



**ESPE**  
UNIVERSIDAD DE LAS FUERZAS ARMADAS  
INNOVACIÓN PARA LA EXCELENCIA

**DEPARTAMENTO DE CIENCIAS DE LA  
COMPUTACIÓN**

**CARRERA DE INGENIERÍA EN SISTEMAS E INFORMÁTICA**

**TRABAJO DE TITULACION PREVIO A LA OBTENCIÓN DEL  
TÍTULO DE  
INGENIERO EN SISTEMAS E INFORMÁTICA**

**TEMA:**

**DESARROLLO DE UN SISTEMA DE TRUEQUE VIRTUAL  
PARA DISPOSITIVOS MÓVILES ANDROID, UTILIZANDO  
SOFTWARE LIBRE BASADO EN LENGUAJES DE  
PROGRAMACIÓN INTERPRETADOS**

**AUTORES:**

**NARANJO TORRES CARLOS ISAAC  
VILLARRUEL DUQUE PABLO PATRICIO**

**DIRECTORA:**

**ING. PRISCILA RODRÍGUEZ**

**SANGOLQUÍ  
2016**

## CERTIFICADO

Quito, 6 de Enero de 2016


Señor Ingeniero  
Mauricio Campaña O. MDU. Ms.  
Director de la Carrera de Ingeniería de Sistemas  
UNIVERSIDAD DE LAS FUERZAS ARMADAS ESPE  
Presente.

Señor Director:

En atención al memorando en el que se me designó Directora de la Tesis titulada **“DESARROLLO DE UN SISTEMA DE TRUEQUE VIRTUAL PARA DISPOSITIVOS MÓVILES ANDROID, UTILIZANDO SOFTWARE LIBRE BASADO EN LENGUAJES DE PROGRAMACIÓN INTERPRETADOS”**, desarrollada por los señores egresados Pablo Patricio Villarruel Duque y Carlos Isaac Naranjo Torres, tengo a bien poner en su conocimiento que se realizó la revisión de la misma tanto en su parte teórica como práctica por lo cual solicito a usted permitir que los señores puedan continuar con su trámite para su titulación.

Por la atención que se sirva dispensar a la presente, le expreso mis agradecimientos.

Atentamente,

  
Ing. Priscila Rodríguez  
Directora Tesis

## AUTORIA

Nosotros Pablo Patricio Villarruel Duque con cédula de identidad N° 1722382940 y Carlos Isaac Naranjo Torres con cédula de identidad N° 1718659194, declaramos que este trabajo de titulación "DESARROLLO DE SISTEMA DE TRUEQUE VIRTUAL PARA DISPOSITIVOS MÓVILES ANDROID, UTILIZANDO SOFTWARE LIBRE BASADO EN LENGUAJES DE PROGRAMACIÓN INTERPRETADOS" ha sido desarrollado considerando los métodos de investigación existentes, así como también se ha respetado los derechos intelectuales de terceros considerándose en las citas bibliográficas.

Consecuentemente declaramos que este trabajo es de nuestra autoría, en virtud de ello nos declaramos responsable del contenido, veracidad y alcance de la investigación mencionada.

Sangolquí, 6 de enero de 2016



---

Pablo Villarruel



---

Carlos Naranjo

## AUTORIZACION

Nosotros Pablo Patricio Villarruel Duque y Carlos Isaac Naranjo Torres, autorizamos a la Universidad de las Fuerzas Armadas ESPE publicar en la biblioteca Virtual de la institución en presente trabajo de titulación "DESARROLLO DE SISTEMA DE TRUEQUE VIRTUAL PARA DISPOSITIVOS MÓVILES ANDROID, UTILIZANDO SOFTWARE LIBRE BASADO EN LENGUAJES DE PROGRAMACIÓN INTERPRETADOS" cuyo contenido, ideas y criterio son de nuestra autoría y responsabilidad.

Sangolquí, 6 de enero de 2016



---

Pablo Villarruel



---

Carlos Naranjo

## **DEDICATORIA**

Dedico el presente trabajo a todas las personas que no se conforman con aprender un solo lenguaje de programación, gracias a ellas no habría motivación para investigar y dedicar esfuerzos para nuevas tecnologías.

Pablo Patricio Villarruel Duque

Dedico este trabajo a mis padres que han sido pilar fundamental en mi vida y me han apoyado a lo largo de mi vida. También quiero dedicarlo a todas aquellas personas que invierten sus esfuerzos en generar conocimiento y lo difunden porque por ellos es que temas como este, se pueden llegar a conocer.

Carlos Isaac Naranjo Torres.

## AGRADECIMIENTO

Agradezco a Dios por siempre cuidarme y estar a mi lado guiándome cada día de mi vida. A mis padres por siempre motivarme y ayudar a seguir mis metas; a mi abuelita por formar mi niñez para ser una persona responsable.

A David y Gustavo por mostrarme la innovación en tecnología, y ayudarme a crecer profesionalmente.

A Andrea por nunca soltar mi mano, y siempre estar presente en las buenas y en las malas. Gracias por ayudarme a alcanzar el peldaño en el que me encuentro.

Gracias a todas las personas que formaron mi vida de estudio en la universidad, de alguna forma se convirtieron en un ejemplo a seguir o evitar.

Pablo Patricio Villarruel Duque

Primero quiero agradecer a Dios por permitirme culminar esta etapa de mi vida que ha tenido sus altos y bajos pero al final siempre me ha guiado por el camino correcto.

Quiero agradecer también a mis padres a mis hermanos y familiares que me han brindado su apoyo incondicional en todo este tiempo.

Finalmente quiero agradecer a todas las personas que estuvieron presentes en esta etapa de mi vida y se preocuparon por mí, así mismo a todos los maestros y amigos con los que compartí las aulas que han sido de cierta forma una motivación para mejorar.

Carlos Isaac Naranjo Torres.

## INDICE

<b>CERTIFICADO .....</b>	<b>i</b>
<b>AUTORIA.....</b>	<b>ii</b>
<b>AUTORIZACION.....</b>	<b>iii</b>
<b>DEDICATORIA.....</b>	<b>iv</b>
<b>AGRADECIMIENTO .....</b>	<b>v</b>
<b>LISTADO DE FIGURAS .....</b>	<b>ix</b>
<b>LISTADO DE TABLAS .....</b>	<b>xi</b>
<b>RESUMEN.....</b>	<b>xii</b>
<b>ABSTRACT.....</b>	<b>xiii</b>
<b>1. CAPITULO I.....</b>	<b>1</b>
1.2. ANTECEDENTE .....	1
1.3. PLANTEAMIENTO DEL PROBLEMA.....	2
1.4. JUSTIFICACIÓN.....	3
1.5. OBJETIVOS.....	4
1.6. ALCANCE .....	5
1.7. HERRAMIENTAS.....	5
1.8. FACTIBILIDAD .....	12
1.8.1. Factibilidad Técnica .....	12
1.8.2. Factibilidad Económica.....	13
1.8.3. Factibilidad Operativa.....	15
<b>2. CAPÍTULO II .....</b>	<b>16</b>
2.1. TRUEQUE .....	16
2.2. LEY DE COMERCIO ELECTRÓNICO DE ECUADOR .....	18
2.3. MODELO PRODUCT CANVAS.....	19

2.4.	SISTEMA OPERATIVO ANDROID.....	21
2.5.	LENGUAJES DECLARATIVOS.....	22
2.6.	RUBY ON RAILS.....	23
2.7.	JAVASCRIPT .....	32
2.8.	SENGHA TOUCH .....	32
2.9.	CORDOVA .....	34
2.10.	DISPOSITIVOS MÓVILES.....	38
<b>3.</b>	<b>CAPITULO III.....</b>	<b>41</b>
3.1.	DESCRIPCIÓN DEL PROCESO ACTUAL.....	41
3.1.1.	Proceso de Trueque en redes sociales .....	41
3.2.	FASE DE EXPLORACIÓN CON PRODUCT CANVAS .....	42
3.2.1.	Visión.....	42
3.2.2.	Personas .....	43
3.2.3.	Historia de Usuarios.....	43
3.2.4.	Diseño .....	44
3.2.5.	Restricciones .....	45
3.2.6.	Iteraciones de usuario.....	46
3.3.	ANÁLISIS DE REQUERIMIENTOS. ....	47
3.3.1.	Descripción general.....	47
<b>4.</b>	<b>CAPITULO IV .....</b>	<b>55</b>
4.1.	GESTIÓN DEL CONOCIMIENTO .....	55
4.1.1.	Ruby .....	55
4.1.2.	JavaScript .....	64
4.1.3.	Sencha Touch .....	65
4.2.	IMPLEMENTACIÓN .....	72
4.2.1.	Cliente Rest .....	72



4.3. PRUEBAS Y VERIFICACIÓN.....	74
4.3.1. Pruebas de métodos para el cliente REST.....	74
4.3.2. Pruebas interfaces.....	77
<b>5. CAPITULO V.....</b>	<b>83</b>
5.1. RESULTADOS .....	83
5.2. CONCLUSIONES.....	86
5.3. RECOMENDACIONES .....	87
<b>BIBLIOGRAFÍA.....</b>	<b>89</b>
<b>ANEXOS .....</b>	<b>94</b>

## LISTADO DE FIGURAS

Figura 1 Diseño de tabla Canvas.....	20
Figura 2 Proceso Product Canvas .....	21
Figura 3 Arquitectura de Ruby on Rails .....	24
Figura 4 Arquitectura Córdoba .....	34
Figura 5 Estructura Fuera de línea Córdoba .....	36
Figura 6 Estructura en línea Córdoba.....	37
Figura 7 Proceso actual de truque .....	41
Figura 8 Producto Canvas .....	42
Figura 9 Visión Product Canvas .....	42
Figura 10 Personas Product Canvas.....	43
Figura 11 Historias de usuarios Product Canvas .....	44
Figura 12 Diseño Product Canvas.....	45
Figura 13 Restricciones Product Canvas .....	45
Figura 14 Iteraciones de usuario Product Canvas .....	46
Figura 15 Casos de uso .....	53
Figura 16 Diagrama físico de base de datos.....	54
Figura 17 Patrón MVC Ruby on Rails.....	59
Figura 18 Ejemplos de pluralización.....	60
Figura 19 Interfaces y componentes aplicaciones ROR .....	62
Figura 20 Estructura de objeto en JSON.....	63
Figura 21 Estructura de lista en JSON .....	63
Figura 22 Pantalla de inicio Sencha Touch.....	66
Figura 23 Estructura aplicación Sencha Touch.....	67
Figura 24 Código archivo app.js .....	67
Figura 25 Código archivo application.js .....	68
Figura 26 Estructura de una Vista Sencha Touch .....	69
Figura 27 Estructura componentes Sencha Touch .....	69
Figura 28 Estructura de controlador Sencha Touch.....	70
Figura 29 Estructura modelos en Sencha Touch.....	71
Figura 30 Estructura store en Sencha Touch.....	71
Figura 31 Rutas métodos servidor web service REST.....	72

Figura 32 Prueba 1 .....	74
Figura 33 Prueba 2 .....	74
Figura 34 Prueba 3 .....	75
Figura 35 Prueba 4 .....	75
Figura 36 Prueba 5 .....	75
Figura 37 Prueba 6 .....	76
Figura 38 Receptor mail interesado en publicación.....	76
Figura 39 Receptor mail dueño de publicación. ....	77
Figura 40 Prueba 1 .....	77
Figura 41 Prueba 2 .....	78
Figura 42 Prueba 4 .....	78
Figura 43 Prueba 5 .....	78
Figura 44 Prueba 5 .....	79
Figura 45 Prueba 6 .....	79
Figura 46 Prueba 7 .....	79
Figura 47 Prueba 8 .....	80
Figura 48 Prueba 9 .....	80
Figura 49 Prueba 10 .....	80
Figura 50 Prueba 11 .....	81
Figura 51 Prueba 12 .....	81
Figura 52 Prueba 13 .....	81
Figura 53 Prueba 14 .....	82
Figura 54 Calendario Scrum .....	84
Figura 55 Flujo Product Canvas.....	84

## LISTADO DE TABLAS

<b>Tabla 1</b> Herramientas a usar para el desarrollo .....	13
Tabla 2 Software a usar.....	13
Tabla 3 Mano de obra .....	14
Tabla 4 Asesoría.....	14
Tabla 5 Materia de soporte.....	14
<b>Tabla 6</b> Total Factibilidad Económica .....	14
Tabla 7 Características de usuarios registrados para la aplicación móvil.....	47
Tabla 8 Características de usuarios no registrados para la aplicación móvil.....	48
Tabla 9 Historia de usuario 1 .....	48
Tabla 10 Historia de usuario 2 .....	49
Tabla 11 Historia de usuario 3 .....	49
Tabla 12 Historia de usuario 4 .....	50
Tabla 13 Historia de usuario 5 .....	50
Tabla 14 Historia de usuario 6 .....	51
Tabla 15 Historia usuario 7 .....	51
Tabla 16 Historia usuario 8 .....	52

## RESUMEN

El siguiente trabajo tiene como objetivo dar a conocer nuevas herramientas innovadoras para el desarrollo de software; enfocándose en el paradigma de programación con lenguajes declarativos. Estos lenguajes se acoplan a metodologías de desarrollo ágiles, por lo cual se usó una nueva herramienta para el diseño de software llamada Product Canvas. Esta herramienta ayuda a plasmar el software como un conjunto de ideas y permite tener una gran interacción con el cliente. Un estudio hecho en España sobre social commerce reveló que el 46% de los usuarios de Internet utilizó la web y redes sociales para tomar una decisión de compra; debido al auge del social commerce surge la idea de desarrollar una aplicación móvil para sistema android que permita hacer truke entre los usuarios, de objetos, bienes o servicios. Presentar un producto software en menor tiempo ha sido una de las principales motivaciones por la cual se utilizó los leguajes de programación Ruby y JavaScript para el desarrollo de la aplicación móvil. Esta aplicación al tener flujos no tan complejos de trabajo, sirvió para generar una base de conocimientos sobre estas herramientas, por lo cual el resultado del desarrollo del trabajo es, dar una visión del alcance que se puede tener al usar un paradigma declarativo con modelos o metodologías extremadamente ágiles; al usar el Modelo Product Canvas no solo se está diseñando un software, se está elaborando un documento con un lenguaje no tan técnico, de fácil entendimiento para el cliente y las personas involucradas, reduciendo tiempos de trabajo y haciendo cada iteración más corte al momento de presentar un avance en el proyecto y mejorando la comunicación con el cliente.

### **Palabras Clave:**

- RUBY ON RAILS
- SENCHA TOUCH
- JAVASCRIPT
- DECLARATIVO.
- PRODUCT CANVAS

## **ABSTRACT**

The following paper aims to present new and innovator tools for software development; focusing on the para-digm of declarative programming languages. These languages have a good coupling with agile development meth-odologies, for this reason we used a new tool for designing software called Product Canvas. This tool helps shape the software as a set of ideas and allows a great customer experience. A study in Spain about social commerce reveals that 46% of Internet users use the web and social networks to make a purchase decision; due to the rise of social commerce it arises the idea of developing a mobile application for Android that allows for barter system between users, objects, goods or services. Present a software product in less time has been one of the main rea-sons that we used languages like Ruby and JavaScript programming for the development of mobile application. This application have easy workflows, for this reason it can be useful to generate a knowledge base about these tools, so we can give an overview of the scope that may have to use a declarative paradigm with models or ex-tremely agile methodologies; when we use the Product Model Canvas is being designed not only software, is pre-paring a document with a less technical language, easy to understand for the customer and the people involved, reducing working time and doing more cutting each iteration when submit an advance in the project and improving communication with the client.

### **Keywords**

- RUBY ON RAILS.
- SENCHA TOUCH
- JAVASCRIPT
- DECLARATIVE.
- PRODUCT CANVAS

## **1. CAPITULO I**

### **1.1. TEMA**

Desarrollo de un sistema de trueque virtual para dispositivos móviles android, utilizando software libre basado en lenguajes de programación interpretados.

### **1.2. ANTECEDENTE**

En el inicio de la sociedad cuando el hombre paspo a ser sedentario y las actividades de agricultura generaban más de lo necesario, surge el hábito de intercambiar los excedentes entre las comunidades, lo que da el nacimiento al trueque. El mismo que se puede definir como la acción entre dos o más personas que acceden por trato mutuo a intercambiar algún objeto o servicio por otro de la misma o diferente índole, en el cuál puede o no interferir dinero de cualquiera de las dos partes. (González M, El trueque y la moneda, Origen, Recuperado , 1)

Aún en sociedades nómadas, las personas intercambiaban vinos, licores, alimentos. Cuando surge el concepto de familia y aparece la definición de núcleo familiar; las organizaciones familiares se unen entre ellas, formando sistemas de gobierno apropiadas para época dando paso a la moneda.

En la sociedad actual el uso masivo de dispositivos móviles y aplicaciones ha generado un gran mercado que permite la venta de productos y servicios a través de estos medios virtuales. (Báez M, 1)

Así como la tecnología avanza para brindarnos mayor comodidad, de la misma forma lo hacen las herramientas que nos permiten desarrollar aplicaciones. De tal modo que surgen nuevos paradigmas para el desarrollo de aplicaciones; como es el paradigma de lenguajes interpretados. (A. C. , 2015)

Los lenguajes interpretados son aquellos que se ejecuta definiendo su estructura y métodos, estos lenguajes permiten tener una independencia de plataforma, el código está disponible solo cuando se lo necesita sin ocupar memoria. El ámbito en el que se ejecuta es dinámico y ocupa menos espacio que los lenguajes compilados. (A. C. , 2015)

### **1.3. PLANTEAMIENTO DEL PROBLEMA**

Siempre se ha escuchado la famosa frase: La basura de un hombre, es el tesoro de otro. Desde la antigüedad ha existido esta frase muy poderosa. Si nos basamos en la historia, desde el principio de las eras de la humanidad, exactamente desde el Paleolítico; las personas se han esforzado por tener buenas relaciones para poder intercambiar objetos de valor por otros de su interés, así también se ha llegado a intercambiar servicios ya sea por bienes o por otros servicios. Todas estas actividades se han observado hasta la edad de los metales. (ClickD Sitio, 2015)

El uso intangible de los recursos naturales, junto con el cambio tan rápido y constante de la tecnología ha provocado en las personas a recuperar esta acción tan primitiva del trueque. La conciencia por no generar más desechos o tal vez, el deseo de las personas por consumir un producto nuevo pocos días después de su salida al mercado, ha generado en las personas un deseo indirecto por acumular productos que a corto plazo ya no usan; lo cual ha dado paso a un mercado donde se pueden intercambiar productos usados, principalmente productos tecnológicos. (ClickD Sitio, 2015)

Conseguir a una persona interesada en algún producto usado, puede llegar a ser una actividad muy difícil; además que puede generar pérdida de tiempo si no se llega a un acuerdo. En el peor de los casos se puede adquirir un producto defectuoso.

