanto, el recinto deportivo solo debe preocuparse de sus canchas. Esto les da la oportunidad de mayor penetración de mercado, mayor público, mejora la calidad de sus reservas y reduce los riesgos de pérdidas por cancelación de último minuto.

3.2.2.3. Canales

Para el caso de Smatch se consideraron las siguientes fases del canal:

- **Información**: la idea es usar el mismo canal de comunicación general, basado en las redes sociales y un sitio web promocional. Además se planifica la idea de visitar universidades ya que se concentra gran parte del público objetivo.

- **Evaluación**: la plataforma debe ayudar a observar qué tan bien se están realizando las reservas de canchas, midiendo la cantidad de reservas concretadas de la plataforma.
- **Compra**: para los clientes las compras se harán a través de suscripciones mensuales, donde el medio de venta será principalmente web. Sin embargo, en una primera instancia, se necesitan vendedores presenciales.
- **Entrega**: la propuesta de valor se basa en la entrega de reservas en línea, desde una web que las gestione.
- **Postventa**: la plataforma para los recintos deportivos estará activa 24/7, con vendedores que se encargará de que el cliente pueda tener la mejor experiencia posible con el software.

3.2.2.4. Relación con los clientes

Para este proyecto se considera lo siguiente:

- **Usuarios**: la relación se basa principalmente por *comunidades y creación colectiva*. Ya que para los jugadores el valor está en reunir la gente suficiente para generar el evento deportivo; esto último genera un valor directo al recinto deportivo, debido a que se genera una reserva de cancha.
- **Clientes**: en este caso se considera el *servicio automático*, donde el proyecto se encarga de toda la logística de una organización y reserva de cancha, lo cual se traduce a una serie de herramientas que ayudan a los recintos deportivos a tener mayor utilidad y control de su negocio.

3.2.2.5. Fuentes de ingresos

En particular, para este proyecto se han definido dos posibles fuentes de ingresos:

- **Propuesta 1:** cobrar una comisión por cada vez que se concrete una reserva, es decir, gastos por corretaje. Esto se debe a que la principal propuesta de valor para los jugadores es organizar el partido y hacer la reserva en línea.
- **Propuesta 2**: enfocado a los recintos deportivos, se basa en hacer planes de suscripción enfocado en diferentes segmentos de clientes. El valor que genera el

proyecto se basa en la logística de la organización de todo su negocio, lo que facilita gran parte de su negocio.

El Anexo 1 muestra en más detalle los posibles ingresos con este proyecto.

3.2.2.6. Recursos clave

En el caso del proyecto se requieren de los siguientes recursos:

- **Físicos**:
  - **Servidores**: para el alojamiento en producción del software.
  - **Dominio del sitio web**: se refiere a la compra de las URL[16] que sirve para que los usuarios puedan ingresar al sitio web.
- **Humanos**:
  - **Ingenieros de softwares**: son los encargados del desarrollo, mantención y gestión del software en cuestión.
  - **vendedores**: son las personas que serán la fuerza de venta directa del negocio.

3.2.2.7. Actividades clave

En el caso de Smatch, se definen dos actividades basadas en el tipo Plataforma/Red. La principal actividad se basa en un inicio el desarrollo del producto y sus posteriores actualizaciones constantes. La segunda actividad consiste en servicio posventa la cual Smatch debe proveer a los clientes en todo momento.

3.2.2.8. Asociaciones clave

Se consideran dos asociaciones clave:

- **Servicio de pago en línea**: la transacción más importante para el negocio del proyecto es la reserva en línea de una cancha, por lo que es de suma importancia constatar un buen proveedor que nos garantice el pago en línea, destacando su buena funcionalidad y bajo costo. Esta asociación se considera *relación cliente-proveedor*.

[16] URL es una secuencia de caracteres que se utiliza para nombrar y localizar recursos, documentos e imágenes en Internet.

- **Universidades**: como se describe en el segmento de clientes, los usuarios entre 18 y 25 años son un conjunto importante de personas con mucha disponibilidad y abiertos a jugar con otras personas. Es por esto que la asociación con alguna universidad es clave para una buena base de datos de usuarios.

3.2.8.2: Estructura de costes

En el caso del proyecto Smatch, se enfoca en una estructura de valor centrada tanto en los *costes* como en el *valor*, ya que se deben reducir los costos como sea posible, pero sin dejar de lado la gran experiencia de usuario/cliente que deben recibir. Dicho esto se define la siguiente estructura de costes:

- **Costes fijos**:
  - **Servidores Web**: será donde se aloja el *software***,** por lo tanto requiere de al menos el montado de tres servicios, la API hecha en *Ruby on Rails* junto con los servicios de *React* y *React Native.*
  - **Ingenieros de software**: para el desarrollo de el proyecto se requieren de al menos dos ingenieros de software (uno encargado de React y el otro encargado de Ruby on Rails).
  - **Publicidad y fuerza de venta**: se necesitan los servicios de publicidad tanto presencial como online para la venta del servicio. Además se requiere de al menos un vendedor que pueda hacer el trato con los recintos deportivos.
  - **Dominios (.cl) y costos por App/Play store**: estos costos se basan en los pagos por mantener en el internet los servicios web y móvil respectivamente.
- **Costes variables**:
  - **Costo de transacciones por pago en línea**: el principal valor que tiene el proyecto es hacer el pago en línea de las reservas desde la aplicación. Por lo tanto se requiere de algún servicio que pueda facilitar esta funcionalidad al sistema.

En el Anexo 1 muestran los detalles del flujo de caja de estos valores.

3.2.9. Diferencias entre el proyecto inicial

Como se observa en la Sección 3.2 el proyecto Smatch plantea solucionar las problemáticas que existen en la organización de partidos y en la gestión de reservas de

canchas. Desde este punto de vista, existe una gran similitud con el proyecto anteriormente solucionado. Sin embargo existen ciertas diferencias en:

1. **Propuesta de valor**: el principal valor para generar ingresos en el proyecto anterior fue en que se tenía la hipótesis de que se aumentaría la cantidad de reservas en los recintos deportivos asociados. Esto es refutado en este estudio, debido a que los recintos deportivos tienen horarios en los cuales siempre venden (horarios tarde) y el proyecto no asegura que pueda conseguir reservas en los horarios poco comunes. En este estudio se logra percibir la real propuesta de valor del proyecto, el cual es que los jugadores tengan la mejor experiencia al momento de practicar algún deporte colectivo: esto se puede generar una buena oportunidad de mercado para el mercado de canchas, ya que se puede aumentar la fidelización con sus jugadores, encontrar nuevas oportunidades de ventas con nuevos jugadores y tener un canal de ventas online el cual el proyecto se hace cargo las 24 horas del día.

2. **Tecnología detrás de la idea**: la gran diferencia que existen entre estos dos proyectos es la tecnología con la que se construyó versus la nueva propuesta tecnológica planteada en este documento. Como el proyecto anterior fue hace 5 años atrás, en ese momento las aplicaciones móvil no gozaban de gran popularidad, por lo que se pensó hacer un sitio totalmente web. Sin embargo, hoy en día las organizaciones de partidos se hacen a través del teléfono, por lo que es indispensable entrar en dicha tecnología.

3.3. REINGENIERÍA

En la Sección 3.1 se logra definir el problema que percibe el mercado y la propuesta de valor que ofrece la idea de negocios. En este extracto se generan los requerimientos funcionales y no funcionales, se revisa a nivel de software qué tan lejos está el proyecto anterior con la nueva propuesta de valor. Dicho esto, se resuelve mediante un método de reingeniería el cual se extraen los requerimientos funcionales y no funcionales del proyecto desarrollado en la Feria de *Software*. Finalmente esta sección concluye con una propuesta de desarrollo.

3.3.1. Requerimientos no funcionales: Proyecto anterior

Tal como muestra la Figura 2.1, la reingeniería tiene dos subsecciones, la ingeniería inversa y la directa. Para obtener los requerimientos no funcionales se hace una revisión

del software, se analizan los atributos de calidad del software y el nivel de importancia que se tenía para ese entonces.

"Smatch antiguo" se basa en una plataforma web hecho en Ruby on Rails 2.0.0 monolítico, que contiene funcionalidades tanto de los jugadores como de los recintos deportivos. Los requerimientos no funcionales que tiene este software se basan principalmente en velocidad de desarrollo, robustez y disponibilidad; ya que por motivos de la Feria de Software debía hacerse con mucha velocidad en su implementación para lograr las metas que requería la asignatura. Por el lado de la robustez, el proyecto tuvo que albergar todos los casos de usos planificados a lo largo de un año, para llegar al final del ramo con un proyecto muy completo en cuanto a funcionalidades. Finalmente la disponibilidad, es algo intrínseco de la idea del proyecto, esto se debe a que Smatch es una plataforma abierta a todos los deportistas y recintos deportivos, por lo que se requiere la mayor disponibilidad posible para que los clientes tengan una grata experiencia de usuario, además que puedan manejar su negocio con la plataforma.

La Tabla 3.1 muestra una síntesis de los requerimientos no funcionales del proyecto anterior basados en una escala del 1 al 10 en nivel de importancia para el proyecto en cuestión. Los bordeados con color rojo son los puntos más importantes al momento de desarrollar el software, cuando su nivel de importancia supera el 7, se considera un requerimiento no funcional crítico. La información obtenida en esta tabla se basan en documentos hechos por la asignatura, investigación de la plataforma y su arquitectura compuesta.

3.3.2. Requerimientos no funcionales: Comparación con las nuevas necesidades

Hace unos años atrás las redes sociales y la comunicación a través de internet se basaba en plataformas web, hoy en día existe un mayor abanico de opciones de dispositivos comandados por los el mobile. Por lo tanto los jugadores se organizan a través de mayores medios y el proyecto debe estar a la par con los nuevos dispositivos que estén saliendo al mercado. Este punto es muy importante y se considera crítico para el proyecto en cuestión. Otro elemento importante para el nuevo proyecto es la escalabilidad del software; el método de desarrollo de la mayoría de startups se basa en la metodología ágil, es decir en pocas palabras, desarrollar pocas funcionalidades y validar mucho, por lo tanto el software debe estar adaptable sin perder la calidad. Por lo anterior, se define la Tabla 3.2, que representa los requerimientos no funcionales considerando el mercado actual.

Requerimiento no funcionales	Importancia
Disponibilidad	10
Velocidad de desarrollo	10
Robustez	10
Usabilidad	8
Extensibilidad	5
Tiempo de respuesta	5
Seguridad	5
Resiliencia	4
Mantenibilidad	3
Escalabilidad	3
Portabilidad	2
Reutilización	2

Tabla 3.1: Requerimientos no funcionales del proyecto anterior. Fuente: “elaboración propia”.

Requerimiento no funcionales	Importancia
Extensibilidad	10
Escalabilidad	10
Mantenibilidad	9
Portabilidad	9
Velocidad de desarrollo	9
Usabilidad	8
Disponibilidad	7

Resiliencia	6
Seguridad	6
Tiempo de respuesta	5
Reutilización	5
Robustez	3

Tabla 3.2: Requerimientos no funcionales del modelo de negocios actual. Fuente: "elaboración propia".

Requerimientos no funcionales	Diferencia de nivel de importancia
Portabilidad	7
Escalabilidad	7
Mantenibilidad	6
Extensibilidad	5
Reutilización	3
Resiliencia	2
Seguridad	1
Tiempo de respuesta	0
Usabilidad	0
Velocidad de desarrollo	-1
Disponibilidad	-3
Robustez	-7

Tabla 3.3: Diferencias de Requerimientos no funcionales entre el el proyecto anterior y el nuevo. Fuente: "elaboración propia".

Como se muestra en la Tabla 3.3, existen cuatro requerimientos no funcionales con más de 5 puntos de diferencia. Por lo tanto se debe plantear la forma de aumentar la extensibilidad, portabilidad, mantenibilidad y escalabilidad del software. A continuación se revisan los puntos importantes y cómo abordarlos:

- **Extensibilidad**: debido a la versión de la tecnología ocupada en el proyecto anterior y a la robustez del proyecto resulta cada vez más complicado tener extensibilidad en el proyecto, se recomienda hacer un refactor de todo el código y documentarlo.

- **Mantenibilidad**: este punto es muy similar al anterior, donde presenta las mismas dificultades en cuanto a la robustez; por lo que con el software actual es muy difícil aumentar 6 puntos, tal como en el punto anterior, se recomienda hacer un refactor del código.

- **Portabilidad**: las arquitecturas monolíticas tienen la ventaja de una velocidad de desarrollo muy rápida pero la gran desventaja de que solo funcionan para una plataforma, en este caso web. Por lo tanto la manera de aumentar 7 puntos al proyectos es cambiar la arquitectura, de tal modo que permita un mayor crecimiento sin depender de una sola tecnología.

- **Escalabilidad**: la escalabilidad se basa en la habilidad en la habilidad del software para reaccionar y adaptarse sin perder su calidad. Las ventajas de la tecnología utilizada, como lo es Ruby, contiene una gran cantidad de gemas que permiten adaptarse rápidamente a nuevas funcionalidades y manteniendo una buena calidad del software. En este caso en particular, es importante mantener un software con la mayor comunidad y mayor estandarización que permita la reutilización de código y rápida implementación.

3.3.3. Requerimientos funcionales: Proyecto anterior

Como bien se describe en el Capítulo 2 los requerimientos funcionales del sistema son todos aquellos que describen cualquier actividad que este deba realizar, en otras palabras el comportamiento o función particular de un sistema o software cuando se cumplen ciertas características. Es de suma importancia comparar las diferencias de los requerimientos funcionales entre ambos proyectos, para poder analizar una propuesta de los solución acorde a la nueva idea de negocios Figura 3.2: Diagrama de casos de uso de los requerimientos funcionales proyecto antiguo

La Figura 3.2 equivale a la lista de requerimientos funcionales generados por la idea del proyecto anterior. Se definen los siguientes actores:

- **Jugador**: se definen como los usuarios, son deportistas que buscan organizar partidos.
- **Administrador de canchas**: son parte de los clientes, el cual son los funcionarios el cual se encarga de la administración de las canchas.
- **Dueño del Recinto**: como su nombre lo dice, son los dueños de los recintos deportivos, buscan datos generales para poder gestionar su negocio.
- **Administrador**: se hace mención a un mantenedor del sistema que tenga una interfaz de crear recintos deportivos e ingresarlos al sistema.

Figura 3.2: Diagrama de casos de uso del proyecto anterior. Fuente “elaboración propia”.

La Tabla 3.4 describe los requerimientos funcionales clave del proyecto anterior.

Requerimientos	Descripción
Crear Partido	Tarea que tiene como objetivo crear un evento deportivo.
Buscar partido	Tarea que tiene como fin buscar eventos deportivos que necesitan jugadores para llevarse a cabo. el jugador envía una solicitud para unirse al partido.
Escoger recinto deportivo	Tarea cuya función es buscar los recintos más cercanos partir de la localización GPS o IP del usuario.
Armar alineación del equipo.	Tarea que tiene como objetivo armar el equipo con los amigos que se desea jugar, a los cuales se le envia una invitacion ya sea por medio dentro del sitio o por facebook.
Reservar y pagar la cancha	Una vez escogida la cancha, se puede reservar mediante el pago completo o una cuota del total de la cancha.
Administrar canchas.	Tarea que es utilizada por el usuario “administrador de canchas”el cual podrá ver todos los partidos agendados en sus canchas y ver los pagos de estas.

Tabla 3.4: Tabla de descripción de los requerimientos funcionales del proyecto anterior.

Cabe mencionar que el enfoque de negocios es muy similar a la propuesta de negocios ofrecida en el capítulo 3. Sin embargo, el enfoque de negocios es para recintos deportivos populares y con muchas canchas asociadas.

La Figura 3.3 representan los casos de usomás importantes consideradas en el proyecto nuevo. A diferencia de la Figura 3.2, se definen solo dos actores; esto es, porque la nueva idea tiene un enfoque más general del proyecto, debido a que solo es una lista de requerimientos no validados. Por lo tanto, para el caso de un proyecto startup, se debe tener una idea lo más general posible para que no existan problemas técnicos y que se puedan cambiar estos requerimientos fácilmente.

Los actores se definen como:

> **Jugador**: definido como el usuario del sistema, su principal dolor es la organización y reservas de partidos. Es la entidad de un deportista que busca la generación y/o participación de un evento deportivo.

Recintos deportivos: definido como el cliente del sistema, es la entidad que representa el sistema como los que gestionan los reservas de canchas.

3.3.4. Requerimientos funcionales: Proyecto Nuevo

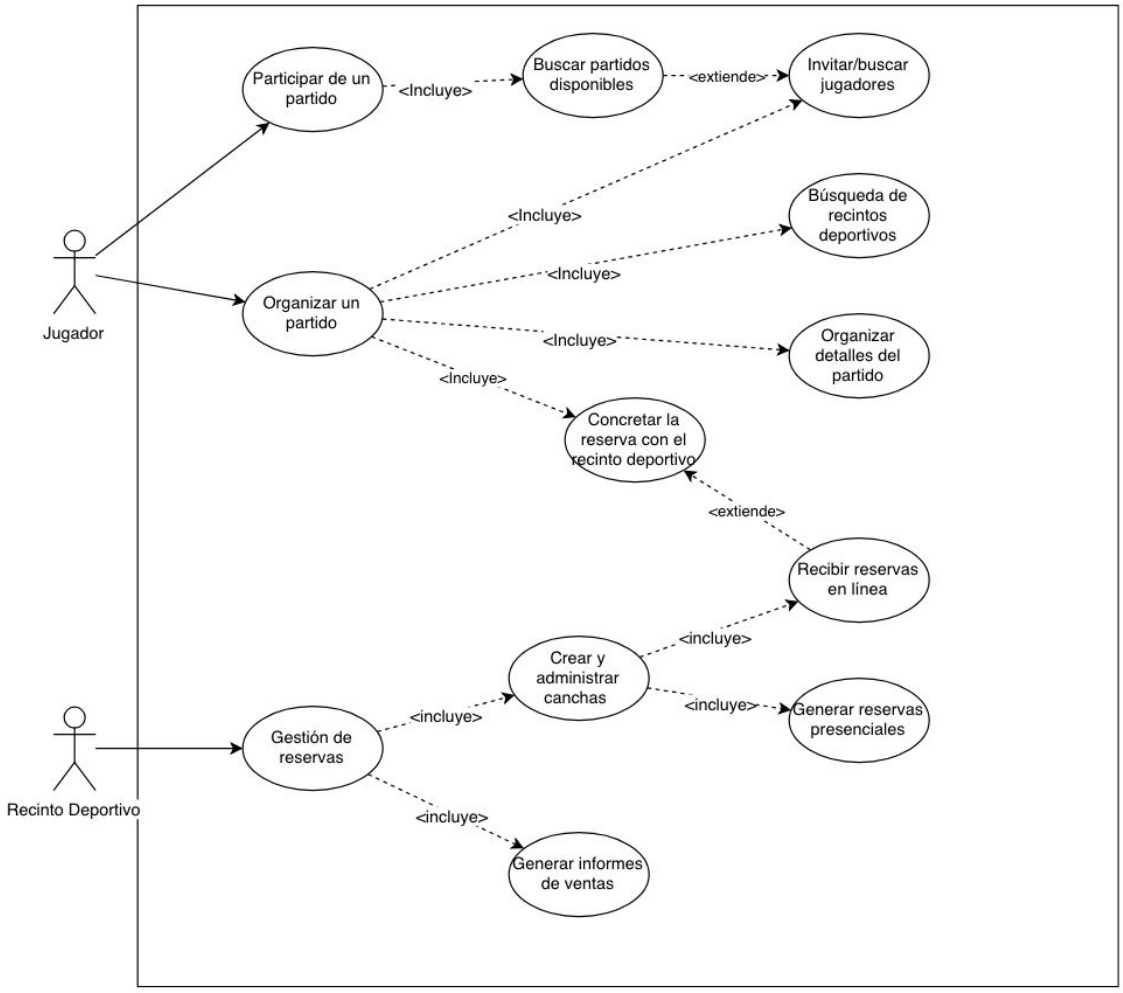


Figura 3.3: Diagrama de casos de usos de los requerimientos funcionales del nuevo proyecto. Fuente “elaboración propia”.