El principal problema observado en las aplicaciones actuales es que se obliga a los usuarios a que paguen un porcentaje por cada transacción exitosa o es necesario



pagar por un cierto tiempo el uso de la aplicación para poder conocer los datos del potencial vendedor.

Otro problema que se ha detectado es el constante desecho sobre las cosas usadas; indirectamente se desea resolver un problema ambiental, se desea fomentar el intercambio de cualquier producto así se estará reduciendo el desperdicio entre los usuarios de la aplicación.

Todos estos procesos de validaciones de usuarios, de registro de datos procesados; conllevan un mayor tiempo y dificultad de programación dependiendo la metodología y las herramientas de desarrollo. Con lenguajes interpretados el código se minimiza, así como el impacto ante algún cambio en el desarrollo de la aplicación. Los lenguajes interpretados además proponen una metodología ágil, buenas prácticas y tendencias de programación, el uso recursivo de componentes existentes minimizando el tiempo de desarrollo, da más valor al cliente ya que en un menor tiempo se puede presentar resultados y los cambios por más grandes que sean, no son tan graves para la continuidad del desarrollo como lo sería con lenguajes compilados.

#### **1.4. JUSTIFICACIÓN**

Actualmente se ha generado una tendencia en los mercados virtuales por intercambiar objetos de valor, tanto de consumo personal como para hogares, en redes sociales se observa grupos en los cuales se puede hacer trueque pero no se encuentra un procesos en los cuales se clasifique la información donde los usuarios puedan encontrar publicaciones de interés sin perder tiempo. (Mariño, Trueque social en tiempos de crisis, 2015)

Con esta aplicación, se desea fomentar una cultura, donde los usuarios puedan ser más conscientes de lo que ofrecen y de lo que están adquiriendo; dando lugar a una transacción más segura y las dos partes queden satisfechas. (ClickD Sitio, 2015)

En el país actualmente no existe una normativa legal referente al trueque, sin embargo para el desarrollo de la presente tesis tomaremos como referencia legal el

CÓDIGO CIVIL (Libro II, Título VII, Párrafo 1o y Párrafo 2o) y la LEY DE COMERCIO ELECTRÓNICO, FIRMAS ELECTRÓNICAS Y MENSAJES DE DATOS.

## **1.5. OBJETIVOS**

### **1.5.1. Objetivo General**

Desarrollar una aplicación móvil para trueque virtual en dispositivos móviles Android, utilizando software libre (Ruby on Rails, Sencha Touch) basado en lenguajes de programación interpretados.

### **1.5.2. Objetivos específico**

- Diseñar un software con el lenguaje de programación Ruby, usando tecnología web service Rest, que permita la interacción de usuarios al momento de intercambiar información para hacer trueque.
- Diseñar un cliente con el lenguaje de programación JavaScript para consumir Web service Rest, de fácil implementación que permitan divulgar, buscar, eliminar publicaciones de bienes o servicio para realizar trueque.
- Mejorar la interacción de comunicación con el cliente al momento de diseñar software utilizando el modelo Product Canvas.
- Identificar una línea de productos software con las funcionalidades necesarias, que permitan reutilizar código existente; para reducir tiempo de trabajo en el desarrollo de software.
- Documentar la información recopilada para fomentar el uso de herramientas innovadoras para el desarrollo de software.

## **1.6. ALCANCE**

Como se definió en el planteamiento del problema, el intercambio de productos o servicio es una de las actividades que el hombre ha venido desarrollando por mucho tiempo. Con el desarrollo de la tecnología es más fácil llegar a promocionar un objeto o un servicio, nuestro proyecto pretende realizar una aplicación móvil que permita publicar, calificar, comentar y determinar un valor para un producto que se desee intercambiar por otro, esto se realizará a través de la auto calificación entre los usuario.

La aplicación móvil a desarrollar será exclusiva para dispositivos móviles Android, dependiendo de la complejidad del desarrollo se determinará desde que versión de este sistema operativo es posible implementar la aplicación. El estudio del lenguaje interpretado Ruby y Java Script se realizará hasta obtener una base de conocimiento que nos permita desarrollar la aplicación que pretendemos implementar.

Además la aplicación no será cargada en play store o alguna plataforma de distribución de aplicaciones móviles, razón por la cual solo se lo entregará a un número limitado de usuarios (2), que harán el papel de testers del aplicativo, adicional a esto la aplicación no generará ningún tipo de reporte.

## **1.7. HERRAMIENTAS**

Para el desarrollo de la aplicación se utilizará Java Script y Ruby; que son lenguajes interpretados muy potentes por su extensión en el mercado y por conocimientos previos que hemos obtenido en experiencias laborales. Además se utilizará un modelo innovador para el desarrollo de la aplicación llamado Product Cambas; el cual permite tener una visión del software total y objetiva.

## Java Script



Es un lenguaje de programación que se ejecuta en el cliente, es decir que no se ejecuta sobre el servidor, obtener la información de forma dinámica es su principal objetivo, así como mejorar el rendimiento de los procesos sobre el front end. (E., 2015)

En los últimos años JavaScript se ha convertido en un lenguaje integrador, debido a que se encuentra en muchos ambientes de programación, y ya no solo para contenido web, sino también en SO y dispositivos, lo que ha permitido que se difunda e integre con otros lenguajes. Este lenguaje permite tener por separado a la información y a los procesos, por lo que la mayoría de validaciones se las realice en el cliente permitiendo que los servidores tengan una mejor respuesta a las tareas que ejecuta el cliente. (DesarrolloWeb, 2015)

## Sencha Touch



Sencha Touch y Ext JS son frameworks de JavaScript para la construcción de ricas aplicaciones móviles así como de escritorio, ya que ejecutan una aplicación como nativa en el navegador. Este framework está enfocado en un Desarrollo Rápido de Aplicaciones (RAD).

El framework usa herramientas como Visual Basic y Delphi, estos cores permiten a los desarrolladores ensamblar rápidamente una aplicación con una amplia gama de componentes de interfaz de usuario y de acceso a datos configurables. La experiencia de desarrollo es similar a las herramientas gráficas tradicionales, con una API orientada a componente utilizado en HTML y SVG (o VML) en lugar de dibujar en un contexto gráfico. Las aplicaciones se escriben completamente en JavaScript, aprovechando un sistema de clases de propiedad que emula programación orientada a objetos (POO) patrones tradicionales y facilita la creación de instancias de objetos declarativos, así como la configuración mediante objetos JSON. (David, 2014)

A pesar que se basan en patrones y convenciones similares, Sencha Touch y Ext JS son dos productos distintos. Sencha Touch está orientado para el uso con los navegadores de dispositivos móviles e incluye un conjunto de Touch, optimizado componentes de la interfaz de usuario que emulan la apariencia de sus componentes nativas de los sistemas iOS, Android, BlackBerry y Windows Phone. Por el contrario, Ext JS está dirigido para su uso en navegadores de escritorio e incluye un conjunto de componentes de interfaz de usuario de mouse y teclado optimizado que emulan la apariencia de las aplicaciones de escritorio estándar. Si bien existe cierta superposición en términos de clases, de sintaxis y de núcleo, en la actualidad hay similitudes significativas que ayudan a la reutilización directa entre estos dos marcos. (David, 2014)

Sencha Touch y Ext JS son parte de un conjunto más amplio de herramientas, incluyendo:

- Arquitectura Sencha, un diseño de la aplicación visual y herramienta de prototipado;
- Sencha Eclipse Plugin, una extensión para el popular Eclipse IDE;

- Sencha Mobile y Aplicaciones de Escritorio empaquetados, herramientas que pueden empaquetar aplicaciones Sencha Touch y Ext JS como aplicaciones nativas; Sencha CMD.

En conjunto, estas herramientas ofrecen una solución completa de desarrollo de aplicaciones web front-end en torno a los marcos Sencha Touch y Ext JS.

Sencha Touch y Ext JS se componen de componentes dentro de contenedores en una ventana gráfica, se describe de forma declarativa en JavaScript utilizando objetos JSON anidados como la configuración de JavaScript (es decir, objetos literales). Estos objetos de configuración incluyen un alias (alias 'xtype') para sus clases de componentes de interfaz de usuario correspondientes y que especifique los parámetros de configuración que se debe asignar a las propiedades por defecto o el comportamiento de esos componentes. Los componentes se suministran como hijos de contenedores, que gestionan los elementos dentro de un diseño configurable.

El sistema de diseño Sencha Touch y Ext JS ofrece componentes de diseño de estilo de aplicaciones flexibles (ej. 'Tarjeta', 'hbox', 'vbox', etc.) Que abstraer la complejidad de la CSS subyacente y los métodos basadas en JavaScript para alcanzar el objetivo en el diseño. (David, 2014)

Además de estos componentes de la interfaz de usuario, Sencha Touch y Ext JS también proporcionan un amplio conjunto de APIs para hacer frente a otros problemas de aplicación web comunes, tales como la manipulación del DOM, la definición y la prestación de plantillas HTML, detectar las capacidades del navegador, e interactuar con los datos de JavaScript a través de métodos. Ambos marcos suministran un paquete de datos compuesta de Modelo, Vista y Controlador, que facilitan el acceso a datos, la transformación, la consulta, la clasificación, filtrado y la persistencia a los datos, almacenamiento local o en servidores remotos a través de Ajax, JSONP, REST o Ext directa, serializado como JSON, SOAP o AMF. (David, 2014)

Sencha Touch y Ext JS incluyen una implementación de Modelo Vista Controlador (MVC) para fomentar la separación de intereses entre la presentación de la interfaz de usuario, el manejo de los gestos del usuario, y la ejecución de la lógica

de negocio. La implementación MVC suministrado incluye aplicación de base y clases Controller que se pueden extender para coordinar las interacciones entre los componentes de la interfaz de usuario y Modelos. (David, 2014)

## Cordova



Cordova es un servidor apache que permite construir aplicaciones instaladas de forma nativa utilizando HTML y JavaScript. La forma más fácil de pensar en Cordova es una vista del contenedor web que es 100% de ancho y 100 de altura%, con una interfaz de programación de JavaScript que le permite acceder a las funciones del sistema operativo nativo.

Al construir una interfaz de usuario usando las habilidades de desarrollo web tradicionales (HTML, CSS, y JavaScript), y utiliza el contenedor Cordova implementado en diferentes ambientes y dispositivos de aplicación. Cuando todos los componentes son envasados para la implementación, el servidor Cordova crea un archivo distribible binario que puede ser ejecutado por la mayoría de mercados de aplicaciones (iTunes, Google App Market, Amazon mercado, etc....). (Galeano, 2015)

Cordova es de código abierto 100%, y también se conoce con el nombre de Apache Cordova. Cordova se puede utilizar para crear aplicaciones que se dirigen a múltiples plataformas, incluyendo iOS de Apple, Android de Google, Windows Phone, BlackBerry, HP webOS, Symbian y Bada. (Galeano, 2015)

Hay que tener en cuenta que Cordova depende por completo de los componentes que ofrezca cada plataforma para incrustar un navegador web en una aplicación nativa. En el caso de Android es a través de un WebView.

## Ruby



Ruby es "un lenguaje de programación interpretado para la programación orientada a objetos rápida y fácil. (Matz, 2001)

Intérprete de lenguaje script:

- Capacidad de hacer que el sistema operativo opere directamente.
- Operaciones de gran alcance con expresiones regulares.
- Una respuesta inmediata durante el desarrollo.

Rápido y fácil:

- Declaraciones de variables son innecesarias
- Variables no se declaran.
- Sintaxis es simple y consistente
- La gestión de memoria es automática

Programación orientada a objetos:

- Todo es un objeto



- Clases, métodos, herencia, etc.
- Métodos singleton
- Funcionalidad por módulo
- Iteradores y cierres

También:

- Múltiples números enteros de precisión
- Conveniente procesamiento de excepciones
- La carga dinámica
- Apoyo en hebra.

## **NinjaMock**



Herramienta online que permite desarrollar el pre diseño de las pantallas para las aplicaciones móviles; esta herramienta permite tener el diagrama de diseño de la aplicación. Tiene una complejidad media pero tiene un diseño intuitivo para el fácil manejo de la herramienta. Es una herramienta pagada pero en su versión de prueba permite diseñar un solo proyecto que para el caso del desarrollo de la tesis fue la mejor herramienta que pudimos encontrar.

## **1.8. FACTIBILIDAD**

Dentro de una sociedad de consumo en la que los productos son intercambiados por dinero existe un pequeño grupo de personas que están dispuestas a cambiar los objetos que tienen por algo de su necesidad; actualmente existen en páginas web sociales como Facebook grupos de personas que pueden publicar sus productos pero no existe un gestor o repositorio que ayude a ubicar fácilmente los productos de interés, por esta razón es necesario desarrollar un aplicativo que permita este intercambio y que se ajuste a las necesidades de las personas para facilitar dicha operación de intercambio. (Mariño, Trueque social en tiempos de crisis, 2015)

Existe un sin número de herramientas con las que se puede desarrollar un sistema que cumpla con estos requisitos, pero teniendo una visión futurista y analizando la evolución de los sistemas de computación, se desea incursionar en el campo de los lenguajes interpretados para desarrollar una base de conocimiento a futuras investigaciones sobre estos lenguajes y con un ejemplo tangible basado en la investigación. (A. C. , 2015)

### **1.8.1. Factibilidad Técnica.**

El uso de lenguajes interpretados no es reciente, algunos de ellos como es el caso de Ruby están en nuestro medio desde hace algunos años y cuentan con su documentación pertinente que permite conocer sobre su funcionalidad.

Además el desarrollo del aplicativo está orientado hacia herramientas que se encuentran estables, de tal forma que podemos tener la confianza en que el aplicativo que se desarrolle contará con un gran soporte a nivel técnico.

Por estas razones el aplicativo orientado al intercambio de productos o bienes, usando lenguajes interpretados es un gran paso hacia el cambio de paradigma que estamos acostumbrados a estudiar.

**Tabla 1**  
Herramientas a usar para el desarrollo

<b>Lenguajes</b>
• <b>Java Script</b>
• <b>Ruby</b>
<b>FRAMEWORK</b>
• <b>Sencha Touch</b>
• <b>Rails</b>
<b>Emuladores</b>
• <b>Cordova</b>

### 1.8.2. Factibilidad Económica.

Los costos asociados al desarrollo de un sistema de trueque usando lenguajes interpretados serán asumidos por nosotros como estudiantes interesados en el desarrollo del tema, los costos son los siguientes.

Se está usando software libre:

**Tabla 2**  
Software a usar

<b>Nombre</b>	<b>Descripción</b>	<b>Costo</b>
<b>Sencha Touch</b>	sistema de clases	\$ 0
<b>Cordova</b>	Framework	\$ 0
	<b>Total:</b>	<b>\$ 0</b>

La mano de obra es asignada por los estudiantes:

**Tabla 3**

Mano de obra

Servicio	Descripción	Costo
Desarrolladores (2)	0/hora (960 horas)	0
<b>Total:</b>		<b>\$0</b>

Se buscará una persona que pueda solventar las dudas y guiar el desarrollo de la tesis.

**Tabla 4**

Asesoría

Servicio	Descripción	Valor
Capacitación en Ruby	Duración 5 semanas	\$ 250
Accesoría	50 horas (10 hora)	\$ 500
<b>Total:</b>		<b>\$750</b>

Material para a impresión de la tesis. Y diseño del software

**Tabla 5**

Materia de soporte

Unidades	Descripción	Valor
--	Material de oficina	\$ 100
<b>Total</b>		<b>\$ 100</b>

Resumen de gastos para el desarrollo de la tesis.

**Tabla****6**

Total Factibilidad Económica

Descripción	Valor
Software	\$ 0
Mano de obra	\$ 0
Asesoría	\$ 750
Materia Soporte	\$ 100
<b>Total.</b>	<b>\$ 850</b>

### **1.8.3. Factibilidad Operativa**

Para el desarrollo de la presente tesis nosotros los integrantes del equipo, hemos adquirido conocimiento relacionado con las herramientas que van a ser usadas debido a nuestra experiencia laboral dentro de las empresas en las que nos desempeñamos (Altura Soluciones, ANF AC).

Por otro lado también contamos con el apoyo de la persona que está auspiciando la tesis, la cual nos brindará soporte en temas específicos.

## 2. CAPÍTULO II

### MARCO TEÓRICO

#### 2.1. TRUEQUE

Si alguna vez has intercambiado uno de tus juguetes con un amigo a cambio de uno de sus juguetes, o en los juegos de canicas que se cambiaba una canita por otro, has hecho un intercambio. “El trueque es el comercio de servicios o bienes con otra persona cuando no hay dinero de por medio” (Mint, 2015). Este tipo de intercambio fue invocado por las primeras civilizaciones. Hay culturas incluso dentro de la sociedad moderna que todavía dependen de este tipo de intercambio. El trueque ha existido alrededor de mucho tiempo, y últimamente se ha visto necesario retomar esta acción por interés propio.

El trueque surge para ayudar a ocultar el verdadero valor de mercado. Por otra parte, el trueque es visto por muchos expertos como una forma para evitar el pago de impuestos al distorsionar el verdadero valor de ganancias. De esta manera un pago de bienes permite eludir el pago impuesto.

Un argumento en economía virtual afirma que el trueque ayuda a pretender que el sector produzca valor a los objetos cuando en realidad no lo es. Una última explicación del uso de cambio de mercado no monetario que es complementaria explica que el trueque ayuda a reutilizar las cosas y que no pierdan valor en el tiempo.

Cuando más personas se involucran en trueque, los costes de búsqueda de mercado aumentan y por lo tanto se hace más difícil para el intercambio de bienes por dinero o por otro bien y mantener el pensamiento de "trueque justo" se pierde. Esta mentalidad genera posibles incrementos en los costos a largo plazo del uso de trueque como un sistema de intercambio, ya que este último sistema puede persistir incluso cuando es ineficiente.

El trueque encaja en la actual economía como una institución económica para hacer frente a los problemas que surgen en la transición cuando el sistema y mercado de capitales legales están poco desarrollados. En un documento reciente Blanchard y Kremer (1997) argumentan que la gran disminución de la producción ha sido causada por “desorganización”. La economía sufre de una falta de confianza.