Una vez presentada la lista de requerimientos funcionales, se puede concluir que no existe una diferencia considerable entre ambos proyectos, sin embargo, los nuevos requerimientos funcionales logran tomar las funcionalidades más importantes observadas

en el proyecto anterior y generalizar dichos requerimientos en ideas más globales, de este modo se podrá tener una mayor resiliencia al momento de la validación del mercado.

Requerimientos	Descripción
Organización	Este requerimiento se enfoca en la necesidad de un jugador para crear eventos deportivos, este requerimiento contiene la búsqueda de recintos deportivos, reserva de la cancha, búsqueda de jugadores y organización del evento mismo.
Participación	Su función es que un jugador que quiera jugar un partido tenga la opción de buscar distintos partidos disponibles y participar de ellos. Esta acción puede incluir o no un pago por su participación.
Gestionar	Tarea general para los recintos deportivos. Busca dar una lista de funcionalidades que se encarguen de la organización de reservas tanto presenciales como en línea. Además incluye la configuración de bloques de canchas, asignarles precios y disponibilidad.

Tabla 3.5: Tabla de descripción de los requerimientos funcionales del nuevo proyecto

3.3. ARQUITECTURA DEL NUEVO SOFTWARE

En el Capítulo 3.2 se pudo observar las diferencias que existen entre el proyecto anterior y el nuevo, a través de los requerimientos funcionales y no funcionales. Este apartado hace un análisis de lo anterior y propone una nueva arquitectura de software junto a los requerimientos funcionales por el cual se debe partir por validar.

3.3.1. Arquitecturas de software posibles

Para la elección de la arquitectura de software se considera principalmente los requerimientos no funcionales, por sobretodo elegidos como críticos, debido a que se define como la principal distancia entre el proyecto anterior y el nuevo para llegar a a la adaptación al mercado. A continuación se presentan tres propuestas de arquitectura:

- **Arquitectura Monolítica:** esta arquitectura se destacó por su velocidad de desarrollo, sin embargo carece en todos los objetivos de la empresa, ya que pierde mucha escalabilidad y extensibilidad, puntos cruciales para el modelo de negocios y el proceso iterativo que se busca.
- **Arquitectura MVVM:** a diferencia de la arquitectura monolítica esta arquitectura permite separar las interfaces de jugadores y recintos deportivos, ajustándose a la necesidad del negocio. Además permite la separación de una API de datos el cual

abstrae lo visual con la lógica del negocio. Esta arquitectura se ajusta a los requerimientos no funcionales.

- **Arquitectura de Microservicios:** finalmente, la arquitectura de microservicios permite absraerse aún más, sobretodo en la lógica del negocio, permitiendo separaciones entre servicios de plataformas. Este tipo de arquitectura si bien, es técnicamente superior al MVVM, su configuración y desarrollo es más complejo que el MVVM, por lo tanto su velocidad de desarrollo se ve disminuida. Otro punto importante es que la metodología ágil hace que aún no se tenga del todo claro los servicios que se van a trabajar ya que se deben pasar por un proceso de validación.

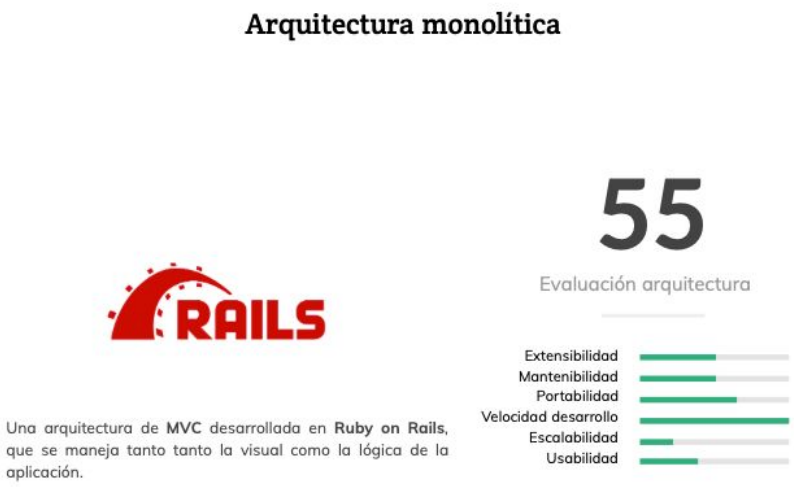


Figura 3.4: Arquitectura monolítica basada en Ruby on rails. Fuente “elaboración propia”.

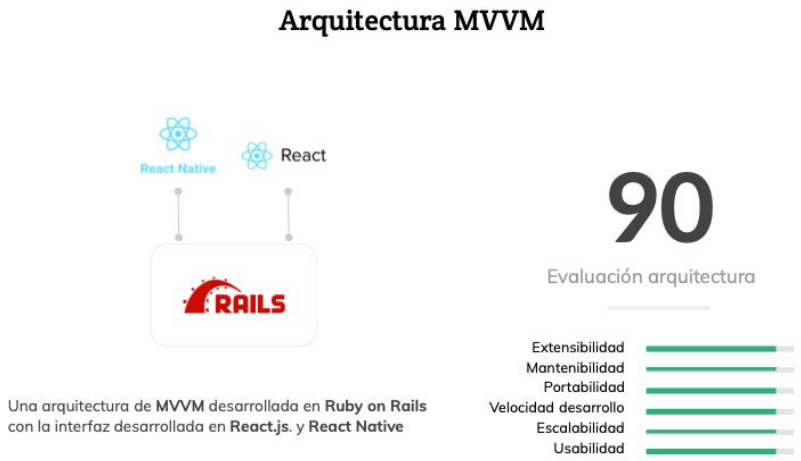


Figura 3.5: Arquitectura MVVM basada en Ruby on rails, React js y React native. Fuente "elaboración propia"

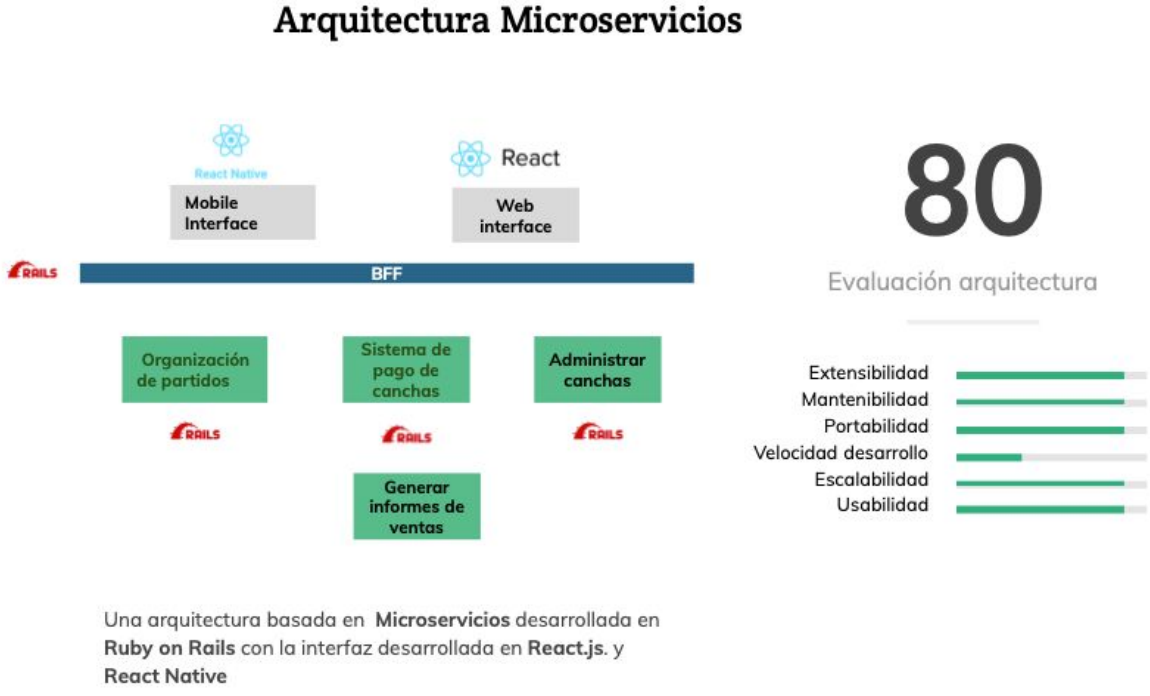


Figura 3.6: Arquitectura Microservicios basada en Ruby on rails, React js y React native. Fuente "elaboración propia".