No tener dinero en efectivo requiere un crédito comercial del interesado del producto y buscar medios intermedios para poder negociar, que les ayude a no hacer frente a la desorganización y la falta de confianza en el sistema de trueque. La falta de dinero del comprador introduce también oportunidades para realizar un intercambio que ayuda a equilibrar el poder de negociación entre las partes y reducir las distorsiones. Sin embargo, trae sus propios problemas, en particular la incertidumbre sobre el cumplimiento de los contratos, que puede ser tratado por el comercio de trueque.

Un sistema organizado de trueque puede mitigar los riesgos contractuales cuando los mercados en los focos sociales son desorganizados y hace que la transacción de las actividades de negocio sea posible que de otro modo no tendría lugar. Si no se establece una organización en el sistema el canal del trueque puede impedir la salida del proceso disminuyendo aún más de lo que debería.

### **2.1.1. Sistema de trueque**

En un sistema de trueque las personas intercambian servicios y bienes por otros servicios y bienes de interés. Hoy en día, el trueque ha hecho una reaparición utilizando técnicas que son más sofisticadas para ayudar en la negociación; por ejemplo, la Internet. En la antigüedad, en este sistema solo participaban personas de la misma zona, sin embargo hoy el trueque es global. El valor de los elementos de trueque se puede negociar con la otra parte interesada. El trueque no necesariamente implica dinero que es una de las ventajas. Se puede comprar artículos mediante el intercambio de otro elemento que y no se quiere o no se necesita. En general, el comercio de esta manera se hace por redes sociales y con páginas en línea. Esto

puede presentar cierta comodidad el ver desde un lugar cómodo lo que las personas ofrecen; pero muchas veces puede ser tedioso y cansado ver publicación por publicación hasta encontrar lo que uno está interesado. Un sistema de trueque debe estar disponible en todo momento y permitir acceder a todas las publicaciones con facilidad.

### **2.1.2. Historia del trueque**

Se dice que el trueque fue introducido por Mesopotamia en el año 6000 antes de Cristo. Después fue acogido por los fenicios, ellos no solo intercambiaban productos; al viajar mucho por los océanos, ellos también intercambiaban bienes. En Babilonia se desarrolló un sistema de trueque mejorado, lo más común que se intercambiaba eran productos como: alimentos té, armas y especias. Lo más valioso en esos tiempos era la sal, ya que permitía conservar por más tiempo los alimentos. Por lo que era normal desde los egipcios hasta los romanos que se intercambien cosas por sal. Los europeos en la edad media viajaron por todo el mundo intercambiando intercambiando artesanías, pieles y perfumes. Incluso después de que se inventó el dinero, el trueque nunca ha dejado de faltar en la sociedad. Debido a la falta de dinero, el trueque se hizo popular en la década de 1930 durante la Gran Depresión se utilizó para obtener alimentos y otros servicios. Pero en este tiempo surgieron las casas de cambio, donde los interesados dejaban sus productos y alguien más realizaba el intercambio. (EdenorOficial, 2015)

## **2.2.LEY DE COMERCIO ELECTRÓNICO DE ECUADOR**

La ley de comercio electrónico actual tiene por objetivo regular los mensajes de datos, la firma electrónica, los servicios de certificación, la contratación electrónica y telemática, la prestación de servicios electrónicos, a través de redes de información, incluido el comercio electrónico y la protección a los usuarios de estos sistemas.



Para el desarrollo de la presente tesis, nos basaremos en Capítulo I Principios Generales, Título 1 De los mensajes de datos; y al Capítulo III De los derechos de los usuarios o consumidores de servicios electrónicos de la Ley de comercio electrónico, firmas electrónicas y mensajes de datos. Principalmente de los artículos:

**Art. 2.-** Reconocimiento jurídico de los mensajes de datos.- Todos los mensajes de datos tendrán valor jurídico como si fueran documentos físicos. Cada mensaje de dato digital se someterá al cumplimiento de lo establecido en la ley.

**Art. 6.-** Información escrita.- Toda información debe ser almacenada para su posterior consulta.

**Art. 10.-** Procedencia e identidad de un mensaje de datos.- Se entiende que la información es propiedad de la persona quien la envía, y que dicha información es mandada previa autorización del receptor para que sea usada de acuerdo al contenido de la misma.

**Art. 48.-** Consentimiento para aceptar mensajes de datos. El usuario debe ser informado sobre los requerimientos que necesita para acceder a la información. Y consentimiento para recibir información.

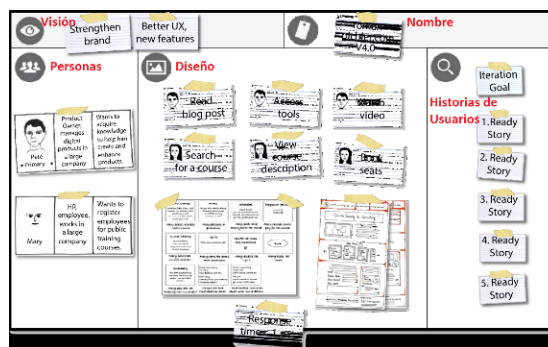
### **2.3. MODELO PRODUCT CANVAS**

Product Canvas es una herramienta de planificación, que permite capturar rápidamente las ideas para describir, diseñar, y tener un amplio espectro de visión sobre los cambios que se desea o funciones que debe hacer el producto en una sola página. (Pichler R. , 2015)

La realidad sobre documentos tradicionales de negocio, los casos de uso, planes de negocios, etc. toman demasiado tiempo para escribir, rara vez se actualizan, y casi nunca es leído por otra persona. Pero está claro que documentar, compartir y obtener información sobre su visión del producto en las primeras etapas de un producto es fundamental.

El Product Canvas resuelve todos estos problemas a través de una plantilla de una página que le permite capturar rápidamente los componentes clave de cualquier producto de una manera estructurada, concisa, que permite comunicar su visión a los interesados en un formato fácilmente comprensible. (Pichler R. , 2015)

### 2.3.1. Estructura de Product Canvas



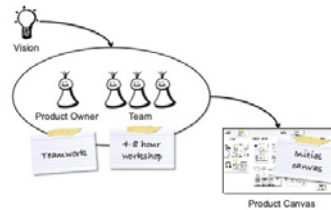
**Figura 1** Diseño de tabla Canvas

**Fuente:** (Pichler P. )

- La visión es un breve resumen de lo que el producto va a hacer.
- El nombre es el título comercial que va a tener el producto
- Las personas, son los usuarios que necesitan el producto o para quien va dirigido.
  - El diseño, son breves dibujos o flujogramas de cómo será el funcionamiento de la aplicación y que objetos gráficos tendrá.
  - Las historias de usuarios, son las tareas y acciones que debe hacer el producto. En metodologías de desarrollo conservadoras son conocidas como casos de uso. (Pichler R. , 2015)

A esta tabla también se le puede aumentar más columnas, como restricciones; la cual permite saber bajo qué circunstancias está sometida el producto o que necesita para su funcionamiento.

### 2.3.2. Cómo funciona Product Canvas



**Figura 2 Proceso Product Canvas**

**Fuente:** (Pichler P. )

Después de generar una idea concisa y tener la visión del producto, las personas interesadas se reúnen hablar sobre lo que se desea que el producto haga, debería ser parecido a una tormenta de ideas. Con un solo día de trabajo discutiendo lo que se desea para el producto debe ser más que suficiente para plasmar la idea en el Canvas. (Pichler R. , 2015)

Conforme va pasando el tiempo se puede ir llenando cada porción de la tabla del Canvas más a detalle pues el proceso debe ser iterativo, con el objetivo de presentar cada vez una versión más elaborada del producto al cliente.

## 2.4. SISTEMA OPERATIVO ANDROID.

En la actualidad el uso de dispositivos móviles se ha extendido en la sociedad debido al avance de la tecnología y las facilidades que brindan a los usuarios. Esto ha desencadenado en que diferentes compañías empiecen una competencia en el desarrollo de SO. Los SO de los dispositivos móviles son similares a los SO que instalamos en nuestros computadores, la que es gestionar de la memoria del dispositivo o computador según el caso.

Android es un SO basado en el Kernel de Linux el mismo que permite una comunicación entre los componentes de Software y Hardware de los dispositivos móviles. Esta capa contienen los drivers necesarios para que cualquier componente de hardware pueda ser usado. (G, 2015)

Actualmente Android es uno de los SO más usados a nivel mundial llegando a ser instalado en cerca de mil millones de dispositivos. Además es el que cuenta con una gran cantidad de aplicaciones por lo que lo hace preferida por los usuarios. (M., 2015)

Esta es una de las razones por las que orientaremos nuestra aplicación a Android debido a que su uso extendido permite una mayor distribución de la aplicación y que esté disponible para la gran mayoría de usuarios.

#### **2.4.1. Soporte HTML5 para Android**

HTML5 se puede usar desde la versión 4 de Android denominada (Ice Cream Sandwich), existe otras configuraciones para hacer que funcione HTML5 pero en la actualidad ya no se encuentran dispositivos por esas versiones.

### **2.5. LENGUAJES DECLARATIVOS**

“Los lenguajes declarativos se basan en una especificación lógica de lo que la máquina debe hacer. Pero sin entrar en detalle en el cómo debe hacerlo. Es un lenguaje más cercano al lenguaje natural que al lenguaje máquina, que se basa en la entrada de predicados para, a través de inferencia, llegar a un resultado” (M, 2009).

### **2.5.1. Paradigma declarativo**

Este paradigma a diferencia de la programación estructurada, que se enfoca en abstraer literalmente los objetos de la vida real a la programación; trata de describir los objetos y los procesos; es decir: deja las clases y las soluciones indeterminadas y se vayan usando o ejecutando conforme la programación lo necesite. Es más complicado de implementar cuando se tiene una cultura de lenguajes imperativos o estructurados; existen muchas personas que califican a estos lenguajes como poco eficientes pero en algunos casos se ha demostrado que son más factibles que los otros tipos de lenguajes.

## **2.6. RUBY ON RAILS**

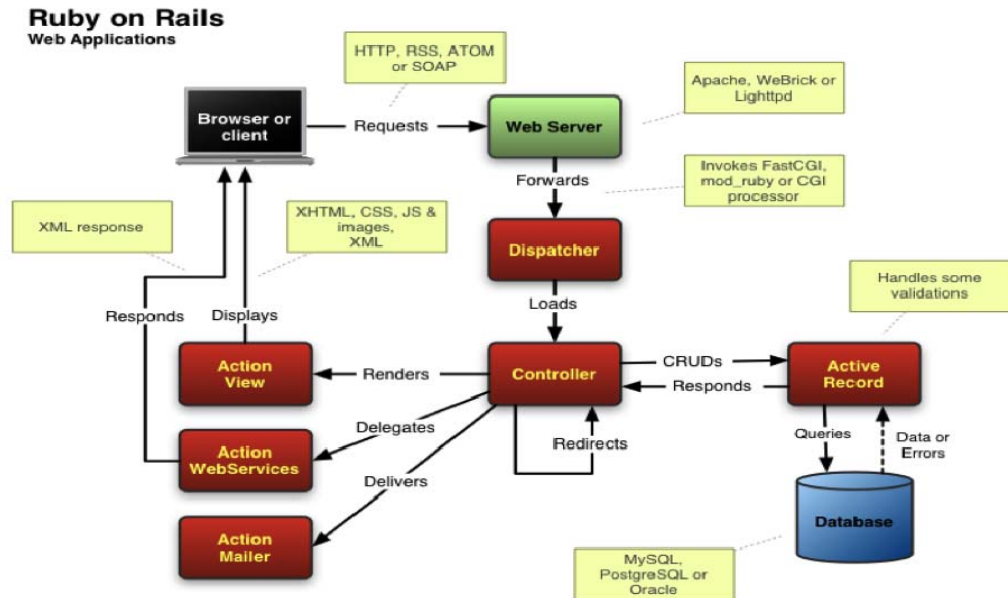
Ruby es un lenguaje que fue inventado en el año de 1995 por Yukihiro Matsumoto, el cual es un lenguaje interpretado el cual es una combinación de varios lenguajes favoritos de Matsumoto. (D G. , 2015)

Ruby es un lenguaje totalmente libre, el cual nos brinda total independencias para su modificación y distribución. Actualmente Ruby es un lenguaje que se está abriendo paso en el sin número de lenguajes de programación, ahora ocupa el puesto número 13 en el ranking mundial que hace referencia al crecimiento que ha tenido. Una de las razones del crecimiento de Ruby es el uso del Framework Ruby on Rails, esto por la cantidad y calidad de las aplicaciones desarrolladas en él. (Ruby, 2015)

En los inicios de Ruby su creador Matsumoto busco que su lenguaje sea “más poderoso que Perl y más orientado a objetos que Python”. Por este motivo en Ruby, todo es un objeto y se le puede asignar propiedades y acciones.

El surgimiento que ha tenido y que la facilidad de codificación es una de las ventajas que presente Ruby y unas de las razones por las que hemos propuesta el uso de este lenguaje para el desarrollo del sistema. (Matz e. A., 2001)

### 2.6.1. Arquitectura de Ruby on Rails



**Figura 3** Arquitectura de Ruby on Rails  
(Mejia, 2015)

Ruby on Rails tiene los siguientes componentes:

- Arquitectura Modelo Vista Controlador.
- REST para web service
- Soporta las principales bases de datos.
- Lenguaje del servidor es de código abierto.
- Convenciones para las configuraciones.
- Script auto generado para automatizar tareas.
- El uso de la máquina YAML , que es un formato de serialización de datos

legible por humanos

Todo lo antes mencionado se encuentra en los paquetes:

- Action Mailer
- Action Pack

- Action Controller
- Action Dispatcher
- Action View
- Active Model
- Active Record
- Active Resource
- Active Support
- Railties

## **2.6.2. Modelo Vista Controlador**

Ruby on Rails utiliza el patrón de arquitectura Modelo - Vista-Controlador (MVC) con el fin de mejorar la capacidad de mantenimiento de la aplicación. Centraliza el modelo de la lógica de negocio, la Vista gestiona la lógica visual, mientras que el controlador controla las peticiones para el flujo de la aplicación. El MVC permite una separación limpia de los requerimientos en el backend, así como en la forma en que se mantiene la lógica de negocio separados de vistas HTML, frontend.

### **2.6.2.1. Vista**

Todas las vistas son implementadas a través de Action View y Action Controller llamadas Action Pack. Todas las peticiones web son manejadas por el Action Pack, ya que puede dividir acciones que hace el controlador y las que hacen la vista. Normalmente, Action Controller estará preocupado por la comunicación con la base de datos y la realización de acciones CRUD cuando sea necesario. Acción View es responsable de la compilación de la respuesta y encargado de mostrar las respuestas. Básicamente se puede hacer peticiones directo a la base de datos sobre la vista y presentar dicha información sin necesidad de acceder al controlador.

### **2.6.2.2. Controlador**

Rails necesita saber dónde están las funciones que se van a usar ya que al no ser un lenguaje estructural no se guarda en memoria las dependencias ni las asociaciones entre clases. Por lo cual existe un archivo de configuración que ayuda a declarar dichas relaciones y poder trabajar con las peticiones web del servidor.

Los navegadores solicitan páginas de Rails haciendo una solicitud de una dirección URL utilizando un método HTTP específico, como GET, POST, PARCHE, PUT y DELETE. Cada método es una solicitud para realizar una operación con el recurso especificado. La ruta asigna un número de solicitudes referentes a las acciones en el controlador.

Cuando la aplicación recibe la ruta, Rails encuentra el método solicitado y dependiendo la solicitud html asigna el recurso necesario para poder ejecutar el método.

### **2.6.2.3. Modelo**

La capa Modelo lleva la lógica de negocio de la aplicación y las reglas para manipular los datos. En Ruby on Rails, se utilizan los modelos para gestionar la interacción con sus correspondientes elementos de la base de datos. Los modelos representan la información en la base de datos y ayuda hacer las validaciones apropiadas.

## **2.6.3. Persistencia**

### **2.6.3.1. Active Record**

Active Record es un patrón de arquitectura utilizado para administrar los datos en bases de datos relacionales a través de objetos. En Ruby on Rails el módulo Active Record ofrece mapeo objeto-relacional de clases. Este módulo se basa en la capa del Modelo que conecta las tablas de bases de datos con su representación en las



clases de Ruby. Rails proporcionan herramientas para implementar la funcionalidad CRUD sin tener que configurar o desarrollar métodos para dichas tareas.

El CRUD permite crear, leer, actualizar y eliminar registros de la base de dato a través de objetos de Ruby. Además, también proporciona capacidades de búsqueda avanzadas y la capacidad de crear relaciones o asociaciones entre modelos. Active Records se basa en gran medida en las convenciones sobre cómo las clases deben ser nombrados, las tablas de la base de datos, las claves foráneas y claves primarias. Sin embargo, el mapeo de base de datos puede llevarse a cabo utilizando configuraciones manuales, pero es altamente recomendable seguir las configuraciones automáticas de Rails.

Active Record se utiliza para crear clases del modelo, el cual contiene la lógica de negocio, ayuda a manejar las validaciones y las relaciones, y encapsula el acceso a datos, proporciona captadores y definidores, excepciones de llamada y también es compatible con varias bases de datos.

### **2.6.3.2. Mapeo Objeto-Relacional**

Active Record tiene incorporado un ORM, que facilita a la sincronización de los objetos con las tablas de la base de datos. Todas las propiedades y relaciones de los objetos en una aplicación pueden ser almacenadas y recuperadas de una base de datos sin necesidad de escribir sentencias SQL directamente y con menos código generado al tratar de acceder a una base de datos.

Entre las principales funciones de Active Record:

- Representar modelos y sus datos.
- Representar a las asociaciones entre estos modelos.
- Representar jerarquías de herencia a través de modelos relacionados.
- Validar los modelos antes de que se conservan en la base de datos.
- Realizar las operaciones de base de datos de una manera orientada a objetos.

### **2.6.3.3. Migraciones**

Rails y Active Record proporciona un lenguaje de dominio específico para la gestión de un esquema de base de datos llamada migraciones. Las migraciones se almacenan en archivos que se ejecutan sobre cualquier base de datos que Active Record es compatible. Esto depende de las gemas que tenga la aplicación.

### **2.6.3.4. Asociaciones**

En Rails, una asociación es una conexión entre los modelos. Las asociaciones se implementan mediante llamadas de tipo macro, por lo que se puede añadir de forma declarativa las características. Rails soporta seis tipos de asociaciones:

- `belongs_to`
- `has_one`
- `has_many`
- `has_many :through`
- `has_one :through`
- `has_and_belongs_to_many`

Todas estas asociaciones permiten interactuar con la base de datos y acceder a los objetos entre ellos.