3.3.2. Propuesta de desarrollo

Debido a la explicación anterior, se propone trabajar con la arquitectura MVVM, debido a su gran ajuste tanto para el mercado como para los requerimientos no funcionales. Sin embargo, se propone analizar la posibilidad de cambio para microservicios en un futuro. La ventaja de trabajar en MVVM es justamente porque luego se puede adaptar a microservicios fácilmente ya que solo se debe cambiar la lógica del negocio, es decir, pasar la API de MVVM a una arquitectura de microservicios.

3.3.2.1 Historias de usuario

Para el desarrollo de las historias de usuario, se basa directamente en los requerimientos funcionales y en las necesidades del mercado. Smatch busca crear un desarrollo y validación para el desarrollo de la organización de jugadores y otra para la gestión de las canchas. Por lo tanto se consideran cuatro etapas esenciales:

- **Etapa 0:** Considera la primera validación de mercado el cual se propone crear un mínimo producto viable para los jugadores que organizan partidos:
    - **Montado inicial de la arquitectura**: busca la creación de la arquitectura MVVM y sus conexiones.
    - **Listado de Recintos deportivos**: para la aplicación en react native, el mínimo valor viable para los jugadores es que puedan tener una interfaz donde puedan tener un catálogo de recintos deportivos que los ayude a la organización de su partido.
- **Etapa 1:** Esta etapa considera la segunda validación de mercado con los recintos deportivos, se busca un mínimo producto viable para el segmento de las canchas, el cual puedan ayudar a la gestión de su negocio.
    - **Sistema de inicio de sesión de recintos deportivos**: se requiere que los recintos deportivos puedan contar con cuentas para la administración de su negocio.

- **Sistema de creación de canchas**: en esta historia se busca que los recintos deportivos puedan tener un sistema de CRUD de canchas, incluyendo sus horarios disponibles y precios.

- **Sistema de organización de partidos presenciales**: sistema que permite el registro de reservas presenciales de cada cancha.

- **Etapa 2**: La siguiente etapa busca generar el valor principal para los jugadores, la organización de un partido. Donde lo jugadores podrán tener su cuenta de usuario, invitar jugadores o buscar jugadores disponibles.

  - **Sistema de Login de usuarios:** se requiere el login de usuarios para poder vincularse con más cuentas asociadas.

  - **Plataforma de organización de canchas**: esta historia desarrolla la funcionalidad de organizar un partido, que permita la invitación y confirmación de otros jugadores, establecer fecha y hora del partido, así como los datos del recinto deportivo, etc. Además que permita la recomendación de jugadores.

- **Etapa 3**: Esta última etapa iterativa se busca la conexión entre el recinto deportivo y la organización de partidos, lo que permite la principal funcionalidad de generar reservas en línea.

  - **Aceptar reservas en línea para los jugadores y recintos deportivos**: este punto busca finalizar la organización y gestión en la etapa 1 y 2; para ello se permite que los jugadores puedan pedir la cancha directamente desde la aplicación y que los recintos deportivos puedan aceptarlas a través del sistema de gestión.

  - **Sistema de pago en línea:** el sistema debe permitir que los jugadores puedan cancelar la cancha directamente desde la aplicación y dejarlo a las cuentas de los recintos deportivos.

CAPÍTULO 4: CONSTRUCCIÓN Y VALIDACIÓN DE LA ETAPA CERO.

En este capítulo, se hace el hincapié inicial para este proyecto. Se busca desarrollar la etapa 0, propuesta en el la sección 3.3.2.1 , la que considera el mínimo producto viable. Luego se realiza una validación con los usuarios, para observar si la aplicación logra el mínimo valor para que permita pasar a la etapa 1. Se utiliza la metodología Lean Startup del Capítulo 2.3.2.

4.1. PLANTEAMIENTO

La etapa 0 consiste en principalmente en el montaje inicial de la aplicación y el listado de recintos deportivos, por lo tanto se propone un modelo de datos y vistas asociadas para la solución propuesta.

La etapa 0 contempla:

1. Montado inicial de la arquitectura.
    a. (API) Modelo de datos relacional.
    b. Generar *endpoints.*
2. Listado de recintos deportivos.
    a. Diseño y construcción de la vista y conexión con la API.

4.1.1. Modelo de datos relacional

El modelo de datos contempla la información necesaria para listar recintos deportivos; se compone de 5 tablas:

- ***Company*****:** contiene los datos de un recinto deportivo, para esta etapa solo se define el nombre y el sitio web. Los recintos deportivos tienen por lo menos una o más sucursales asociadas.
- ***Branch*****:** guarda los datos de las sucursales que tiene un recinto deportivo, destacando su ubicación, el cual compete en la historia de usuario *mapa de recintos deportivos*. Las sucursales tienen por lo menos una o más canchas.
- ***Province*****:** contiene los datos de las comunas de Santiago. Ayuda principalmente a la funcionalidad de listado de recintos deportivos, ya que permite hacer un filtrado por comuna. Una comuna tiene una o muchas asociaciones con un recinto deportivo.

- ***Court*:** contiene los datos relevantes de una cancha. Para la etapa 0 no está contemplada en esta entidad, sin embargo se considera la importancia de ella para futuras etapas.
- ***SportType***: son los tipos de canchas posibles dentro del sistema, por ejemplo "futbolito", "baby futbol", etc.

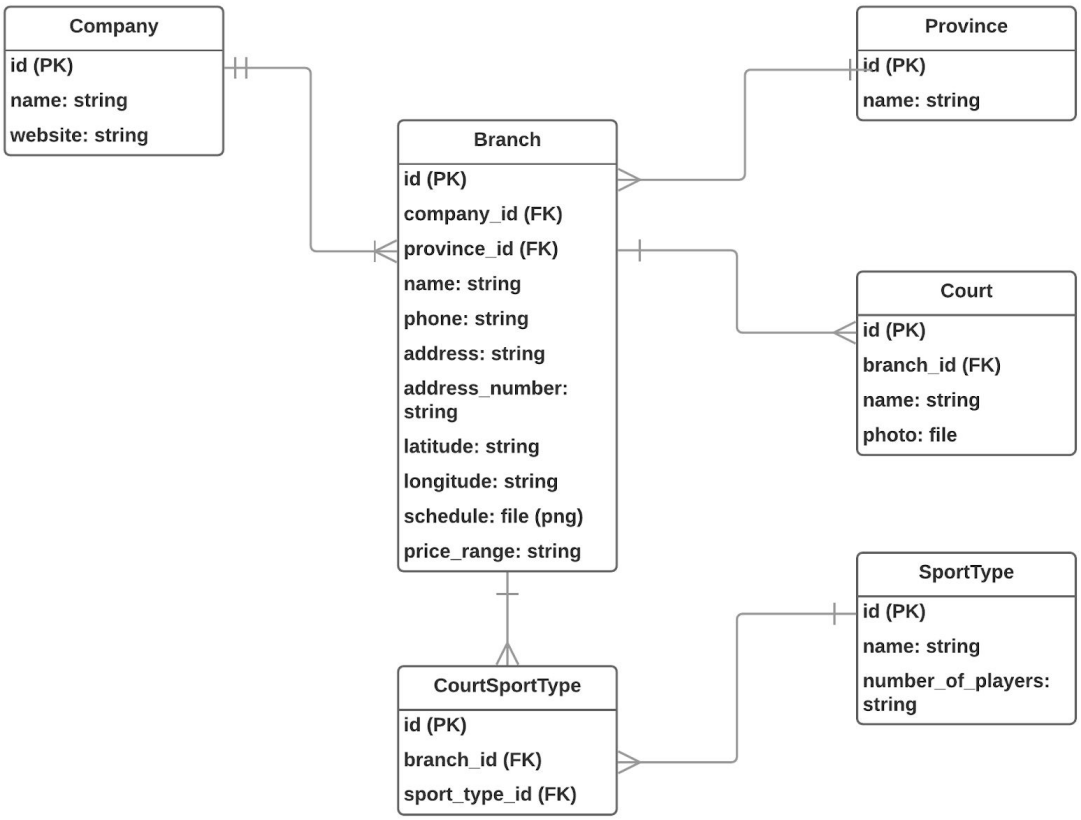


Figura 4.1: Modelo de datos para el desarrollo de la etapa 0. Fuente "elaboración propia".

4.1.2. *Endpoints*

Los *endpoints* son las peticiones que se hacen desde el *frontend* para el rescate de los datos. Para este proceso se recomienda descargar la aplicación Postman[17] la cual permite una interfaz para generar peticiones a la *API* sin necesidad de tener el *frontend* listo.

Para poder generar esto se debe tener en consideración qué es lo que necesita el frontend. Por como se menciona al principio del capítulo, esta etapa se concentra en un

[17] https://www.getpostman.com

listado de recintos deportivos, filtrado por comunas, el cual, al presionar un recinto se despliega una vista de información detallada del recinto.

4.1.2.1. Listado de Recintos deportivos

Para mostrar el listado de los Recintos deportivos se consideran las siguientes URLs:

- Mostrar todos los recintos deportivos:
  - http://localhost:3000/api/branches
- Mostrar todos los recintos deportivos filtrados por comuna:
  - http://localhost:3000/api/branches?province=19


```
{
    "branches": [
        {
            "id": 1,
            "title": "Soccer Pro",
            "address": "Francisco Meneses 1580, Ñuñoa",
            "phone": "(56) 2 2237 9026"
        },
        {
            "id": 2,
            "title": "Soccer Pro",
            "address": "Francisco Meneses 1580, Ñuñoa",
            "phone": "(56) 2 2237 9026"
        }
    ]
}
```


Figura 4.2: Respuesta de la consulta por recintos deportivos. Fuente “elaboración propia”.

La Figura 4.2 muestra la respuesta enviada por la API para que el frontend pueda mostrar sus datos. Se consideran solo el *title*, *address* y *phone* como información preliminar. Para poder ver más información se considera otra vista que muestra el detalle del recinto.

4.1.2.2. Detalle de un Recinto deportivo


```
{
    "branch": {
        "id": 2,
        "title": "Soccer Pro",
        "address": "Francisco Meneses 1580, Ñuñoa",
        "phone": "(56) 2 2237 9026",
        "latitude": "-33.4715762",
        "longitude": "-70.6196143",
        "prices": "Min: $22.000 Max: $34.000",
        "total_courts": 1,
        "sport_types": [
            "Futbolito",
            "Baby Futbol"
        ],
        "schedule": "/uploads/branch/photo/2/1.png",
        "courts": [
            {
                "name": "Cancha 1",
                "sport_types": [
                    {
                        "name": "Futbolito"
                    },
                    {
                        "name": "Baby Futbol"
                    }
                ]
            }
        ]
    }
}
```


Figura 4.3: Respuesta de la consulta por un recinto deportivo en particular. Fuente "elaboración propia"

La Figura 4.3 muestra los datos necesarios para la generación de la vista "detalle de un recinto deportivo". Dicha vista considera la siguiente URL:

- Mostrar un recinto deportivo:
    - http://localhost:3000/api/branches/[:id]

El valor ":id" es el que corresponde a cada recinto deportivo en particular, según lo planteado en el Sección 4.1.1.

La Figura 4.3 muestra la respuesta enviada desde la API para la vista del detalle de un recinto deportivo. Cabe destacar que este *JSON* hace una consulta a todas las tablas involucradas.

4.1.2.3 Listado de Comunas

En la vista de listado de recintos deportivos se necesita un filtro por comunas, por lo que este *endpoint* se genera para que el frontend pueda conocer los tipos de comunas que reconoce la API y luego generar el filtro como se muestra en Sección 4.1.2.1.


```
{
    "provinces": [
        {
            "id": 1,
            "name": "Cerrillos"
        },
        {
            "id": 2,
            "name": "Cerro Navia"
        },
        {
            "id": 3,
            "name": "Conchalí"
        },
        {
            "id": 4,
            "name": "El Bosque"
        },
        {
            "id": 5,
            "name": "Estación Central"
        },
```


Figura 4.4: Respuesta de la consulta por el listado de comunas. Fuente “elaboración propia”.

4.1.3. Vistas

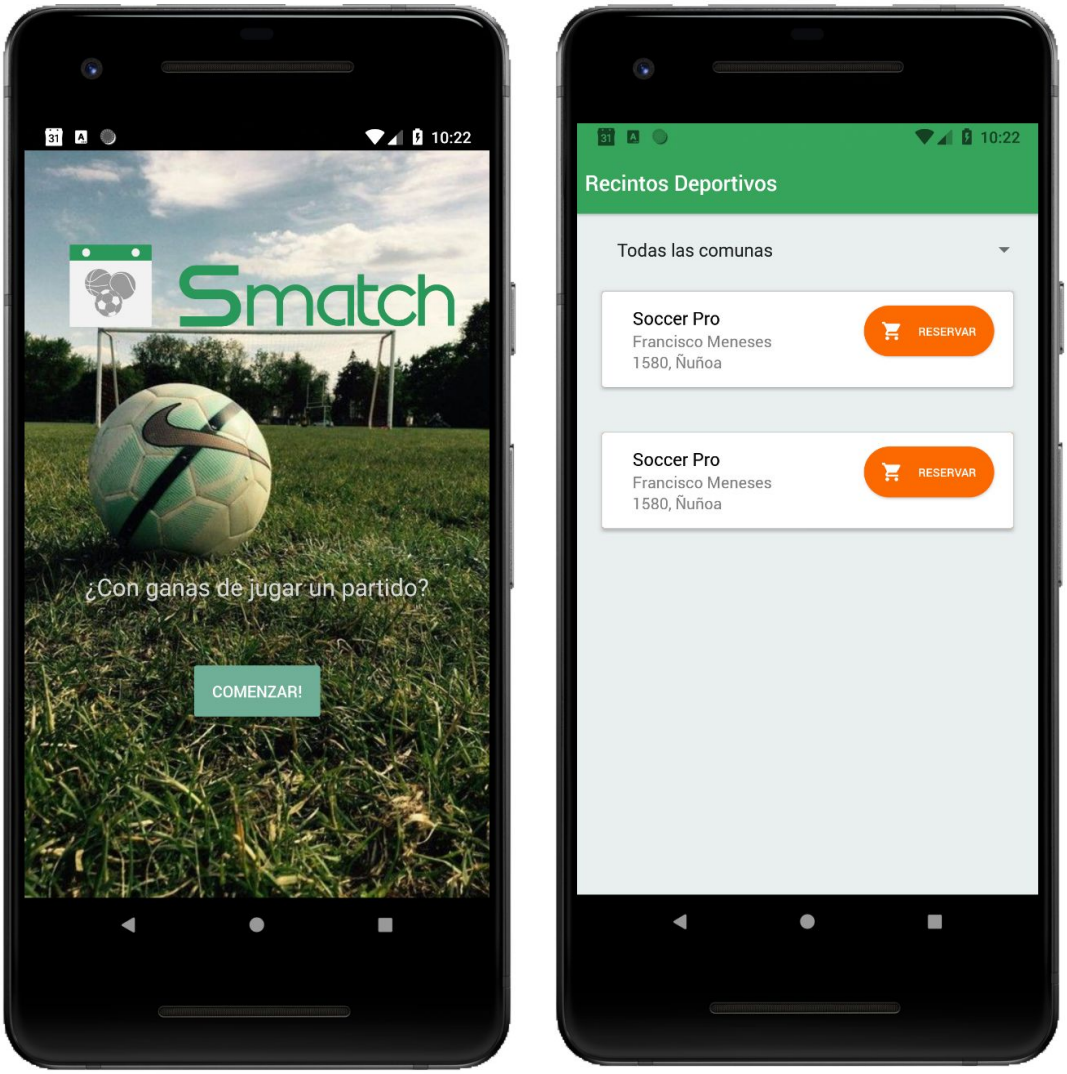


Figura 4.5: Vista inicial y listado de recintos deportivos de la aplicación. Fuente "elaboración propia".

Para el desarrollo de la etapa 0 se consideran al menos tres vistas principales, basadas en el capítulo 2.4 :

1. **Vista de bienvenido**: una vista simple pero visualmente potente que le dé al usuario la sensación familiar a un evento deportivo.

2. **Vista de lista de recintos deportivos:** esta vista contiene tarjetas de recintos deportivos un resumen de su información relevante, el objetivo es que el usuario pueda buscar distintos recintos deportivos/sucursales; filtrados por provincias.
3. **Vista de detalle de los recintos deportivos**: esta vista contiene el detalle de un recinto deportivo en particular; el objetivo es que se pueda otorgar al usuario información más complementaria para la toma de decisión en su reserva. Cabe destacar que debe mostrar los horarios y precios en detalle del recinto.