### **2.6.3.5. Ejecutar query en active record**

Active record proporciona una mejor forma para extraer información de la base de datos que hará que casi nunca se use queries nativos ya que tiene compatibilidad con casi todas las base de datos.

Para recuperar objetos de la base de datos, Active Record ofrece varios métodos de búsqueda. Cada método de búsqueda le permite pasar argumentos en él para realizar ciertas consultas en su base de datos sin escribir SQL nativo. Los métodos que se pueden usar son:

- bind
- create\_with
- distinct
- eager\_load
- extending
- from
- group
- having
- includes
- joins
- limit
- lock
- none
- offset
- order
- preload
- readonly
- references
- reorder
- reverse\_order
- select
- uniq
- where

Todos los métodos mencionados devuelve una instancia declarada en Active Record, es decir; todos los métodos devuelven un objeto las relaciones para acceder a la base de datos.

## **2.6.4. Módulos de Rails**

### **2.6.4.1. Action Mailer**

Este módulo se encarga de prestar servicios de correo electrónico. Procesa mails entrantes y crea otros nuevos. Este módulo puede manejar texto simple o correos electrónicos complejos (ricos-formato). También tiene tareas comunes incorporadas, tales como, el envío de contraseñas olvidadas, mensajes de bienvenida, y el cumplimiento de necesidad de cualquier otra comunicación escrita. Acción Mailer se envuelve alrededor del controlador Acción. Proporciona métodos para desarrollar las plantillas del correo electrónico de la misma manera que la Action View utiliza para hacer las páginas web.

### **2.6.4.2. Action Pack**

El módulo de Action Pack proporciona las capas al controlador y vista de los patrones MVC. Estos módulos captan las solicitudes de los usuarios realizadas por el navegador y mapea estas peticiones a las acciones. Estas acciones se definen en la capa de los controladores y más tarde las acciones hacen una vista que se muestra en el navegador. Action Pack está dividido en 3 sub-módulos, que son: Action Dispatch, Action Controller, and Action View.

#### **2.6.4.2.1. Action Dispatch**

Rails usa una serie de enrutamientos propios del framework para determinar cómo se usaran las diferentes funciones, esto se explicará en el siguiente tema. Maneja el enrutamiento de solicitud del navegador web. Se analiza la solicitud Web y no el procesamiento avanzado alrededor de HTTP, como el manejo de las cookies, sesiones, métodos de petición, y así sucesivamente.

#### **2.6.4.2.2. Action Controller**

Después de que el enrutamiento ha determinado qué controlador va utilizar para cada petición, el controlador se encarga de dar sentido a la solicitud y de determinar la salida apropiada. Por suerte, Action Controller hace la mayoría del trabajo y utiliza las convenciones inteligentes para hacer esto lo más sencillo posible.

Para la mayoría de aplicaciones RESTFUL convencionales, el controlador recibirá la solicitud (esto es invisible para el desarrollador), y dependiendo el método se buscará guardará, eliminar, etc. los datos de un modelo y gestionará los datos para crear una salida HTML. Si el controlador tiene que hacer cosas un poco diferente, eso no es un problema, esto es sólo la forma más común para que un controlador funcione.

#### **2.6.4.2.3. Action View**

Este componente ayuda a presentar la información en la página web solicitada. Ayuda a Rails a generar plantillas, y componentes de otros lenguajes de marca utilizados en Ruby a HTML.

Hay tres plantillas esquemas en Rails, que son RHTML, rxml y RJS. El formato RHTML genera puntos de vista HTML a los usuarios con ERB (código rubí incrustado en HTML). El rxml se utiliza para construir documentos XML utilizando Ruby y rjs permite la creación dinámica de código JavaScript en Ruby útil para implementar la funcionalidad AJAX.

#### **2.6.5. RESTFUL Architecture**

REST significa Transferencia de estado representacional. REST es una alternativa a los servicios web, tales como SOAP y WSDL. Se basa en el protocolo HTTP para todas las operaciones CRUD: crear, leer, actualizar y borrar.

Los Servicios web RESTFUL son apropiados cuando los servicios son completamente sin estado, se tiene un ancho de banda limitado (es muy útil para los dispositivos móviles, ya que no hace la sobrecarga de otros protocolos como SOAP), cuando los datos no se genera dinámicamente por lo que podría ser almacenado en caché así mejorar el rendimiento. (Mejia, 2015)

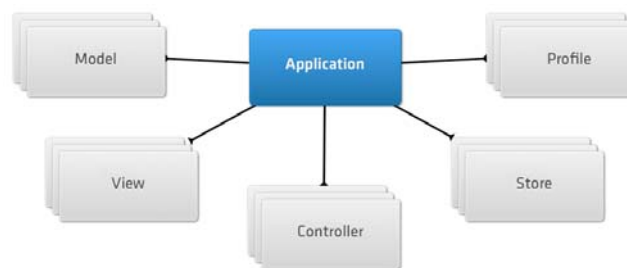
## 2.7. JAVASCRIPT

Es un lenguaje de programación que se ejecuta en el cliente, es decir que no corre sobre el servidor, su objetivo es crear efectos atractivos y dinámicos en las páginas web. (E., 2015)

En los últimos años JavaScript se ha convertido en un lenguaje integrador, debido a que se lo puede encontrar en muchos ámbitos, ya no solo en el internet, sino también en SO y dispositivos. Lo que ha permitido que la mayoría de validaciones se las realice en el cliente, permitiendo que los servidores queden libres de esta tarea. (DesarrolloWeb, 2015)

## 2.8. SENCHA TOUCH

### 2.8.1. Arquitectura Sencha Touch



**Figura 4 Arquitectura Sencha Touch**  
(code, 2015)

Como se puede apreciar en la imagen, Sencha Touch maneja el tradicional patrón MVC; pero también tiene dos nuevos componentes que son los Profile y Store.

Profile son archivos de configuración que dependiendo arquitectura de la maquina permite que la aplicación sea adaptable a los sistemas operáticos existentes para dispositivos móviles. Es decir; cuando la aplicación se ejecuta en una arquitectura de IOS (Apple) la aplicación será para dispositivos de la marca Apple. Si se tiene un entorno Windows se puede elegir entre Android y Windows Phone.

Store, permiten mantener los datos del servidor en el dispositivo y así mejorar el tiempo de respuesta. Guarda la data necesaria para el funcionamiento de la aplicación. También es el encargado de sincronizar los objetos literales JSON a modelos declarados en Sencha Touch, El objeto JSON no necesariamente va a tener toda la estructura de la tabla en la base datos; por lo cual el modelo de Sencha con el modele del servidor no necesariamente van a tener la misa estructura.

### **2.8.2. Store**

Los store es una fuente de datos lineales, dicho de otra forma un store es un arreglo (Array) de arreglos; ya que almacenan JSON que son objetos lineales. Dicho técnicamente un Store es un arreglo de instancias de un modelo.

Los Store sirven para sincronizar con los componentes visuales ya sea un DataView o un ListView; estos componentes ayudan a presentar la información en la aplicación por lo que dependiendo la complejidad de la información se entiende que estarán atados a un Store. Los store se pueden cargar su data por llamadas Ajax, peticiones al servidor o localmente.

### **2.8.3. Profile**

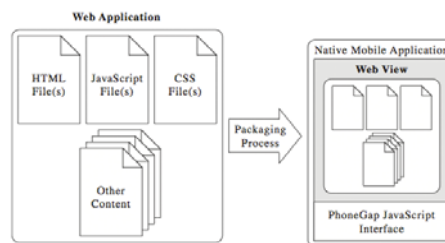
Es una tecnología que se usa para la compatibilidad de la aplicación en los diferentes dispositivos y plataformas.

## 2.9. CORDOVA

Córdoba es un marco de desarrollo de software de Adobe System, que se utiliza para desarrollar aplicaciones móviles. Para desarrollar aplicaciones usando Córdoba, el desarrollador no requiere tener conocimiento del lenguaje de programación móvil, pero sólo lenguas de desarrollo Web como, HTML, CSS y JScript. Cordova produce aplicaciones para todas las plataformas de sistemas operativos móviles populares como iOS, Android, BlackBerry y Windows Mobile OS. (PhoneGap, 2015)

Los dispositivos móviles han comenzado una nueva revolución en ingeniería de software. Estos dispositivos pequeños pero eficientes son capaces de ejecutar las aplicaciones creadas con lenguajes de programación de alto nivel. Las personas que poseen estos dispositivos tienden a utilizar a su máxima potencia ya que estos dispositivos ya son convenientes para usar en cualquier momento y en cualquier lugar.

### 2.9.1. Arquitectura de Córdoba



**Figura 5 Arquitectura Córdoba**  
(Arkaitzgarro, 2015)

Las Apps (software de aplicación) son desarrolladas tanto por la organización propietaria del dispositivo así como desarrolladas fuera de la organización. Lo que hace la arquitectura de Cordova es empaquetar el código de la aplicación; haciéndolo un ejecutable para que el dispositivo móvil lo reconozca.



Un número de sistemas operativos móviles bien reconocidos están disponibles en el mercado en ambas categorías tanto de propietario como de código abierto. La mayor parte móvil usas los sistemas operativos como:

- Android
- iOS
- BlackBerry
- Windows

Cada sistema operativo móvil ofrece su propio conjunto de herramientas y entornos para desarrollar aplicaciones que se ejecutan en ellos. Las apps hechas en un determinado sistema operativo no pueden ejecutar en cualquier otra plataforma, ya que son totalmente diferentes.

Desarrolladores tienden a cubrir todos los principales sistemas operativos móviles con el fin de aumentar interés entre sus usuarios.

De este modo se convierte en una tarea tediosa para desarrollar un programa de aplicación que pueden ejecutarse en todas las principales plataformas de sistemas operativos, copiando su apariencia, sensación y funcionalidad idéntica en todas las plataformas. Para este trabajo, un desarrollador necesita entender todas las plataformas y debe tener una buena comprensión de las principales herramientas de desarrollo para los diferentes sistemas operativos. . (PhoneGap, 2015)

Cordova puede ser visto como una solución a todos los problemas mencionados anteriormente. Cordova es un marco donde los desarrolladores realizan sus aplicaciones usando APIs web estándar para los principales sistemas operativos móviles. Es de código abierto y libre.

Los desarrolladores sólo necesitan saber programación web con HTML, CSS y JavaScript. Cordova se encarga del resto de la obra, como la apariencia de la aplicación y la portabilidad entre diferentes sistemas operativos móviles.

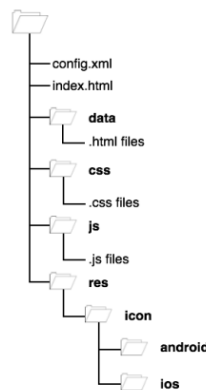
Utilizando Cordova, se puede crear aplicaciones para los principales OS móviles como iOS de Apple, Android, BlackBerry, Windows, etc. Y esto se puede hacer sin que el programador tenga conocimientos sobre cualquiera de las

plataformas anteriormente mencionadas, ni el desarrollador requiere conocer la programación para codificar la aplicación desde cero.

Cordova permite a sus usuarios subir el contenido de datos en el sitio web y automáticamente lo convierte en diversos archivos necesarios para cada plataforma.

Una de las ventajas es poder realizar pruebas en sitios web fuera de línea, se copian la app en el disco duro local y se accede siempre que el usuario necesite sin conexión a Internet por la red LAN. Del mismo modo, esta aplicación web offline le permitirá crear una aplicación web que se descarga en su totalidad a los dispositivos móviles de usuario y el usuario puede acceder a esa línea.

### 2.9.1.1. Fuera de Línea



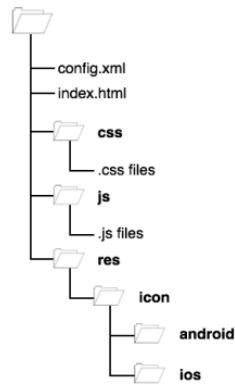
**Figura 6 Estructura Fuera de línea Córdoba**  
(Docs.phonegap.com, 2015)

La imagen siguiente representa la estructura de carpetas sin conexión de aplicaciones. A raíz directorio que requiere sólo dos archivos, config.xml y index.xml.

El config.xml contiene los valores de configuración de la aplicación, el archivo index.html es que contiene la página principal de la app.

Una cosa que aprender aquí importante es que todo el enlace interior tiene todos los archivos HTM, el contenedor tiene sólo ruta relativa.

### 2.9.1.2. En línea



**Figura 7 Estructura en línea Córdoba**  
(Docs.phonegap.com, 2015)

La siguiente imagen muestra la estructura de carpetas para nuestra aplicación para estar en el modo online. En modo en línea todos los contenidos de red se cargan desde el sitio web de Internet

Usted puede ver que la carpeta de datos no se encuentra en el modo de aplicación en línea, ya que todos los archivos están en el servidor actual y accesible a través de internet. El archivo `index.html` contiene enlaces reales, ya que contiene en el servidor web y todos sus enlaces son absolutas o usado con la etiqueta `href`.

Después de haber decidido el modo de su aplicación y organizado su archivo en el archivo estructura se mencionó anteriormente, es necesario comprimir el archivo con cualquier herramienta de compresión estándar y guárdelo.

Cordova utiliza herramientas para comprimir el proyecto para los diferentes sistemas operativos.

## 2.10. DISPOSITIVOS MÓVILES

Un dispositivo móvil es básicamente cualquier computadora de mano. Está diseñado para ser extremadamente portátil, a menudo ajustada en la palma de la mano o en el bolsillo. Algunos dispositivos móviles son más poderosos, permiten hacer muchas tareas, mismas que puedes hacer con un ordenador de sobremesa o portátil. Estos incluyen computadoras tablet, e-readers y smartphone.

### **Smartphone**

Un smartphone es un teléfono móvil integrado en un sistema operativo para móviles, con capacidad de computación más avanzada y conectividad que un teléfono normal.

Los primeros teléfonos inteligentes fueron una combinación de un asistente personal digital (PDA) y un teléfono móvil normal. Algunas funciones se añadieron en los modelos posteriores, como reproductores de medios portátiles, de gama baja las cámaras digitales compactas, de vídeo de bolsillo cámaras y unidades de navegación GPS para formar un dispositivo multi-uso, las pantallas táctiles de alta resolución y los navegadores web para de los navegadores estándar y las páginas móviles optimizados. Además, Wi-Fi proporciona acceso a datos de alta velocidad y móvil de banda ancha.

Los sistemas operativos móviles más habituales utilizados por los teléfonos inteligentes modernos incluyen el Android de Google, iOS de Apple, Symbian de Nokia, BlackBerry OS de RIM, Bada de Samsung, y Windows Phone de Microsoft. Tales sistemas operativos son capaces de ajustarse a muchos modelos de teléfonos diferentes. Además, por lo general cada dispositivo puede tener varios SO instalados en su vida útil.

## Tablet

Son más grandes que un teléfono móvil o un asistente digital personal. Son un tipo de dispositivos móviles integrado en una pantalla táctil plana y funciona principalmente al tocar la pantalla. No tiene teclado físico ya que tiene un teclado virtual. Pueden tener un lápiz óptico pasivo, o un lápiz digital. Normalmente, la

Tablet puede estar conectada a una red inalámbrica. En modelos híbridos un teclado desmontable está incluido de manera que la pantalla táctil se puede utilizar como una parte independiente.

Los primeros tipos del concepto tableta se originaron en los siglos 19 y 20, principalmente como prototipos e ideas conceptuales.

Los primeros dispositivos electrónicos portátiles comerciales basados en el concepto aparecieron a finales del siglo 20.

Apple lanzó el iPad con el sistema operativo y la tecnología de pantalla táctil en 2010 y se convirtió en el primer éxito de tablet computadora móvil para lograr fines comerciales en todo el mundo. Esto ha provocado un nuevo mercado para la Tablet y después de este éxito muchos otros fabricantes han producido versiones de su propio estilo como Samsung, HTC, Motorola, RIM, Sony, Amazon, HP, Microsoft, Archos, etc. Entre las tabletas de la competencia, la concentración principal en el uso de la SG fue en iOS (Apple), Android (Google), Windows (Microsoft) y QNX (RIM).

Las funciones típicas de los Tablet son:

- Funciones del navegador móviles inalámbricos (usando 2G , 3G , 4G o WiFi )
- Correo electrónico y redes sociales (por lo general con aplicaciones)
- Funciones del teléfono celular propias (mensajería, video llamada, el altavoz o auricular del teléfono móvil utiliza)
- Navegación por satélite GPS
- Cámara de video, la foto y la visualización de vídeo y edición
- Aplicaciones descargables (juegos , educación , servicios públicos)
- Función del reproductor multimedia.

- Pesa alrededor de una o dos libras ( 0,5 a 1 kilogramo )
- Duración de la batería de cada tres a doce horas según el uso.

### 3. CAPITULO III

#### 3.1. DESCRIPCIÓN DEL PROCESO ACTUAL

##### 3.1.1. Proceso de Trueque en redes sociales

Como se puede observar en la figura 7, en la actualidad las redes sociales ofrecen un sistema para poder intercambiar y vender productos de distintas variedades. Si uno desea intercambiar o adquirir algún producto debe acceder a los grupos de redes sociales y empezar una búsqueda publicación por publicación hasta encontrar el producto de su interés. Además que para eso necesita crearse una cuenta en dicha red social.



**Figura 8** Proceso actual de trueque (Facebook, 2015)

### 3.2. FASE DE EXPLORACIÓN CON PRODUCT CANVAS

La facilidad del Model Product Canvas es que permite la modificación de sus columnas, en la figura 8 se puede observar cual ha sido el formato que se adoptó para el diseño del software.