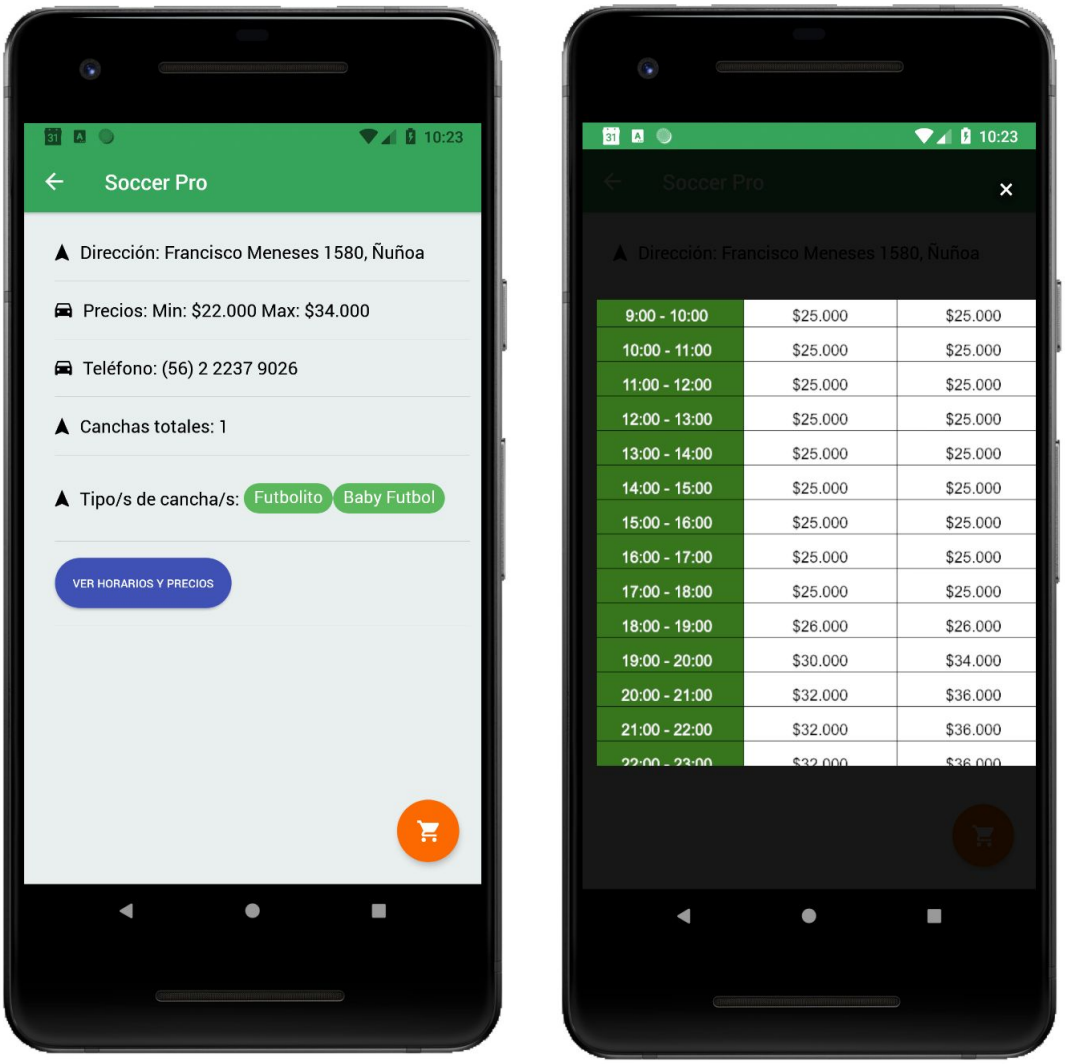


Figura 4.6: Vista del detalle de un recinto deportivo y la acción del botón “ver horarios y precios”. Fuente “elaboración propia”.

La Figura 4.5 muestra las siguientes vistas:

- **Presentación**: el objetivo es poner en contexto al usuario de la aplicación y que se sienta ameno en ella. Además es el hincapié para la etapa 2 cuando se requiera un registro de usuario.
- **Lista de Recintos deportivos**: se muestra una lista de recintos ordenados por comuna. Se agrega el botón "Reservar", el cual al presionarlo lleva directamente al usuario a la aplicación de teléfono para llamar al recinto.

La Figura 4.6 muestra las siguientes vistas:

- **Detalle Recintos deportivo**: se muestra toda la información relevante que necesita un jugador para reservar una cancha. En la esquina inferior derecha se muestra nuevamente el ícono del botón de reservar, entendiendo que el usuario logra detectar en la Figura 4.5 que son la misma acción (por el color y el ícono).
- **Botón Horarios y Canchas**: este botón hace la acción de desplegar una imagen con los horarios y precios de los recintos deportivos, información relevante para tomar la decisión de reservar en dicha cancha.

4.2. VALIDACIÓN DE LA SOLUCIÓN

4.2.1. Contexto

En la Sección 4.1 se desarrolla la plataforma de la etapa 0, en esta sección se hace un test de usabilidad para validar esta etapa. Para ello se hacen pruebas de usuario de tareas concretas. Dichas tareas son:

1. Saber el número de canchas que tiene el recinto Soccer Pro.

2. Saber el precio de las canchas para el día lunes a las 17:00 de Soccer Arena.

3. Buscar los recintos deportivos que estén ubicados solo en Ñuñoa

4. Reservar una cancha para el día sábado (cualquiera) que esté ubicada en Ñuñoa y tenga la reserva más económica.

Para la realización de la tarea se definen:

- **Nivel de usuario**:

  - A : un usuario muy avanzado que conoce a la perfección las interfaces de usuario móvil.
  - B : un usuario promedio, que usa frecuentemente aplicaciones móviles pero sin conocimientos críticos.
  - C : un usuario más reacio a la tecnología, de uso muy poco frecuente de aplicaciones.
- **Tareas**: son las acciones que debe hacer el usuario que se pone a prueba.
- **Tiempo**: el tiempo en segundos que toma el usuario en realizar la tarea.

Se encuesta una total de 10 usuarios, 5 de los cuales se consideran del nivel A, 3 del nivel B y 2 del nivel C. El protocolo a seguir es explicar el contexto de la aplicación y los objetivos que se quieren cumplir. Luego los usuarios se envuelven en una situación en particular para luego realizar de la tarea de forma aislada (sin consultar). Finalmente, al terminar cada tarea se evalúa si realizó la tarea o no, si es que se entiende la interacción de la aplicación para realizar determinada tarea y algunos comentarios.

Se espera tener al menos un **90%** de las tareas logradas para considerar una validación exitosa de la etapa 0.

4.2.2. Resultados

Tareas	Tiempo promedio de realización de una tarea (segundos)	Porcentaje tareas logradas	Porcentaje de entendimiento de interfaz
Saber el número de canchas que tiene el recinto Soccer Pro.	20,5	100%	100%
Saber el precio de las canchas para el día lunes a las 17:00 de Soccer Arena.	21,3	100%	90%
Buscar los recintos deportivos que estén ubicados solo en Ñuñoa	19,6	100%	100%
Reservar una cancha para el día sábado (cualquiera) que esté ubicada en Ñuñoa y tenga la reserva más económica.	54,7	100%	100%

Tabla 4.1: Resultados del test de usuario. Fuente: “elaboración propia”.

La Tabla 4.1 muestra los resultados del test de usuario (ver Anexo 2) realizado a 10 usuarios, del cual el 100% de los datos lograron realizar la tarea satisfactoriamente, lo que permite concluir que se tiene una buena interfaz y que cumple con la necesidad

requerida por etapa 0. Sin embargo, hay ciertas mejoras que se deben realizar, gracias a los comentarios de los usuarios, tales como:

- **Mejorar la imagen de horarios y precios**: muchos usuarios se vieron algo complicados al ver dicha imagen, por lo que se recomienda mejorar la interfaz. Sin embargo dicha funcionalidad cumple el objetivo planteado.
- **Mejorar el filtro por comunas**: debido a la cantidad de comunas que existen, los usuarios recomiendan tener un autocompletar dentro de la búsqueda por comunas para su facilidad de uso.
- **Encontrar una forma más fácil de comparación de recintos**: si bien la tarea fue realizada con éxito en el 100% de los casos, los usuarios encuentran que la interfaz podría entregar mejores herramientas para comparar precios en recintos deportivos. Se recomienda aumentar los filtros en el listado de recintos deportivos que permitan la comparación desde dicha vista.

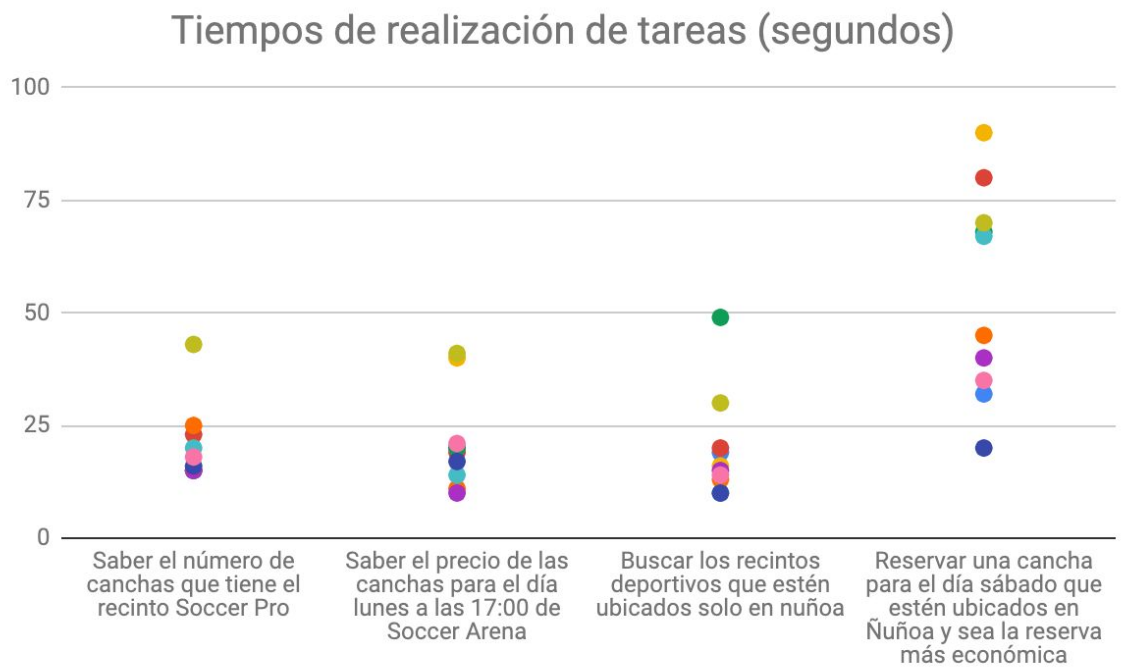


Figura 4.7: Tiempos de realización de tareas. Fuente: “elaboración propia”.

La Figura 4.7 muestra la dispersión de los tiempos que hubo en cada tarea. En general no existe mayor dispersión en las primeras tres tareas, salvo los usuarios tipo C que muestran una diferencia notable con respecto a los otros tipos de usuario, sin embargo los tiempos mostrados para cada uno, no presentaron incomodidad en los usuarios, todos lograron

sentir una buena experiencia con respecto al tiempo de realización de cada tarea. De todas maneras se debe mejorar la experiencia para que los usuarios tipo C se logren acercar aún más a los de tipo B y A. Con respecto a la última pregunta de la encuesta se logra observar mayor dispersión en los datos, lo que evidencia que no hay un patrón exacto en la realización de la tarea, es decir, no es tan evidente para el usuario lo que se debe hacer para la realización de la última tarea. Para esto último, se recomienda refactorizar esta funcionalidad para hacer más claro lo que el usuario tiene que hacer.