<b>Visión:</b> El aplicativo móvil para trueque o intercambio de bienes, será una herramienta social y tecnológica que permitirá mantener informado las 24 horas del día a los usuarios acerca de las ofertas y propuestas de bienes u objetos de valor, así como datos importantes sobre la persona que está ofertando y generando estándares para poder ofertar los productos		<b>Nombre:</b>  Yankiy 1.0	Lista de usuarios • Desarrollo limitado a 1 fase de proyecto, por analizar sistema de notificación
<b>Personas:</b> Usuarios de dispositivos móviles con sistema operativo android	<b>Historia de Usuario:</b> <ul style="list-style-type: none"> <li>• El usuario tendrá su perfil el cual accederá con su clave y nombre de usuario.</li> <li>• Usuarios podrán ver los productos ofertados por categorías</li> <li>• Se podrá visualizar cada producto por componentes</li> <li>• Cada usuario interesado por un producto podrá contactarse con el ofertante</li> <li>• Cada usuario podrá ver sus productos y dar de baja a cada producto</li> <li>• Cada usuario así como el ofertante recibirá un mail con los datos necesarios para que las dos partes se pongan en contacto.</li> <li>• Cada usuario podrá calificar la reputación y experiencia con el ofertante.</li> <li>• Cada usuario podrá ver su reputación.</li> <li>• Publicar bienes o productos.</li> </ul>	<b>Diseño:</b> 	<b>Restricciones:</b> <ul style="list-style-type: none"> <li>• Perfil dedicado para cada usuario.</li> <li>• Registro correcto de mail para recepción de información</li> <li>• Conectividad a internet</li> <li>• La aplicación será construida con framework híbrido</li> <li>• Usuarios podrán calificar una sola vez la reputación de otro usuario</li> </ul>

Figura 9 Producto Canvas

#### 3.2.1. Visión

Idea principal del producto.

<b>Visión:</b> El aplicativo móvil para trueque o intercambio de bienes, será una herramienta social y tecnológica que permitirá mantener informado las 24 horas del día a los usuarios acerca de las ofertas y propuestas de bienes u objetos de valor, así como datos importantes sobre la persona que está ofertando y generando estándares para poder ofertar los productos
--

Figura 10 Visión Product Canvas



### 3.2.2. Personas

Usuarios que van a usar el producto software.

Personas:	
• Usuario Registrado:	
Debe tener	
experiencia en el	
manejo de	
dispositivos móviles	
android. Y debe	
registrarse en portal	
web.	
• Usuario No	
Registrado: Debe tener	
experiencia en el	
manejo de	
dispositivos móviles	
android.	

**Figura 11 Personas Product Canvas**

### 3.2.3. Historia de Usuarios

Procesos que debe de hacer el producto.

**Continua**

Historia de Usuario:
• El usuario tendrá su perfil el cual accederá con su clave y nombre de usuario.
• Usuarios podrán ver los productos ofertados por categorías
• Se podrá visualizar cada producto por componentes
• Cada usuario interesado por un producto podrá contactarse con el ofertante
• Cada usuario podrá ver sus productos y dar de baja a cada producto
• Cada usuario así como el ofertante recibirá un mail con los datos necesarios para que las dos partes se pongan en contacto.
• Cada usuario podrá calificar la reputación y experiencia con el ofertante.
• Cada usuario podrá ver su reputación.
• Publicar bienes o productos.

**Figura 12 Historias de usuarios Product Canvas**

### 3.2.4. Diseño

Diseño de las pantallas de los del producto software.

**Continua**



**Figura 13 Diseño Product Canvas**

### 3.2.5. Restricciones

Lo que no debe hacer el producto software.

<p><b>Restricciones:</b></p> <ul style="list-style-type: none"> <li>• Perfil dedicado para cada usuario.</li> <li>• Registro correcto de mail para recepción de información</li> <li>• Conectividad a internet</li> <li>• La aplicación será construida con framework híbrido</li> <li>• Usuarios podrán calificar una sola vez la reputación de otro usuario</li> </ul>
--

**Figura 14 Restricciones Product Canvas**

### 3.2.6. Iteraciones de usuario

Actualizaciones del usuario, son cambios o nuevos requerimientos que se debe aplicar al producto software.

Iteraciones de usuarios
<ul style="list-style-type: none"><li>• La pantalla debe de tener un botón para poder contactar a los usuarios.</li><li>• Desarrollo limitado a 1 fase de proyecto, por analizar sistema de notificación</li><li>• Antes de poder comentar el usuario previamente debió de haber contactado al otro usuario.</li><li>• Se debe de mandar un mail con datos de usuario para el contacto.</li><li>• Se debe de cambiar el color del botón de contacto.</li><li>• El buscador debe de mostrar las categorías</li><li>• Las categorías también deben de estar en la pantalla de insertar.</li></ul>

Figura 15 Iteraciones de usuario Product Canvas

### 3.3. ANÁLISIS DE REQUERIMIENTOS.

#### 3.3.1. Descripción general

A continuación por desarrollo de la tesis se procederá a tratar de una forma más técnica a los requerimientos para desarrollar el software.

##### 3.3.1.1. Funcionalidad del producto.

El software servirá para realizar trueque de productos o bienes; esto se llevará a cabo por medio de una aplicación orientada a dispositivos móviles, la cual los usuarios deberán instalar en su dispositivo Android.

##### 3.3.1.2. Caracterización de los usuarios.

Los usuarios que han de usar la aplicación son los siguientes:

**Tabla 7**  
Características de usuarios registrados para la aplicación móvil

Tipo de usuario	Usuario registrado
Formación	No requiere una formación específica.
Habilidades	Debe tener experiencia en el manejo de dispositivos móviles android.
Actividades	<ul style="list-style-type: none"> <li>• Instalar la aplicación en el dispositivo.</li> <li>• Publicar productos a ser intercambiados</li> <li>• Responder preguntas de interesados en intercambiar productos.</li> <li>• Búsqueda de productos.</li> </ul>

**Tabla 8**

Características de usuarios no registrados para la aplicación móvil

Tipo de usuario	Usuario no registrado
Formación	No requiere una formación específica.
Habilidades	Debe tener experiencia en el manejo de dispositivos móviles android.
Actividades	<ul style="list-style-type: none"> <li>• Instalar la aplicación en el dispositivo.</li> <li>• Búsqueda de productos.</li> </ul>

**3.3.1.3. Historias de usuarios****Tabla 9**

Historia de usuario 1

Número: <b>1</b>	Usuario: <b>Usuario registrado</b>
Nombre historia: <b>Usuario tendrá su perfil</b>	
Prioridad en negocio: <b>Alta</b> (Alta/Media/Baja)	<b>Riesgo en Desarrollo: Alto</b> (Alto, Medio, Bajo)
<b>Puntos Estimados:</b>	<b>Iteración asignada: 2</b>
Programador responsable: <b>Pablo Villarruel – Carlos Naranjo</b>	
Descripción: <b>El usuario al establecer la conexión con el servidor solo podrá acceder a la información propia y no podrá acceder a la información de otro usuario</b>	
Entrada:	
<ul style="list-style-type: none"> <li>• <b>Usuario</b></li> <li>• <b>Contraseña</b></li> </ul>	
Salida:	
<ul style="list-style-type: none"> <li>• <b>Datos del usuario.</b></li> </ul>	
Error:	
<ul style="list-style-type: none"> <li>• <b>Mensaje de error</b></li> </ul>	
Observaciones: <b>Información se guarda en cache, de debe borrar información en caso de que el sistema no detecte ningún dato</b>	

**Tabla 10**  
**Historia de usuario 2**

Número: 2	Usuario: <b>Usuario registrado/ no registrado</b>
Nombre historia: <b>Visualización de productos</b>	
Prioridad en negocio: <b>Alta</b> (Alta/Media/Baja)	<b>Riesgo en Desarrollo: Alto</b> (Alto, Medio, Bajo)
<b>Puntos Estimados:</b>	<b>Iteración asignada: 1</b>
Programador responsable: <b>Pablo Villarruel – Carlos Naranjo</b>	
Descripción: <b>Para presentar la información los usuarios podrán acceder a un menú de opciones para poder filtrar toda la información que tendrá el servidor y así mejorar la respuesta.</b>	
Entrada:	
<ul style="list-style-type: none"> <li>• <b>Seleccionar todos los productos de la base de datos</b></li> </ul>	
Salida:	
<ul style="list-style-type: none"> <li>• <b>Todos los productos de la base de datos</b></li> </ul>	
Error:	
<ul style="list-style-type: none"> <li>• <b>Mensaje de error</b></li> </ul>	
Observaciones: <b>Ninguna</b>	

**Tabla 11**  
**Historia de usuario 3**

Número: 3	Usuario: <b>Usuario registrado/ no registrado</b>
Nombre historia: <b>Visualización de información en las búsquedas</b>	
Prioridad en negocio: <b>Alta</b> (Alta/Media/Baja)	<b>Riesgo en Desarrollo: Alto</b> (Alto, Medio, Bajo)
<b>Puntos Estimados:</b>	<b>Iteración asignada: 1</b>
Programador responsable: <b>Pablo Villarruel – Carlos Naranjo</b>	
Descripción: <b>Para poder mostrar la información se usará componentes gráficos que tengan un formato específico y pueda el usuario tener una vista preliminar del producto ofertado.</b>	
Entrada:	
<ul style="list-style-type: none"> <li>• <b>Seleccionar categoría</b></li> </ul>	
Salida:	
<ul style="list-style-type: none"> <li>• <b>Productos de esa categoría</b></li> </ul>	

---

Error:
<ul style="list-style-type: none"> <li>• <b>Mensaje de error</b></li> </ul>
Observaciones: <b>Ninguna</b>

---

**Tabla 12**  
Historia de usuario 4

---

Número: <b>4</b>	Usuario: <b>Usuario registrado</b>
Nombre historia: <b>Ponerse en contacto con ofertante</b>	
Prioridad en negocio: <b>Alta</b> (Alta/Media/Baja)	<b>Riesgo en Desarrollo: Alto</b> (Alto, Medio, Bajo)
<b>Puntos Estimados:</b>	<b>Iteración asignada: 1</b>
Programador responsable: <b>Pablo Villarruel – Carlos Naranjo</b>	
Descripción: <b>Cada usuario registrado podrá ponerse en contacta con el ofertante para poder realizar el proceso de trueque</b>	
Entrada:	
<ul style="list-style-type: none"> <li>• <b>Accionar botón de contacto</b></li> </ul>	
Salida:	
<ul style="list-style-type: none"> <li>• <b>Mensaje confirmación.</b></li> <li>• <b>Envío de mail a las dos partes (interesado y publicador).</b></li> </ul>	
Error:	
<ul style="list-style-type: none"> <li>• <b>Mensaje de Error</b></li> </ul>	
Observaciones: <b>Ninguna</b>	

---

**Tabla 13**  
Historia de usuario 5

---

Número: <b>5</b>	Usuario: <b>Usuario registrado</b>
Nombre historia: <b>Dar de baja a productos publicados</b>	
Prioridad en negocio: <b>Alta</b> (Alta/Media/Baja)	<b>Riesgo en Desarrollo: Alto</b> (Alto, Medio, Bajo)
<b>Puntos Estimados:</b>	<b>Iteración asignada: 1</b>
Programador responsable: <b>Pablo Villarruel – Carlos Naranjo</b>	
Descripción: <b>Los usuarios que hayan publicado un producto podrán eliminar la publicación para que no se muestre más a los demás usuarios.</b>	
Entrada:	
<ul style="list-style-type: none"> <li>• <b>Seleccionar publicación.</b></li> </ul>	

---



Salida:
<ul style="list-style-type: none"> <li>• <b>Publicación para poder eliminar.</b></li> </ul>
Error:
<ul style="list-style-type: none"> <li>• <b>Mensaje de error</b></li> </ul>
Observaciones: <b>Ninguna</b>

**Tabla 14**  
**Historia de usuario 6**

Número: <b>7</b>	Usuario: <b>Usuario registrado</b>
Nombre historia: <b>Calificar a un usuario</b>	
Prioridad en negocio: <b>Alta</b> (Alta/Media/Baja)	<b>Riesgo en Desarrollo: Alto</b> (Alto, Medio, Bajo)
<b>Puntos Estimados:</b>	<b>Iteración asignada: 1</b>
Programador responsable: <b>Pablo Villarruel – Carlos Naranjo</b>	
Descripción: <b>Cada usuario registrado podrá calificar a otro usuario con el fin de saber la reputación de los usuarios y su experiencia</b>	
Entrada:	
<ul style="list-style-type: none"> <li>• <b>Seleccionar publicación</b></li> </ul>	
Salida:	
<ul style="list-style-type: none"> <li>• <b>Publicación seleccionada.</b></li> <li>• <b>Mensaje de confirmación</b></li> </ul>	
Error:	
<ul style="list-style-type: none"> <li>• <b>Mensaje de error.</b></li> </ul>	
Restricción:	
<ul style="list-style-type: none"> <li>• <b>Haber contactado previamente al usuario que publica el producto.</b></li> </ul>	
Observaciones: <b>Ninguna</b>	

**Tabla 15**  
**Historia usuario 7**

Número: <b>8</b>	Usuario: <b>Usuario registrado</b>
Nombre historia: <b>Ver perfil</b>	
Prioridad en negocio: <b>Alta</b> (Alta/Media/Baja)	<b>Riesgo en Desarrollo: Alto</b> (Alto, Medio, Bajo)
<b>Puntos Estimados:</b>	<b>Iteración asignada: 1</b>
Programador responsable: <b>Pablo Villarruel – Carlos Naranjo</b>	
Descripción: <b>Un usuario podrá conocer la experiencia que tuvieron los demás con usuarios con él. Podrá ver a través de ponderación si el usuario es confiable o no.</b>	

Esto se mostrará a través de estrellas del 1 al 5.

Entrada:

- **Seleccionar opción de perfil**

Salida:

- **Datos del usuario**

Error:

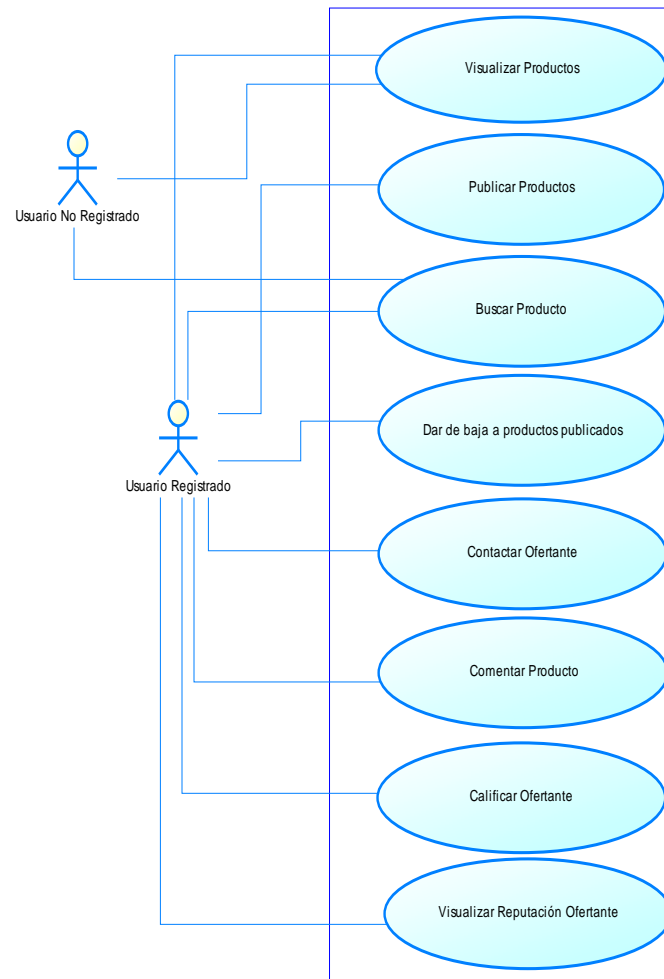
- **Mensaje de error**

Observaciones: **Ninguna**

**Tabla 16**  
Historia usuario 8

Número: <b>9</b>	Usuario: <b>Usuario registrado</b>
Nombre historia: <b>Publicar un producto</b>	
Prioridad en negocio: <b>Alta</b> (Alta/Media/Baja)	<b>Riesgo en Desarrollo: Alto</b> (Alto, Medio, Bajo)
<b>Puntos Estimados:</b>	<b>Iteración asignada: 1</b>
Programador responsable: <b>Pablo Villarruel – Carlos Naranjo</b>	
Descripción: <b>Los usuarios registrados podrán publicar su producto o bien a través de la aplicación móvil.</b>	
Entrada:	
<ul style="list-style-type: none"> <li>• <b>Encabezado de publicación</b></li> <li>• <b>Descripción.</b></li> <li>• <b>Categoría.</b></li> <li>• <b>Fotos(no pueden estar presentes)</b></li> </ul>	
Salida:	
<ul style="list-style-type: none"> <li>• <b>Mensaje de confirmación.</b></li> </ul>	
Error:	
<ul style="list-style-type: none"> <li>• <b>Mensaje de error</b></li> </ul>	
Observaciones: <b>Ninguna</b>	

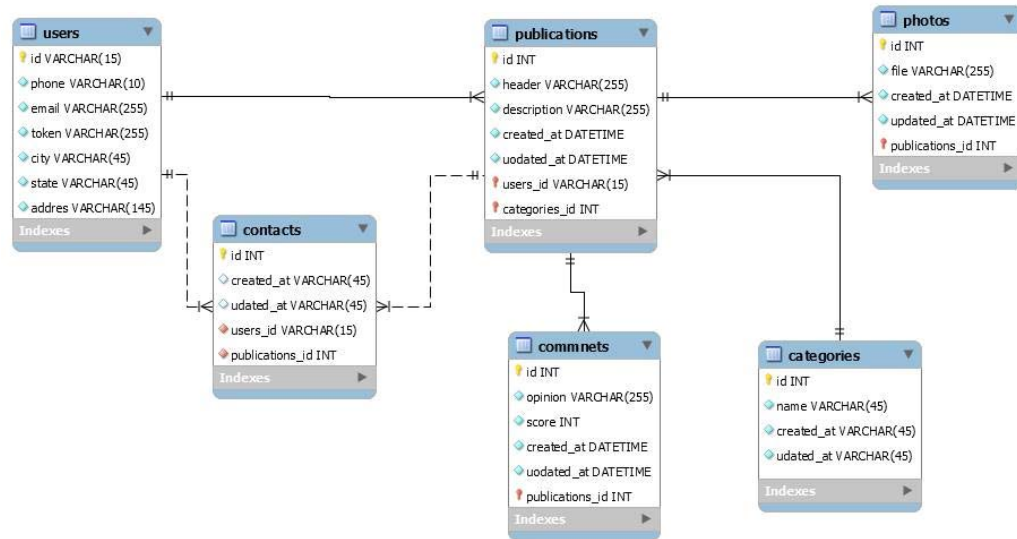
### 3.3.1.4. Caso de uso



**Figura 16 Casos de uso**

### 3.3.1.5. Diagrama físico de base de datos

**Continúa**



**Figura 17 Diagrama físico de base de datos**

### 3.3.1.6. Restricciones

- Fácil uso del cliente rest,
- Registro correcto del mail para recepción de información.
- La conectividad a internet debe ser rápida para que no afecta la respuesta de los objetos JSON.
- Seguridad en al momento de realizar el ingreso al servidor.
- Seguridad al momento de realizar las peticiones.

## 4. CAPITULO IV

### 4.1. GESTIÓN DEL CONOCIMIENTO

#### 4.1.1. Ruby

Para el desarrollo del servidor usado el lenguaje de programación Ruby con el framework Rails es necesario definir primero algunos puntos importantes. Cabe resaltar que esta información es desarrollada sobre el análisis obtenida en tutoriales e información que hemos leído en revistas páginas web y libros, así también como nuestras experiencias en el desarrollo.

Como ya se ha dicho antes Ruby es "un lenguaje de programación interpretado para la programación orientada a objetos rápida y fácil." (Matz, 2001), y uno de los frameworks más usados es Rails.

Ruby on Rails se desenvuelve con dos filosofías claves, las cuales guían en el desarrollo en este framework, estas son:

- DRY (Don't repeat yourself): esta filosofía busca evitar que el código sea reescrito.
- Convention Over Configuration: lo que busca esta filosofía es evitar invertir más del tiempo necesario en configurar la aplicación y dedicarlo más al desarrollo. (Heinemeier, 2015)

#### 4.1.1.1. Comandos básicos

Es necesario tener instalado ROR y tener noción o haber leído un poco de la sintaxis.

- 1) Crear el proyecto:

```
rails new prueba
```

Se genera una nueva carpeta dentro del directorio en que estás posicionado.

2) Ubicarse en el directorio recién creado. Ejemplo para plataformas

```
cd prueba
```

3) Crear una tabla Empleados:

```
rails g scaffold empleado Nombre:string direccion:string telefono:string  
fechadeEntrada:date
```

4) Se migra (se hace corresponder el modelo del programa con una base de datos) con el comando:

```
rake db:migrate
```

5) Desde el terminal se lanza el servidor:

```
rails s
```

6) Listo. Ahora se ingresa a la siguiente dirección en el navegador para utilizar la aplicación ya desarrollada:

```
localhost:300/Empleados
```

#### **4.1.1.2. Gemas**

Al igual que cualquier ID de programación Ruby utiliza sus propias librerías o en este caso Gemas las cuales proveen de diferentes servicios a la aplicación.

A diferencia de otros ID's de programación en los cuales las librerías solo son agregadas y compiladas en el proyecto como en Java por ejemplo, en Ruby es necesario instalar las gemas extras que se necesiten, las cuales son descargadas de internet, instaladas y agregadas tanto a las gemas del proyecto como a las que Ruby

almacena. Las gemas que un proyecto usará se determinan a través del archivo GemFile, el cual especifica la versión de cada una de ellas.

Existen dos tipos de gemas, las que por sí solas pueden realizar un proceso entero sin que se necesite alguna configuración, como es la gema que crea el proyecto y las gemas que son usadas dentro del proyecto, estas gemas necesitan de elementos de entrada y salida para poder funcionar.

Cuando compartimos un proyecto en Ruby es difícil que las gemas también sean compartidas, si bien podemos compartir la programación las gemas es algo que se configura dentro del propio Ruby por lo que es necesario que se las instale. Ruby provee la sentencia "bundle" para volver a instalar las gemas y sus dependencias; este elemento busca las gemas faltantes, las instala y configura dentro de Ruby sin que el programador tenga que estar buscando cada una de las gemas. (A. B. , 2015)

Lista de gemas para el proyecto:

**Mysql2:** Esta gema nos ayuda a la conexión a la base de datos. Por defecto esta gema se instala cuando se crea un proyecto. (Mario, 2015)

**Byebug:** Ruby al ser un lenguaje interpretado no permite hacer debug al código por lo cual, esta gema ayuda a poner puntos de parada y examinar que pasa en el código. (Mario, 2015)

**Devise:** Permite gestionar usuarios en los proyectos. Por defecto esta gema crea en la base de datos la tabla usuario y permite tener muchas acciones predeterminadas como: Validar datos, correo de confirmación de cuenta, banderas para notificaciones, encriptar password y canal de comunicación para el servidor. (Plataformatec, 2015)

**Carrierwave:** Esta gema permite guardar archivos a la base de datos, tiene métodos predeterminados que permiten subir archivos por medio del servidor. (Thomasdbs, 2015).

**Active Admin:** Permite gestionar todas las tablas de la base de datos. Así como generar CRUD rápidamente.

#### 4.1.1.3. **Helpers**

Ayudan a gestionar de mejor manera los contenidos de las aplicaciones. En este caso vamos a usar los siguientes helpers:

**CORS:** Cross-origin resource sharing, es una especificación de la tecnología de navegador web que define formas para que un servidor web permita el acceso a los recursos a través de una página web de un dominio diferente. Dicho acceso de otro modo sería prohibido por la misma política de origen. CORS define una manera en la que el navegador y el servidor pueden interactuar para determinar si se permite o no la solicitud de origen. Es un compromiso que permite una mayor flexibilidad, pero es más seguro que simplemente permitir todas las solicitudes.

CORS permite especificar las peticiones al servidor: POST, PUT, GET, DELETE.

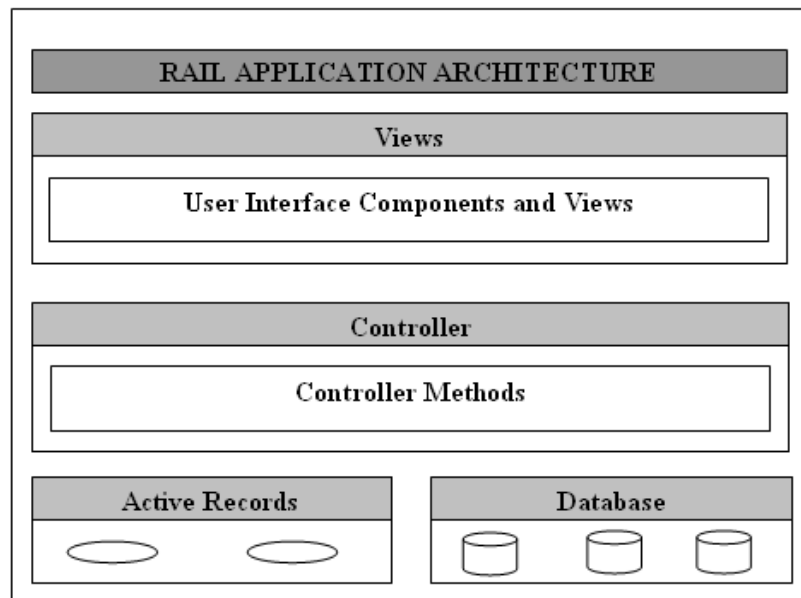
**Mailers:** Ruby on Rails permite tratar a los mails como una clase más, lo que facilita su implementación. Ya que el mail se convierte en una instancia. Rails concatena la vista del mail con el envío por lo cual permite implementar fácilmente los mails.

#### 4.1.1.4. **Patrones de diseño.**

Los patrones de software son soluciones para problemas típicos que se presentan en el desarrollo las cuales ya han sido probadas, y han mostrado su efectividad. El punto de un patrón de diseño es reducir y reutilizar el código evitando la redundancia en las aplicaciones (Rubenfa, 2015), y esto es lo que pretende Ruby on Rails con su filosofía.

Rails es un framework orientado al desarrollo de aplicaciones web de tal forma que al crear un nuevo proyecto este es creado usando patrón de diseño MVC el cual podemos visualizar a continuación. (tutorialspoint, 2015)





**Figura 18 Patrón MVC Ruby on Rails**  
([tutorialspoint.com/ruby-on-rails/rails-framework.htm](http://tutorialspoint.com/ruby-on-rails/rails-framework.htm), 2015)

#### 4.1.1.4.1. Base de datos y Active Record.

La base de datos es la M en el modelo MVC. La conexión a la base de datos de un sistema en Ruby on Rails se lo hace con Active record, el cual es el responsable de representar la lógica del negocio. Active Record facilita la creación y el uso de objetos que requieren ser almacenados, en si es una descripción de un ORM (objeto Relational Mapping). (D H. , 2015)

Este conecta las clases a las tablas de la base de datos relacionales para establecer una capa casi cero configuraciones, lo único que es necesario indicar el usuario y la clave de la base de datos. (Sim, 2015)

Active record usa algunas convenciones para dar nombres a las Clases y a los Modelos. Esto se debe a que al ser un ID interpretativo pluraliza por si las nombres de los Modelos a partir de las Clases. Es necesario indicar que este ide se realiza a partir de palabras en inglés, por ejemplo tenemos la clase **User** la cual al ser

pluralizada y convertida en una tabla de la base pasa a ser llamada **users**, como se puede ver en la Figura 8. (Rubyonrailsthissur, 2015)

Model / Class	Table / Schema
Article	articles
LineItem	line_items
Deer	deers
Mouse	mice
Person	people

**Figura 19 Ejemplos de pluralización**  
([http://guides.rubyonrails.org/active\\_record\\_basics.html](http://guides.rubyonrails.org/active_record_basics.html), 2015)

#### 4.1.1.4.2. Modelos con Active Record

Los modelos en Active Record se los crea a través de la clase ActiveRecord::Base, esta clase se vincula con el modelo y crea los diferentes atributos que se especifican

Los objetos de la clase Active Record no especifican directamente sus atributos, si no que los infiere directamente de la tabla que la vincula. Cuando creamos, añadimos o eliminamos campos estos se reflejan directamente en ella. (Rubydoc, 2015)

Active Record crea automáticamente un método que permite a la aplicación leer y manipular los datos dentro de sus tablas, así mismo, es posible realizar validaciones de la estructura de los objetos que se pretende guardar o actualizar. (D H. , 2015)

Algo importante que tiene Active Record es algo que se llama “Migrations”, que es un " lenguaje de dominio específico para la gestión de un esquema de base de

datos", lo que quiere decir esto es que Active Record nos permite realizar cambios directos a la base como añadir o retirar campos de una tabla, creando un archivo donde se guardan todas las modificaciones que se hagan de tal forma que sea posible hacer un rollback si fuera necesario. (D H. , 2015)

#### **4.1.1.4.3. Controladores.**

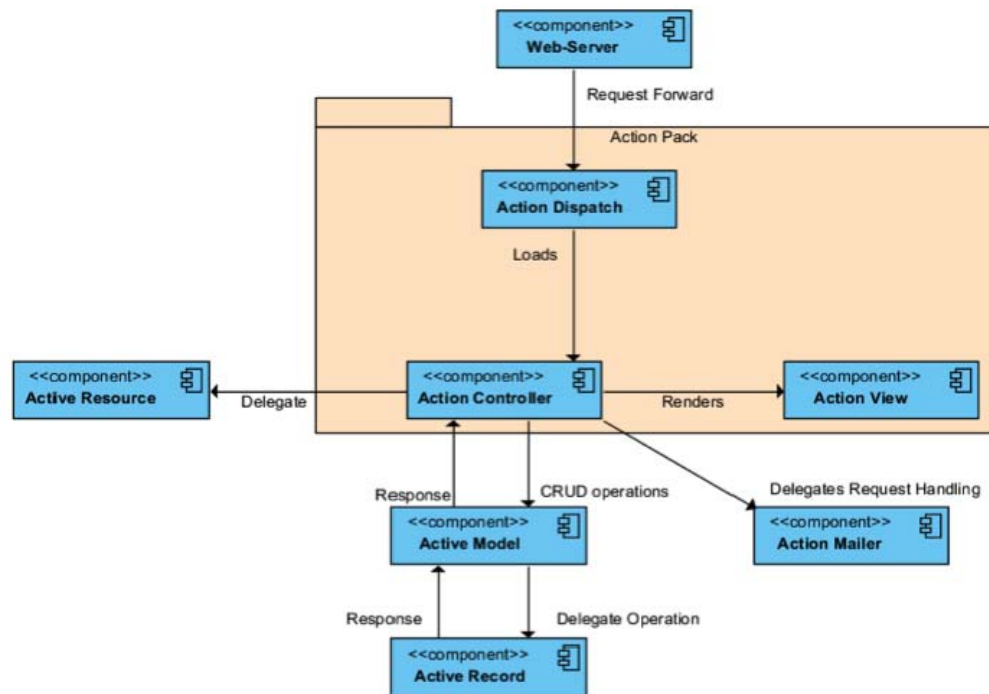
El controlador es la C en el modelo MVC, es el responsable de ejecutar las peticiones y entregar una respuesta apropiada. La ventaja que proporciona Ruby es que la mayoría de peticiones por nosotros y utiliza convenciones inteligentes para hacerlo sencillo.

El controlador recibe las solicitudes y las procesa por ejemplo buscar, guardar, eliminar y borrar elementos de la base no es complicado debido a que esto ya lo implementa Ruby por sí solo.

De tal modo que el controlador es un intermediario entre la capa de persistencia y la capa de vista, la cual permite al desarrollador invertir más tiempo en el desarrollo de la lógica pues los manejos de datos los hace Ruby. (Rubyonrails, 2015)

#### **4.1.1.5. Interfaces**

La siguiente figura es una visión dinámica del sistema y presenta los componentes, las interfaces y conectores. En la representación de interfaces está en formato UML para su rápido entendimiento. Esto proporciona una representación compacta un sistema ROR. Los tipos de conectores se representan como asociaciones e instancias de conectores como enlaces a ser el consistente con la notación UML que estamos utilizando.



**Figura 20 Interfaces y componentes aplicaciones ROR**  
(Mejia, 2015)

#### 4.1.1.6. JSON Web Services

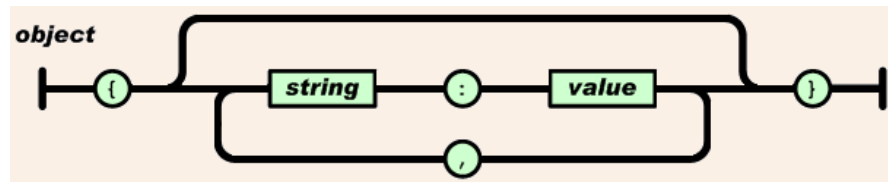
JSON (JavaScript Object Notation) es un formato para el intercambio de datos que tiene un formato ligero, el cual es fácil de escribirlo y leerlo para los humanos y para las máquinas es de fácil interpretación.

JSON está constituido por dos estructuras:

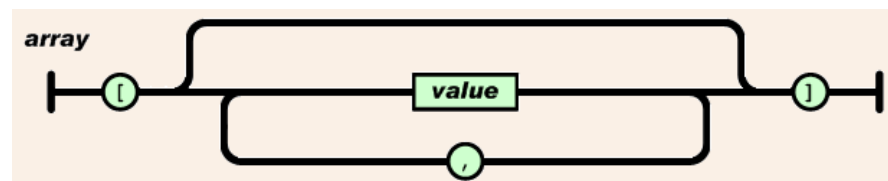
- Un elemento tipo objeto.
- Una lista ordenada de valores.

Estas dos tipos de estructuras es un concepto que se maneja en casi todos los lenguajes de programación sea de una forma u otra, y esta es una de las razones para que usa estas estructuras.

JSON tiene su propia forma de armar un Objeto o una lista, y esto se lo hace como se indica en las Figuras 9 y 10 respectivamente.



**Figura 21 Estructura de objeto en JSON**  
(Json, 2015)



**Figura 22 Estructura de lista en JSON**  
(Json, 2015)

#### 4.1.1.6.1. Clase ActionController::Base

Para usar un Web Service dentro de Rails, es necesario usar la clase ActionController, esta clase es el core de las aplicaciones que hacen solicitudes web.

Esta clase se compone por una o más acciones que se ejecutan bajo petición, y dependiendo de ella la dirige a una acción o un template. Estos pedidos se los recibe

a través de funciones que se hacen accesibles automáticamente por el servidor y la las configuraciones de ruteo.

Cada acción es procesada por el framework ActionController, el cual extrae el nombre de la acción de cada petición que se realiza, una vez que llega la acción ha sido reconocida, se toma el resto de los parámetros que viene en la cadena HTTP son puestos a disposición y la acción es ejecutada.

Las sesiones permiten almacenar objetos entre cada solicitud, esto sirve para preparar un objeto que todavía no se ha terminado de construir, esto es útil para mantener objetos que cambian muy poco en el tiempo, pero se debe tener en cuenta que estos objetos no pueden ser mantenidos en cache porque pueden ser alterados sin saberlo.

La respuesta que se da a una cada acción mantiene los encabezados y el que se enviarán al usuario. El objeto de respuesta es generado de forma automática mediante el uso de renders y redirecciones, esto no requiere la intervención del usuario.

Una de las funcionalidades más versátiles con las que cuenta Ruby es que puede enviar un template a un cliente para ser visualizado. Esta configuración se la hace de forma automática y los controles pasan como objetos a través de variables de instancia. (Ruby on Rails, 2015)

#### **4.1.2. JavaScript**

JavaScript es un lenguaje de programación interpretado, por lo que no es necesario compilarlos para ser ejecutados. Esto es una gran ventaja debido a que la programación se puede probar directamente en cualquier navegador sin necesidad de un intermediario. (Librosweb, 2015)

En los últimos años JavaScript ha sido una herramienta esencial para el desarrollo de aplicaciones móviles debido a que ha incorporado una variedad de elementos a su portafolio para móviles como es el uso de videos o geo-localización. Esto se ha potenciado gracias a frameworks como Cordova que ha ayudado a los

desarrolladores a crear aplicaciones que cruzan la línea entre página web y aplicación.

Dentro del proyecto desarrollado existe un uso extendido de JavaScript debido a que encuentra en todo el nivel de programación (Orenfarhi, 2015)

Para nuestro caso JavaScript es framework en el que se base Sencha Touch el cual usamos para desarrollar nuestro front-end. Sencha contiene un paquete de datos robusta el cual se puede consumir datos desde cualquier back-end. A demás cuenta con un paquete de gráficos y componentes avanzados que permite que sean visibles en dispositivos móviles. (Sencha Touch, 2015)

#### **4.1.3. Sencha Touch**

Como ya se explicó en temas anteriores Sencha Touch usa el patrón de diseño MVC.

##### **4.1.3.1. Patrón MVC Sencha Touch.**

Modelo –Define objetos con los campos de sus datos (por ejemplo, un modelo de usuario con campos de usuario de nombre y contraseña).

Los modelos son también un lugar ideal para cualquier lógica de datos que pueda necesitar, tales como la validación, conversión, etc.

Vista - Una vista es cualquier tipo de componente que se representa visualmente. Son las pantallas que tendrá la aplicación.