CONCLUSIONES

CONCLUSIONES GENERALES

En este trabajo se ha logrado dar puntapié inicial para el desarrollo de un proyecto de emprendimiento, capaz de cumplir con el dinamismo que requiere una *startup* y tener una base tecnológica sólida adaptarse adecuadamente a todos los requerimientos del mercado. Durante el desarrollo se pudo apreciar la transformación del proyecto basado en las metodologías de la Feria de *Software* de la UTFSM, para convertirse en una *startup* con alto potencial de mercado.

Se ha comprobado con estre trabajo la gran utilidad de algunas herramientas como la metodología CANVAS para el replanteamiento de todas las áreas del negocio, como también, la metodología ágil hizo que la forma de construir un software sea más adaptado a las necesidades de una *startup*. Con esto se logró evidenciar las diferencias entre el anterior y el nuevo proyecto, esto permitió un plan de desarrollo y arquitectura para poder resolver dichas diferencias.

Un elemento importante para la realización de este trabajo es la Reingeniería de *Software*, ya que, se logra evidenciar que las principales diferencias están en los requerimientos no funcionales, lo que llevó a replantearse la arquitectura en que se estaba construyendo el software. Es muy probable de que si la principal diferencia hubiese estado en los requerimientos funcionales, habría que haber aplicado medidas distintas a las que se usaron en este trabajo, como por ejemplo, hacer una restructuración del código, cambio de lenguaje de programación, entre otros.

Otro elemento a destacar es la importancia del replanteamiento del modelo de negocios, ya que con ello se logró evidenciar a través de la Reingeniería de Software los requerimientos no funcionales que marcaron la diferencia entre lo que estaba desarrollado y lo que requiere el mercado.

Cabe mencionar que esta memoria será utilizada para el desarrollo de un emprendimiento real, siendo la etapa 0 el gran avance que se tendrá para continuar el negocio. Se espera avanzar con las siguientes etapas y con las metodologías planteadas.

Finalmente vale decir que este documento no presenta una crítica hacia la asignatura, sino más bien, es un manual para levantar proyectos que han pasado por problemas similares a los planteados.

CUMPLIMIENTO DE OBJETIVOS

En la Sección 1.4.2 se definieron los objetivos específicos que se buscaban cumplir con este trabajo. A continuación se hace un análisis sobre el cumplimiento de cada uno de ellos.

Replantear el modelo de negocios para crear una nueva propuesta adaptada a las necesidades del mercado actual:

En la Sección 3.1 se presentaron los resultados de la metodología CANVAS. El proceso fue exitoso, se logra identificar cada una de las partes del negocio, sobretodo el replanteamiento de la propuesta de valor, lo cual, repercuta principalmente en la idea completa del proyecto. Gracias a lo anterior se construye un flujo de caja a 5 años, el cual logra generar un proyecto de valor al mercado. Sin embargo al igual que en la metodología ágil en el desarrollo de *software*, los modelos de negocios también deben ir iterando, por lo tanto esto es un trabajo de replanteamiento continuo.

Aplicar reingeniería al software para verificar, filtrar y/o crear casos de uso acordes a la propuesta de valor, para tener un software acorde a la nueva propuesta de valor:

Utilizando lo planteado en la Sección 3.2, se hace un replanteamiento de los requerimientos funcionales y no funcionales. Por lo tanto, en la Sección 3.3 se modelan los casos de uso anteriores obtenidos a través del proceso de la reingeniería. Gracias a esto y al nuevo modelo de negocios es que se logra levantar los nuevos requerimientos funcionales y no funcionales. En la misma Sección 3.2 se analizan las diferencias que existen entre requerimientos, para luego en la Sección 3.3 plantear una arquitectura de software en base a ellos.

Diseñar y construir una nueva arquitectura de software, permitiendo que el proyecto tenga herramientas que para poder competir en el mercado:

En las secciones 3.2 y 3.3 que logra aplicar Reingeniería de modelo de negocios y de *software*. En la Sección 3.2.9 se evidencian las principales diferencias que existen en el modelo de negocios y del software, debido a esto se llega a la decisión de reconstruir la

arquitectura de software. Se analizaron principalmente los requerimientos no funcionales del proyecto nuevo y las diferencias con el proyecto anterior, con esto se obtuvieron tres modelos de arquitecturas posibles de implementar .

Finalmente se toma la decisión de que la arquitectura más acorde a los requerimientos no funcionales, y por consiguiente al mercado, es la MVVM. Con esto se logra crear las historias de usuario y un plan de desarrollo por 4 etapas.

Construir y validar una etapa de la solución, para verificar si la propuesta entrega un mínimo valor:

El Capítulo 4 se centró en este objetivo. Se hace un desarrollo de todo lo planteado en capítulos anteriores. Se plantea y desarrolla un mínimo producto viable a través de la etapa 0. Dicha etapa consiste en el montaje inicial de toda la arquitectura planteada y en el requerimiento funcional de tener un *booking* de recintos deportivos con datos relevantes para un organizador de partidos. Se desarrollan 3 vistas, la primera de "bienvenido", la segunda el "listado de recintos deportivos" y por último el "detalle de un recinto deportivo", con las heurísticas planteadas en el Capítulo 2. Además se desarrolló el modelo de datos inicial para cumplir con las peticiones de las vistas.

En la Sección 4.2 se desarrolla el test de usuario del *software* construido. Se utiliza el test de usabilidad y se definen 4 tareas. Se mide el cumplimiento y el tiempo que demora el usuario al realizar la tarea. Se logra evidenciar el cumplimiento de todos los objetivos y la facilidad de uso para los usuarios. Sin embargo, hay ciertas partes de la interfaz que se deben mejorar para mantener un cierto grado de consistencia en cuanto a las acciones del usuario.

EXTENSIONES FUTURAS

Si bien, se logra el objetivo de levantar un nuevo proyecto y testear el valor de la etapa 0, no se pudo validar el real valor de Smatch, el cual se basa en resolver la problemática de la organización de partidos (esto estaba en la etapa 2). Por lo tanto, queda como trabajo a futuro continuar con la validación del proyecto hasta lograr resolver su propuesta de valor.

Una extensión para este proyecto es testear la usabilidad completa que se propuso en el Capítulo 4.2; esto permitiría un mejor desempeño en el software y por consiguiente una

mejor solución a la propuesta de valor. Sin embargo, evaluar el diseño de interfaz de la aplicación se escapa del alcance de esta memoria.

En este documento, se revisó un caso de estudio en particular del proyecto Smatch, cuya reingeniería dió como resultado que las principales diferencias se encontraran en los requerimientos no funcionales; por lo tanto se toma la decisión de reconstruir la arquitectura del software. Es importante seguir validando este estilo de reingeniería con más proyectos, así se generan más casos que analizar y ganar experiencia .

APORTE DE LA CARRERA

Las asignaturas más relevantes para el desarrollo de este documento fueron:

- **Base de datos**: gracias a esta asignatura se obtienen los conocimientos esenciales para la construcción de software. Este ramo en particular considera las bases de datos relacionales y las tecnologías de desarrollo de software como *Ruby on Rails.*
- **Ingeniería de software**: la base de este proyecto es la reingeniería, por lo tanto esta asignatura fue de suma importancia para la construcción de este documento. Se toma en consideración el levantamiento de requerimientos y casos de uso propuestos en esta memoria.
- **Evaluación de proyectos**: el modelo de negocios y flujos de caja se basaron en esta asignatura, en la cual se estudia cómo llevar a cabo un software al mercado.
- **Diseño de interfaces de usuario:** la facilidad de uso, heurísticas de Nielsen, test de usuario son contenidos de esta asignatura. Gracias a esto se logra construir exitosamente los secciones 4.1 y 4.2.

REFERENCIAS BIBLIOGRÁFICAS

[1] “TIOBE Programming Community Index.” URL: https://www.tiobe.com/tiobe-index/ (visitado 4-12-2018).

[2] “La evolución de los Lenguajes de Programación - ParcelaDigital”. URL: https://parceladigital.com/2016/04/20/la-evolucion-de-los-lenguajes-de-programacion/ (visitado 5-12-2018).

[3] “¿Qué es Framework? - Definición en WhatIs.com - SearchDataCenter”. URL: https://searchdatacenter.techtarget.com/es/definicion/Framework (vistiado 5-12-2918).

[4] “Software Re-engineering - Semantic Scholar”. URL: https://pdfs.semanticscholar.org/d6b2/89019f9fef924fd3300d1094f29c8bc079f9.pdf (visitado 5-12-2018).

[5]) “Object-Oriented Reengineering Patterns - SCG”. URL: http://scg.unibe.ch/download/oorp/OORP.pdf (visitado 5-12-2018).

[6] “¿Por qué utilizar Ruby?.” URL: https://openwebinars.net/blog/por-que-utilizar-ruby/ (visitado 23-12-2018).

[7] “¿Por qué tantas startups usan ruby on rails?. URL: http://blog.desafiolatam.com/por-que-tantas-startups-usan-ruby-on-rails/ (visitado 23-12-2018).

[8] “Javascript | MDN”. URL: https://developer.mozilla.org/es/docs/Web/JavaScript (visitado 23-12-2018).