Controlador – Es el encargado de vincular las acciones que se hace en la vista con los modelos para gestionar la información.

#### 4.1.3.2. Comandos básicos

Para construir una aplicación desde cero y tener toda la estructura necesaria se ejecuta el siguiente comando:

Crea una aplicación dentro del directorio especificado:

```
sencha -sdk local/path/to/ExtJS generate app MyApp MyApp
```

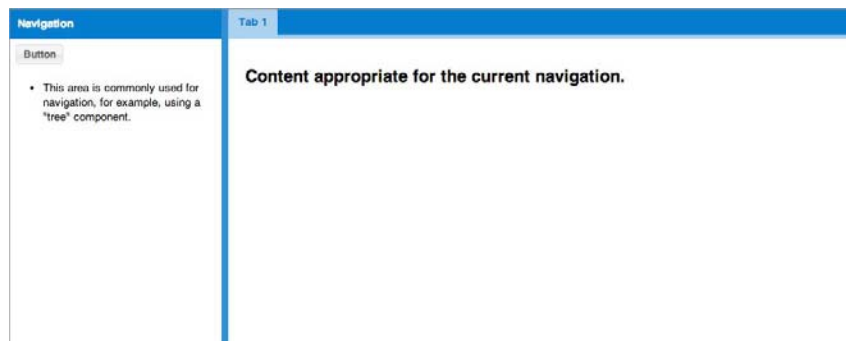
Acceder a la nueva carpeta creada:

```
cd app
```

Ejecutar servidor web:

```
sencha app watch
```

Dependiendo la versión de Sencha que usen se mostrará una pantalla de inicio, pudiendo ser esta pantalla vacía.

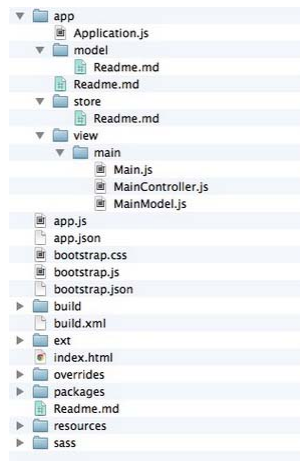


**Figura 23 Pantalla de inicio Sencha Touch**

#### 4.1.3.3. Estructura de carpetas

Sencha Touch proporciona una estructura parecida a Ruby, dentro de la carpeta app podremos encontrar toda la estructura de la aplicación;





**Figura 24 Estructura aplicación Sencha Touch**

#### 4.1.3.4. Archivos principales

##### **App.js**

Es el archivo principal donde está especificado el nombre de la aplicación y cuál será la vista principal. Se mostrará estructura como esta:

```

/*
 * This file is generated and updated by Sencha Cmd. You can edit this file as
 * needed for your application, but these edits will have to be merged by
 * Sencha Cmd when upgrading.
 */
Ext.application({
    name: 'MyApp',

    extend: 'MyApp.Application',

    mainView: 'MyApp.view.main.Main'

    //-----
    // Most customizations should be made to MyApp.Application. If you need to
    // customize this file, doing so below this section reduces the likelihood
    // of merge conflicts when upgrading to new versions of Sencha Cmd.
    //-----
});

```

**Figura 25 Código archivo app.js**

## Application.js

Este archivo permitirá la aplicación ubicar cuales son los modelos y las dependencias.

```
Ext.define('MyApp.Application', {
    extend: 'Ext.app.Application',

    name: 'MyApp',

    stores: [
        // TODO: add global/shared stores here
    ],

    launch: function () {
        // TODO - Launch the application
    }
});
```

**Figura 26** Código archivo application.js

### 4.1.3.5. Estructura de una vista

Una vista no es más que un componente, que es una subclase de Ext.Component y contiene todos los aspectos visuales de la aplicación. Si abre main.js de la aplicación de arranque en la carpeta "main", debería ver el código como se muestra en la siguiente figura:

**Continua**

```

Ext.define('MyApp.view.main.Main', {
    extend: 'Ext.container.Container',

    xtype: 'app-main',

    controller: 'main',
    viewModel: {
        type: 'main'
    },

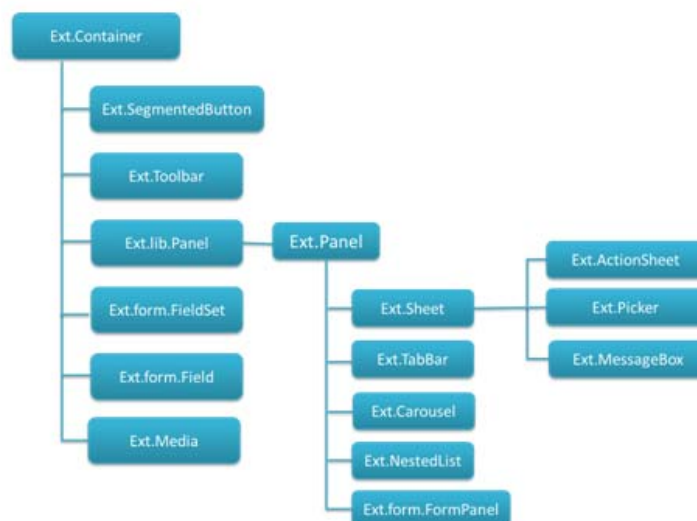
    layout: {
        type: 'border'
    },

    items: [{
        xtype: 'panel',
        bind: {
            title: '{name}'
        },
        region: 'west',
        html: '<ul>...</ul>',
        width: 250,
        split: true,
        tbar: [{
            text: 'Button',
            handler: 'onClickButton'
        }]
    }, {
        region: 'center',
        xtype: 'tabpanel',
        items: [{
            title: 'Tab 1',
            html: '<h2>Content ...</h2>'
        }]
    }]
});

```

**Figura 27 Estructura de una Vista Sencha Touch**

Aquí se nota que la vista no debe tener ninguna lógica de la aplicación. Los componentes visuales dentro de la vista tienen una jerarquía y para su implementación debe estar estructurado el js de la siguiente forma:



**Figura 28 Estructura componentes Sencha Touch**  
(JavaTechig, 2015)

#### 4.1.3.6. Controladores

Los controladores son diseñados para:

- Haga la conexión a puntos de vista el uso de " listener" y "referencia" obvias de las configuraciones.

Aprovecha el ciclo de vida de las vistas para gestionar automáticamente su ViewController asociado. A partir de instancias a las peticiones, Ext.app.ViewController está ligado al componente que hace referencia a ella. Una segunda instancia de la misma clase de vista tendría su propia instancia ViewController. Cuando estos puntos de vista son destruidos, su ejemplo ViewController asociado será destruida también.

- Proporcionar encapsulación para hacer vistas con anidación intuitiva.

En la siguiente figura se puede apreciar como es la estructura de los controladores:

```
Ext.define('MyApp.view.main.MainController', {
    extend: 'Ext.app.ViewController',

    requires: [
        'Ext.MessageBox'
    ],

    alias: 'controller.main',

    onClickButton: function () {
        Ext.Msg.confirm('Confirm', 'Are you sure?', 'onConfirm', this);
    },

    onConfirm: function (choice) {
        if (choice === 'yes') {
            //
        }
    }
});
```

**Figura 29 Estructura de controlador Sencha Touch**

#### 4.1.3.7. Modelos

Representa la data de un objeto que se vaya a usar en la aplicación. Por lo general son utilizados con los Store. Lo que ayuda a que la información sea consumida por algún componente visual para su presentación. El module está estrictamente ligado a la estructura de objeto JSON

```
Ext.define('MyApp.model.User', {
    extend: 'Ext.data.Model',
    fields: [
        {name: 'name', type: 'string'},
        {name: 'age', type: 'int'}
    ]
});
```

**Figura 30** Estructura modelos en Sencha Touch

#### 4.1.3.8. Stores

Los store permiten tratar a la data de la aplicación como objetos lineales en un Array. Estos store se pueden asociar cualquier componente visual permitiendo que la información se distribuya. Incluso se puede cargar store globales para que la aplicación no haga tantas peticiones al servidor.

La estructura de un store es la siguiente:

```
Ext.define('MyApp.store.User', {
    extend: 'Ext.data.Store',
    model: 'MyApp.model.User',
    data : [
        {firstName: 'Seth', age: '34'},
        {firstName: 'Scott', age: '72'},
        {firstName: 'Gary', age: '19'},
        {firstName: 'Capybara', age: '208'}
    ]
});
```

**Figura 31** Estructura store en Sencha Touch

## 4.2. IMPLEMENTACIÓN

### 4.2.1. Cliente Rest

Ruby facilita la implementación del cliente Rest ya que trabaja con los protocolos HTTP para casi todos los métodos implementados en los controladores. Esto lo hace a través de las peticiones PUT, GET, POST, DELETE. Por lo cual el cliente solo consume la url del método del servidor.

```

developer@developer:~/workspace/tests/test$ rake routes
Prefix Verb      URI Pattern
admin_root GET      /admin(.:format)
batch_action_admin_categories POST     /admin/categories/batch_action(.:format)
admin_categories GET      /admin/categories(.:format)
admin_categories POST     /admin/categories(.:format)
new_admin_category GET      /admin/categories/new(.:format)
edit_admin_category GET      /admin/categories/:id/edit(.:format)
admin_category GET      /admin/categories/:id(.:format)
admin_category PATCH    /admin/categories/:id(.:format)
admin_category PUT      /admin/categories/:id(.:format)
admin_category DELETE   /admin/categories/:id(.:format)
batch_action_admin_contacts POST     /admin/contacts/batch_action(.:format)
admin_contacts GET      /admin/contacts(.:format)
admin_contacts POST     /admin/contacts(.:format)
new_admin_contact GET      /admin/contacts/new(.:format)
edit_admin_contact GET      /admin/contacts/:id/edit(.:format)
admin_contact GET      /admin/contacts/:id(.:format)
admin_contact PATCH    /admin/contacts/:id(.:format)
admin_contact PUT      /admin/contacts/:id(.:format)
admin_contact DELETE   /admin/contacts/:id(.:format)
batch_action_admin_photos POST     /admin/photos/batch_action(.:format)
admin_photos GET      /admin/photos(.:format)
admin_photos POST     /admin/photos(.:format)
new_admin_photo GET      /admin/photos/new(.:format)
edit_admin_photo GET      /admin/photos/:id/edit(.:format)
admin_photo GET      /admin/photos/:id(.:format)
admin_photo PATCH    /admin/photos/:id(.:format)
admin_photo PUT      /admin/photos/:id(.:format)
admin_photo DELETE   /admin/photos/:id(.:format)
admin_dashboard GET      /admin/dashboard(.:format)
batch_action_admin_comments POST     /admin/comments/batch_action(.:format)
admin_comments GET      /admin/comments(.:format)
admin_comments POST     /admin/comments(.:format)
Controller#Action
admin/dashboard#index
admin/categories#batch_action
admin/categories#index
admin/categories#create
admin/categories#new
admin/categories#edit
admin/categories#show
admin/categories#update
admin/categories#update
admin/categories#destroy
admin/contacts#batch_action
admin/contacts#index
admin/contacts#index
admin/contacts#create
admin/contacts#new
admin/contacts#edit
admin/contacts#show
admin/contacts#update
admin/contacts#update
admin/contacts#destroy
admin/photos#batch_action
admin/photos#index
admin/photos#create
admin/photos#new
admin/photos#edit
admin/photos#show
admin/photos#update
admin/photos#update
admin/photos#destroy
admin/dashboard#index
admin/comments#batch_action
admin/comments#index
admin/comments#create

```

Figura 32 Rutas métodos servidor web service REST

Además el servidor de esta aplicación se clasifica en dos usuarios:

- Usuario registrado.
- Usuario no registrado.

Las funcionalidades principales que puede acceder el cliente REST son:

- Inicio de sesión
- Ver perfil de usuarios
- Publicar un producto
- Eliminar una publicación
- Buscar productos por categoría

- Ver publicaciones propias.
- Calcular calificaciones de otros usuarios.

### **Módulo Usuario**

El módulo para el usuario en el servidor consta de toda la administración para los clientes, pero para la aplicación móvil, el usuario debe ya estar registrado. Para poder acceder a los métodos REST las funcionalidades de este módulo son:

- Repositorio de la información del usuario.
- Validaciones del usuario.
- Relaciones del usuario con las publicaciones.

### **Módulo de publicación**

Este módulo se encarga de la gestión de las publicaciones de los usuarios. Este módulo es el repositorio de todos los artículos que el usuario publica, también tiene validaciones para los datos ingresados.

En un principio las validaciones iban a ser usadas tanto en la página web y en la aplicación móvil; pero si algún dato no pasa la validación del servidor; se mostrará un mensaje de error y se valida por objeto no por campo.

Las funcionalidades del módulo son:

- Repositorio de las publicaciones
- Repositorio de imágenes.
- Validaciones para las publicaciones
- Relaciones del usuario con los otros usuarios (comentarios)
- Envío de notificaciones

### 4.3. PRUEBAS Y VERIFICACIÓN.

El modelo Product Canvas Permite hacer pruebas unitarias constantemente mientras se va programando, por lo cual en cada iteración a cliente se le entrega un producto cada vez más elaborado y con menos fallas; por lo cual para este ejercicio se presentaran las pruebas finales.

#### 4.3.1. Pruebas de métodos para el cliente REST

Método de inicio de sesión, con la aplicación móvil:

```
Started POST "/api/sign-in?_dc=1451855464947" for 10.0.2.2 at 2016-01-03 16:10:33 -0500
Cannot render console from 10.0.2.2! Allowed networks: 127.0.0.1, ::1, 127.0.0.0/127.255.255.255
Processing by Api::SessionsController#create as JSON
  Parameters: {"email"=>"user1@demo.org", "password"=>"[FILTERED]", "_dc"=>"1451855464947"}
```

**Figura 33 Prueba 1**

Método para listar las publicaciones:

```
Started GET "/api/publications?_dc=1451855466049&page=1&start=0&limit=25" for 10.0.2.2 at 2016-01-03 16:10:34 -0500
Cannot render console from 10.0.2.2! Allowed networks: 127.0.0.1, ::1, 127.0.0.0/127.255.255.255
Processing by Api::PublicationsController#index as JSON
  Parameters: {"_dc"=>"1451855466049", "page"=>"1", "start"=>"0", "limit"=>"25"}
  User Load (0.6ms) SELECT `users`.* FROM `users` WHERE `users`.`token` = 'pPRLN6VwkwsLLPuR5XbH' LIMIT 1
  Publication Load (0.7ms) SELECT `publications`.* FROM `publications`
  User Load (0.8ms) SELECT `users`.* FROM `users` WHERE `users`.`id` IN (1, 2, 4)
```

**Figura 34 Prueba 2**



Método de cálculo de ponderación de los comentarios de los demás usuarios.

```
Started GET "/api/users/17_dc=1451855639739" for 10.0.2.2 at 2016-01-03 16:13:28 -0500
Cannot render console from 10.0.2.2! Allowed networks: 127.0.0.1, ::1, 127.0.0.0/127.255.255.255
Processing by Api::UsersController#show as JSON
Parameters: {"_dc"=>"1451855639739", "id"=>"1"}
User Load (0.9ms) SELECT `users`.* FROM `users` WHERE `users`.`token` = 'pPRLN6VwksLLPuR5XBH' LIMIT 1
User Load (0.6ms) SELECT `users`.* FROM `users` WHERE `users`.`id` = 1 LIMIT 1
(0.8ms) SELECT AVG(`comments`.`score`) FROM `comments` INNER JOIN `publications` ON `comments`.`publication_id` = `publications`.`id` WHERE `publications`.`user_id` = 1
Completed 200 OK in 71ms (Views: 13.7ms | ActiveRecord: 2.3ms)
```

Figura 35 Prueba 3

Método para obtener las categorías desde el servidor.

```
Started GET "/api/categories?_dc=1451855776043&page=1&start=0&limit=25" for 10.0.2.2 at 2016-01-03 16:15:44 -0500
Cannot render console from 10.0.2.2! Allowed networks: 127.0.0.1, ::1, 127.0.0.0/127.255.255.255
Processing by Api::CategoriesController#index as JSON
Parameters: {"_dc"=>"1451855776043", "page"=>"1", "start"=>"0", "limit"=>"25"}
User Load (1.2ms) SELECT `users`.* FROM `users` WHERE `users`.`token` = 'pPRLN6VwksLLPuR5XBH' LIMIT 1
Category Load (5.8ms) SELECT `categories`.* FROM `categories` ORDER BY `categories`.`name` ASC
Completed 200 OK in 311ms (Views: 247.1ms | ActiveRecord: 7.0ms)
```

Figura 36 Prueba 4

Método para cargar las imágenes desde el cliente REST.

```
Started POST "/api/publications/17/photos" for 10.0.2.2 at 2016-01-03 16:19:57 -0500
Cannot render console from 10.0.2.2! Allowed networks: 127.0.0.1, ::1, 127.0.0.0/127.255.255.255
Processing by Api::PhotosController#create as JSON
Parameters: {"file"=>"ActionDispatch::Http::UploadedFile:0xb5939ea4 @tempfile=#<Tempfile:/tmp/RackMultipart20160103-3357-1dxjlc.jpg>, @original_filename="1451856013913.jpg", @content_type="image/jpeg", @headers="Content-Disposition: form-data; name=\"file\"; filename=\"1451856013913.jpg\""}
User Load (0.4ms) SELECT `users`.* FROM `users` WHERE `users`.`token` = 'pPRLN6VwksLLPuR5XBH' LIMIT 1
Publication Load (0.9ms) SELECT `publications`.* FROM `publications` WHERE `publications`.`id` = 17 LIMIT 1
(0.3ms) BEGIN
SQL (4.4ms) INSERT INTO `photos` (`file`, `publication_id`, `created_at`, `updated_at`) VALUES ('1451856013913.jpg', 17, '2016-01-03 21:19:57')
(54.7ms) COMMIT
Completed 200 OK in 178ms (Views: 37.1ms | ActiveRecord: 60.8ms)

Started POST "/api/publications/17/photos" for 10.0.2.2 at 2016-01-03 16:19:57 -0500
Cannot render console from 10.0.2.2! Allowed networks: 127.0.0.1, ::1, 127.0.0.0/127.255.255.255
Processing by Api::PhotosController#create as JSON
Parameters: {"file"=>"ActionDispatch::Http::UploadedFile:0xb4ddf428 @tempfile=#<Tempfile:/tmp/RackMultipart20160103-3357-5diryp.jpg>, @original_filename="1451855971052.jpg", @content_type="image/jpeg", @headers="Content-Disposition: form-data; name=\"file\"; filename=\"1451855971052.jpg\""}
User Load (4.6ms) SELECT `users`.* FROM `users` WHERE `users`.`token` = 'pPRLN6VwksLLPuR5XBH' LIMIT 1
Publication Load (6.9ms) SELECT `publications`.* FROM `publications` WHERE `publications`.`id` = 17 LIMIT 1
(9.0ms) BEGIN
SQL (13.7ms) INSERT INTO `photos` (`file`, `publication_id`, `created_at`, `updated_at`) VALUES ('1451855971052.jpg', 17, '2016-01-03 21:19:57')
(10.6ms) COMMIT
Completed 200 OK in 154ms (Views: 54.2ms | ActiveRecord: 44.3ms)
```