[9] “Everything you need to know about Ruby on Rails ”. URL: https://skillcrush.com/2015/01/29/13-ruby-rails/ (visitado 23-12-2018)

[10] L Richardson, M Amundsen y S Ruby. RESTful Web APIs. O’Reilly Media, 2013. ISBN: 1449358063.

[11] “Top 5 Metodologías de Desarrollo de software tradicional”. URL: https://www.megapractical.com/blog-de-arquitectura-soa-y-desarrollo-de-software/metodologias-de-desarrollo-de-software (visitado 23-12-2018).

[12] Eric Ries. “Learn Startup”, 2013, EEUU. ISBN: 0307887898.

[13] “Definición de ingeniería de software”. URL: (https://definicion.de/ingenieria-de-software (visitado 25-12-2018).

[14] “Qué son las metodologías ágiles y cuales son sus ventajas empresariales”. URL: https://www.iebschool.com/blog/que-son-metodologias-agiles-agile-scrum/ (visitado 27-12-2018).

[15] “La lógica de la usabilidad”. URL: https://www.beeva.com/beeva-view/diseno-y-ux/la-logica-de-la-usabilidad/ (vistiado 27-12-2018).

[16] Kara Pernice and Jacob Nielsen. “Usability Guidelines for Accessible Web design”.isbn: 0201774224.

ANEXOS

1. FLUJO DE CAJA DEL MODELO DE NEGOCIOS

	AÑO	0	1	2	3	4	5
Ingreso							
	Suscripción	$0	$5.625.000	$23.250.000	$41.200.000	$54.700.000	$61.050.000
	Fee	$0	$8.437.500	$34.875.000	$61.800.000	$82.050.000	$91.575.000
	Promoción	$0	$0	$0	$1.706.000	$4.053.930	$7.280.276
	Ingresos	$0	$14.062.500	$58.125.000	$104.706.000	$140.803.930	$159.905.276
			$0	$0	$0	$0	$0
			$0	$0	$0	$0	$0
			$0	$0	$0	$0	$0
Costos			$0	$0	$0	$0	$0
	Servidores Empresas		$ -360.000	$ -461.250	$ -609.376	$ -703.128	$ -849.618
	Servidores Persona		$ -360.000	$ -360.000	$ -517.032	$ -856.941	$ -1.413.370
	Fee por transacciones Online	$0	$ -2.531.250	$ -10.462.500	$ -18.540.000	$ -24.615.000	$ -27.472.500
	Costos de Oficina		$ -750.000	$ -9.000.000	$ -9.150.000	$ -10.800.000	$ -10.800.000
Gastos			$0	$0	$0	$0	$0
	Sueldo Fundadores (DEV)		$ -500.000	$ -6.500.000	$ -12.500.000	$ -18.000.000	$ -18.000.000
	Sueldo Fundadores (Growth y Ventas)		$ -500.000	$ -6.500.000	$ -12.500.000	$ -18.000.000	$ -18.000.000
	Practicante Dev		$ -1.800.000	$ -1.800.000	$ -1.800.000	$ -1.800.000	$ -1.800.000
	Practicante Diseño		$ -1.200.000	$ -1.200.000	$ -1.100.000	$0	$0
	Dev Full stack		$ -4.000.00	$ -12.000.00	$ -12.000.00	$ -12.000.00	$ -12.000.00

			0	0	0	0	0
	Dev Front		$0	$0	$0	$0	$0
	Diseñador		$0	$0	$ -500.000	$ -6.000.000	$ -6.000.000
	Servicio al cliente		$0	$0	$ -350.000	$ -4.200.000	$ -4.200.000
	Publicidad		$ -650.000	$ -1.200.000	$ -1.200.000	$ -2.450.000	$ -3.000.000
	Inicio de actividad	$ -200.000	$0	$0	$0	$0	$0
Inversión							
	Invierno inicial	$ -3.000.000	$0	$0	$0	$0	$0
			$0	$0	$0	$0	$0
Flujo		$ -3.200.000	$1.411.250	$8.641.250	$33.939.593	$41.378.861	$56.369.788

2. TEST DE USUARIO

Nombre:	Felipe Flores
Nivel de usuario:	A

1

Tareas	Tiempo (Segundos)	Logra el objetivo	Se entendió la interacción
Saber el número de canchas que tiene el recinto Soccer Pro	15	si	si
Saber el precio de las canchas para el día lunes a las 17:00 de Soccer Arena	20	si	si
Buscar los recintos deportivos que estén ubicados solo en Ñuñoa	19	si	si
Reservar una cancha para el día sábado (cualquiera) que esté ubicada en Ñuñoa y tenga la reserva más económica	32	si	si

Nombre:	Carlos Bugueño
Nivel de usuario	A

2

Tareas	Tiempo (Segundos)	Logra el objetivo	Se entendió la interacción
Saber el número de canchas que tiene el recinto Soccer Pro	23	si	si
Saber el precio de las canchas para el día lunes a las 17:00 de Soccer Arena	19	si	si
Buscar los recintos deportivos que estén ubicados solo en Ñuñoa	20	si	si
Reservar una cancha para el día sábado (cualquiera) que esté ubicada en Ñuñoa y tenga la reserva más económica	80	si	si

Nombre:	Nicol Vicencio
Nivel de usuario	C

3

Tareas	Tiempo (Segundos)	Logra el objetivo	Se entendió la interacción
Saber el número de canchas que tiene el recinto Soccer Pro	15	si	si
Saber el precio de las canchas para el día lunes a las 17:00 de Soccer Arena	40	si	si
Buscar los recintos deportivos que estén ubicados solo en Ñuñoa	16	si	si
Reservar una cancha para el día sábado (cualquiera) que esté ubicada en Ñuñoa y tenga la reserva más económica	90	si	si

Nombre:	Javiera Quezada	4
Nivel de usuario	B	

Tareas	Tiempo (Segundos)	Logra el objetivo	Se entendió la interacción
Saber el número de canchas que tiene el recinto Soccer Pro	15	si	si
Saber el precio de las canchas para el día lunes a las 17:00 de Soccer Arena	20	si	no
Buscar los recintos deportivos que estén ubicados solo en Ñuñoa	49	si	si
Reservar una cancha para el día sábado (cualquiera) que esté ubicada en Ñuñoa y tenga la reserva más económica	68	si	si

Nombre:	Benjamin Saavedra	5
Nivel de usuario	B	

Tareas	Tiempo (Segundos)	Logra el objetivo	Se entendió la interacción
Saber el número de canchas que tiene el recinto Soccer Pro	25	si	si
Saber el precio de las canchas para el día lunes a las 17:00 de Soccer Arena	11	si	si
Buscar los recintos deportivos que estén ubicados solo en Ñuñoa	13	si	si
Reservar una cancha para el día sábado (cualquiera) que esté ubicada en Ñuñoa y tenga la reserva más económica	45	si	si

Nombre:	Alejandro Lopez	6
Nivel de usuario	B	

Tareas	Tiempo (Segundos)	Logra el objetivo	Se entendió la interacción
Saber el número de canchas que tiene el recinto Soccer Pro	20	si	si
Saber el precio de las canchas para el día lunes a las 17:00 de Soccer Arena	14	si	si
Buscar los recintos deportivos que estén ubicados solo en Ñuñoa	10	si	si
Reservar una cancha para el día sábado (cualquiera) que esté ubicada en Ñuñoa y tenga la reserva más económica	67	si	si

Nombre:	Javier Figueroa	7
Nivel de usuario	A	

Tareas	Tiempo (Segundos)	Logra el objetivo	Se entendió la interacción
Saber el número de canchas que tiene el recinto Soccer Pro	15	si	si
Saber el precio de las canchas para el día lunes a las 17:00 de Soccer Arena	10	si	si
Buscar los recintos deportivos que estén ubicados solo en Ñuñoa	15	si	si
Reservar una cancha para el día sábado (cualquiera) que esté ubicada en Ñuñoa y tenga la reserva más económica	40	si	si

Nombre:	Rosa Pereira	8
Nivel de usuario	C	

Tareas	Tiempo (Segundos)	Logra el objetivo	Se entendió la interacción
Saber el número de canchas que tiene el recinto Soccer Pro	43	si	si
Saber el precio de las canchas para el día lunes a las 17:00 de Soccer Arena	41	si	si
Buscar los recintos deportivos que estén ubicados solo en Ñuñoa	30	si	si
Reservar una cancha para el día sábado (cualquiera) que esté ubicada en Ñuñoa y tenga la reserva más económica	70	si	si

Nombre:	Ian Quiroga
Nivel de usuario	A

Tareas	Tiempo (Segundos)	Logra el objetivo	Se entendió la interacción
Saber el número de canchas que tiene el recinto Soccer Pro	16	si	si
Saber el precio de las canchas para el día lunes a las 17:00 de Soccer Arena	17	si	si
Buscar los recintos deportivos que estén ubicados solo en Ñuñoa	10	si	si
Reservar una cancha para el día sábado (cualquiera) que esté ubicada en Ñuñoa y tenga la reserva más económica. económica	20	si	si

Nombre:	Fabián Vlani	10
Nivel de usuario	A	

Tareas	Tiempo (Segundos)	Logra el objetivo	Se entendió la interacción
Saber el número de canchas que tiene el recinto Soccer Pro	18	si	si
Saber el precio de las canchas para el día lunes a las 17:00 de Soccer Arena	21	si	si
Buscar los recintos deportivos que estén ubicados solo en Ñuñoa	14	si	si
Reservar una cancha para el día sábado (cualquiera) que esté ubicada en Ñuñoa y tenga la reserva más económica	35	si	si