Figura 37 Prueba 5

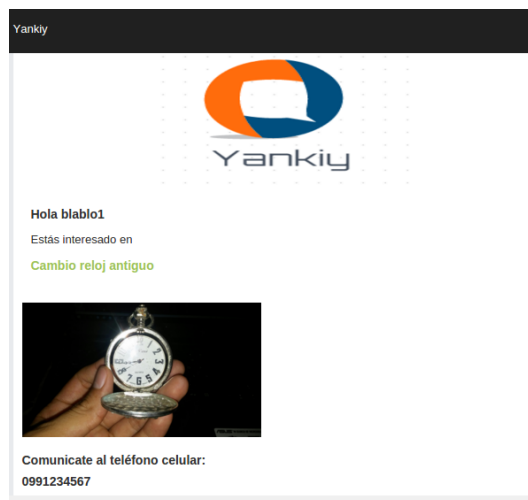
Método de envío de notificaciones. Resultado de servidor al momento de renderizar el código para enviarlo al destinatario.

```
Has recibido este email porque=C3=A9 e=
stas suscrito a www.yankiy.com.ec<br> =
</td>
</tr>
<!-- Spacing -->
<tr>
<td width=3D"100%" height=3D"20"></td>
</tr>
<!-- Spacing -->
</tbody>
</table>
</td>
</tr>
</tbody>
</table>
</td>
</tr>
</tbody>
</table>
</body>

</body>
</html>

----=_mimepart_568a97c09eb46_b1226c6647a6522d--
Completed 200 OK in 1706ms (Views: 163.3ms | ActiveRecord: 37.4ms)
```

**Figura 38 Prueba 6**



**Figura 39 Receptor mail interesado en publicación**



**Figura 40 Receptor mail dueño de publicación.**

#### 4.3.2. Pruebas interfaces

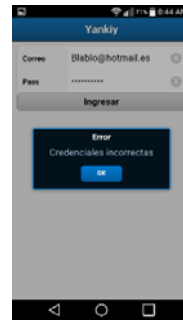
Pantalla de inicio:



**Figura 41 Prueba 1**

Inicio sesión con datos erróneos en los campos de correo y password:

**Continua**



**Figura 42 Prueba 2**

Inicio sesión con datos correctos:



**Figura 43 Prueba 4**

Búsqueda por categorías, al seleccionar todas las categorías se mostrará la siguiente pantalla:



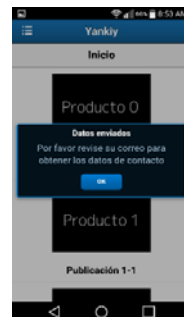
**Figura 44 Prueba 5**

Al elegir el producto deseado se puede contactar con el usuario, este botón solo se mostrará una única vez; después de haberlo contacto el botón se eliminará.



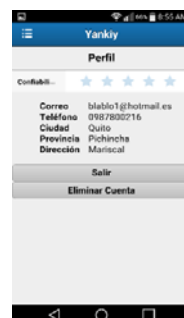
**Figura 45 Prueba 5**

Una vez hecha la acción, se mostrará un mensaje de confirmación.



**Figura 46 Prueba 6**

Vista de perfil del usuario sin calificación o comentario



**Figura 47 Prueba 7**

Vista de perfil del usuario con calificación o comentario



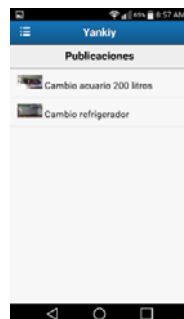
**Figura 48 Prueba 8**

Publicación de un producto; se debe de llenar los datos solicitados en pantalla. Se puede tomar fotos en ese instante o seleccionar una foto de la galería.



**Figura 49 Prueba 9**

Lista de publicaciones de usuario:



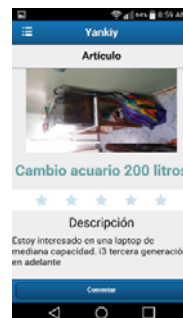
**Figura 50 Prueba 10**

Eliminar un artículo;



**Figura 51 Prueba 11**

Comentar a un usuario; si ya se ha contactado con el usuario del artículo se mostrará el botón de comentar.



**Figura 52 Prueba 12**

Aparecerá una nueva pantalla con datos a llenar.



**Figura 53 Prueba 13**



**Figura 54** Prueba 14

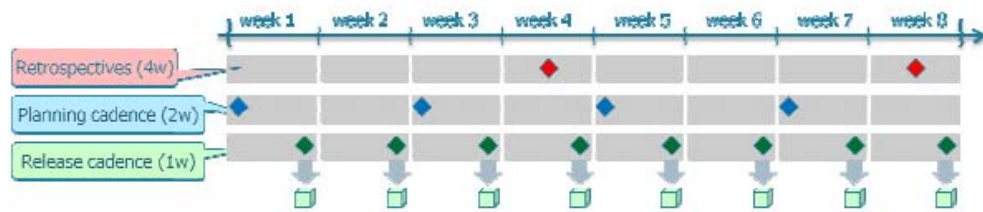


## 5. CAPITULO V

### 5.1. RESULTADOS

- Utilizar un servidor REST hecho en Ruby on Rails, es una solución fácil e integral para los requerimientos de gestión y seguridad. Todos los métodos que se realizan en Ruby son convertidos automáticamente a un Web Service REST, este principio se base sobre la construcción de las rutas de cada método. Todas las rutas en Ruby usan las peticiones PUT, POST, GET, DELETE. Por lo cual no se necesita de ninguna dependencia o librería para generar el servidor.
- El cliente REST, no necesita librerías ni tiene una dependencia para consumir el servicio. Estos lenguajes son muy independientes además que consumen pocos recursos; optimizando el rendimiento. Sencha Touch con Cordova simplifica mucho la dependencia de cualquier librería para poder usar los recursos del dispositivo móvil. Cordova simula el código como si fuera nativo de la plataforma por lo cual, se puede usar la cámara y las interfaces propias de android para hacer cualquier tarea. El patrón de diseño simplifica la implementación y el fácil entendimiento del desarrollo, así como la integración con metodologías ágiles de desarrollo para hacer cada vez mucho más cortas las iteraciones y cualquier respuesta ante un inconveniente.
- Pocas son las empresas y personas que usan un tablero para resumir las actividades, recursos o diseño al momento de desarrollar software. Para dar a conocer el potencial de la herramienta se desarrolló un breve análisis entre Product Canvas con SCRUM al momento de interactuar con el cliente:

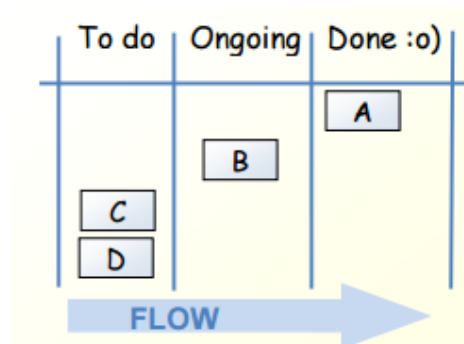
Scrum usa tarjetas por tareas y se le asigna a cada persona para realizarlas en cada iteración.



**Figura 55 Calendario Scrum**  
(Kniberg, 2016)

En Scrum por lo general se planea una iteración cada semana, todo el grupo se reúne para dividir las tareas, el tiempo de iteración se reducirá dependiendo la velocidad del grupo y cada iteración deberá esperar a que las tareas estén hechas para poder empezar una nueva iteración. Y por lo general el líder siempre tiene que estar pendiente que cada persona sea repartida una misma cantidad de tareas para que dure todo el tiempo de iteración.

En Product Canvas, se divide por flujos es decir se sabe lo que se debe hacer y conforme cada flujo se asina recursos:



**Figura 56 Flujo Product Canvas**  
(Kniberg, 2016)

El tiempo de cada iteración dependerá de la velocidad de cada recurso, es decir se limita por el estado del trabajo. Las tareas se asignan tanto al

cliente como al desarrollador ya que la tarea no estará completa hasta que el cliente cambie el estado de la tarea a concluido.

Esta acción de estar dividir al software por estado de trabajo permite a que el desarrollador tenga mucho contacto con el cliente, además que hace que el cliente sea participe del desarrollo. Lo clientes saben y pueden hacer preguntas sobre el desarrollo. Incluso si algo no le gusta al cliente en la nueva retroalimentación se lo cambia enseguida.

La diferencia entre Scrum es que a medida que pasa cada iteración el código se agranda por lo cual si sucede algo en el desarrollo a medida que pase el tiempo será más largo de revisar.

En Product Canvas el objetivo es reducir responsabilidades y carga de trabajo. Por lo cual el tiempo de cada iteración de flujo de trabajo dependerá de que tan rápido puedo empezar una nueva iteración.

- Ruby con todas sus gemas ayuda a identificar una línea de desarrollo para reutilizar código y funciones existentes, como por ejemplo Active Record, que ayuda a generar CRUD a todas las tablas de la base de datos fácilmente. Otra gema muy útil fue Devise que ayuda a generar la gestión de los usuarios, esta gema además de gestionar los usuarios tiene métodos para la seguridad del servidor y el acceso a peticiones no deseadas. Cabe mencionar la gema CarrierWave, automatiza la administración de las imágenes, permitiendo recibir los archivos y guardando en la base de datos la ruta que se guardó en el servidor; por lo que la base de datos no se llena de bytes, volviéndola muy liviana y menorando el tiempo de repuesta.
- Todas las herramientas usadas para el desarrollo de la tesis, son de fácil uso e innovadoras la mayoría de ellas son gratuitas excepto por NingaMock que solo permite tener un proyecto gratuito.

Producto Canvas ayuda a tener una vista global del producto software además que permite plasmar en términos comunes y de fácil

entendimiento lo que quiere el cliente. También ayuda a que el cliente sea parte del desarrollo ayudándole a tener mejores ideas y permitiendo que la documentación sea compartida a muchas personas sin depender de conocimientos de desarrollo software.

Ruby on Rail es un lenguaje muy flexible que rompe con todos los estándares de desarrollo, permitiendo su fácil interpretación y aprendizaje. Ayuda a que los servidores sean más livianos y cuenta con una gran gama de recursos para facilitar el desarrollo. Ayuda a presentar software en menos tiempo con interfaces muy elaboradas.

## 5.2. CONCLUSIONES

- Debido a la amplia aceptación que han tenido las herramientas de programación Ruby y JavaScript, con sus frameworks Rails y Sencha Touch respectivamente, y después de la investigación que hemos realizado podemos decir que estas en conjunto son capaces de proporcionar una arquitectura bastante robusta y evolutiva.
- La mayor fortaleza del entorno de trabajo con lenguajes interpretados, es la forma en que se aprovecha la Convención sobre Configuración; es decir, que siempre se tendrá una homogeneidad de desarrollo en los proyectos y casi nunca se tendrá código ambiguo. Este principio permite enfoques intuitivos, por ejemplo: cambios en el código se vuelve sencillo debido al aislamiento o centralización de ciertos componentes (gemas) ya que el entorno se basa en lugares predeterminados, donde se debe ubicar cierta funcionalidad. Por lo que si se reutiliza código, este no afectará casi en nada al modelo del negocio que se está desarrollando
- Sencha Touch CMD, proporciona una gran ventaja al momento de querer migrar el código a otra plataforma, posee una gran cantidad de controles IU para desarrollar pantallas bastante complejas e interactivas. Ha sido diseñado específicamente para dispositivos táctiles por lo que incluye una amplia

gama de eventos táctiles. Sencha CMD a pesar de ser una plataforma reciente posee la suficiente documentación, soporte, APIs, librerías, entre otros; que permiten una desarrollar una plataforma robusta basada en el lenguaje javascript.

- El Product Canvas permite hacer partícipe al cliente en cualquier momento del desarrollo, maneja un lenguaje simple y común por lo que el cliente no necesariamente debe estar siempre presente físicamente en las reuniones de trabajo; este modelo le permite al cliente estar lejos de la organización y tener una comunicación lineal, ya que todos están hablando el mismo lenguaje.
- Al reutilizar código de terceros, se garantiza que el desarrollo de los proyectos se lo implemente en menor tiempo y con más facilidad. La filosofía de los lenguajes interpretados es generar la menor carga posible en el desarrollo, por otro lado al ser de software libre se puede modificar todos los recursos para que realicen alguna tarea en específico. También al reutilizar código se fomenta la investigación en el desarrollador ya que el implementar código de terceros tiene su complejidad y encontrar justo el código que se necesita conlleva una responsabilidad.

### **5.3. RECOMENDACIONES**

- Realizar capacitación oportuna sobre el desarrollo en Ruby on Rail y Java Script, si bien en internet hay mucha información sobre las herramientas; esta información no siempre es orientada a lo que se desea realizar por lo cual se necesita una guía idónea para empezar con estos lenguajes.
- Implementar mecanismo de seguridad para la conexión entre cliente y servidor. Ruby ayuda a tener la información cifrada y tiene muchas gemas que ayudan a bloquear el acceso a personas no deseadas.

- Realizar aplicaciones móviles que no tengan ninguna información respecto a clientes o datos del servidor, la aplicación móvil deberá obtenerla toda la información desde el servidor.
- Trabajar con objetos JSON es una forma muy viable para optimizar el consumo de recursos ya que no ocupan mucha memoria y los objetos pueden ser renderizados fácilmente.
- Para empezar con el estudio de estos lenguajes se debe seleccionar de forma adecuado las fuentes de información, en el desarrollo de la tesis se puede revisar los ejemplos con los códigos, sin embargo en la web se pueden encontrar muchos tutoriales y ejemplos cada vez más elaborados.
- Rails hereda algunas de las debilidades del lenguaje utilizado para su construcción. Más específicamente, la falta de apoyo de concurrencia; obstaculiza de cierta forma el potencial de escalabilidad de una aplicación construida usando Ruby on Rails. Además, la gestión de memoria no es tan eficiente como otros idiomas disponibles. Por ejemplo, Java exhibe un mejor comportamiento al momento de recolectar basura en comparación con Ruby on Rails. Por lo que, en caso de tener un sistema bastante robusto; se deberá incluir procesos de optimación en el rendimiento.

## BIBLIOGRAFÍA

- A., B. (7 de 10 de 2015). *Ruby for Newbies: Working whit gems*. Obtenido de <http://code.tutsplus.com/articles/ruby-for-newbies-working-with-gems--net-18977>.
- A., C. (1 de 7 de 2015). Obtenido de [http://www.colegiosansaturio.com/deptomatesweb/4ESO/informatica%20web/temas/Unidad\\_6/pagina1.html](http://www.colegiosansaturio.com/deptomatesweb/4ESO/informatica%20web/temas/Unidad_6/pagina1.html)
- Arkaitzgarro. (7 de 12 de 2015). Obtenido de Arkaitzgarro: <http://www.arkaitzgarro.com/phonegap/capitulo-2.html>
- Báez M, B. A. (2015 de 6 de 1). *Introducción a Android*. Obtenido de <http://pendientedemigracion.ucm.es/info/tecnomovil/documentos/android.pdf>
- ClickD Sitio. (15 de 1 de 2015). *Promoviendo el trueque, una contribución especial al medio ambiente*. Obtenido de [http://www.clickafectatumundo.com/index.php?option=com\\_content&view=article&id=305:promoviendo-el-trueque-una-contribucion-especial-al-medio-ambiente&catid=50:nuestra-naturaleza&Itemid=84](http://www.clickafectatumundo.com/index.php?option=com_content&view=article&id=305:promoviendo-el-trueque-una-contribucion-especial-al-medio-ambiente&catid=50:nuestra-naturaleza&Itemid=84)
- code, E. (30 de 11 de 2015). *Espocode Tech*. Obtenido de ExpoCode: <http://expocodetech.com/expo-tutorial-sencha-touch-2-x-26/>
- D, G. (9 de 2 de 2015). Obtenido de Ruby Fácil: [https://s3.amazonaws.com/akitaonrails/files/RubyFacil\\_071105.pdf](https://s3.amazonaws.com/akitaonrails/files/RubyFacil_071105.pdf)
- D, H. (10 de 10 de 2015). *Active Record Basic*. Obtenido de [http://guides.rubyonrails.org/active\\_record\\_basics.html](http://guides.rubyonrails.org/active_record_basics.html)
- David, M. (2014). *Using Sencha Touch to Build Mobile Website*. Waltham: Focal Press.

- DesarrolloWeb. (9 de 2 de 2015). *Javascript a fondo*. Obtenido de <http://www.desarrolloweb.com/javascript/#quees>
- E., G. (9 de 2 de 2015). *¿Qué es y para qué sirve Javascript?* Obtenido de [http://www.aprenderaprogramar.com/index.php?option=com\\_content&id=590:iue-es-y-para-que-sirve-javascript-embeber-javascript-en-html-ejercicio-ejemplo-basico-cu00731b&Itemid=192](http://www.aprenderaprogramar.com/index.php?option=com_content&id=590:iue-es-y-para-que-sirve-javascript-embeber-javascript-en-html-ejercicio-ejemplo-basico-cu00731b&Itemid=192)
- EdenorOficial. (6 de 12 de 2015). *Taringa*. Obtenido de <http://www.taringa.net/posts/imagenes/18250925/El-curioso-origen-de-la-palabra-salario.html>
- Facebook. (13 de 12 de 2015). *Facebook*. Obtenido de <https://www.facebook.com/groups/1402528743335820/?fref=ts>
- G, P. (9 de 2 de 2015). *Sistemas Operativos en Dispositivos Móviles*. Obtenido de [http://exa.unne.edu.ar/informatica/SO/Sistemas\\_Operativos\\_en\\_Dispositivos\\_Moviles.pdf](http://exa.unne.edu.ar/informatica/SO/Sistemas_Operativos_en_Dispositivos_Moviles.pdf)
- Galeano, D. (5 de 3 de 2015). *Aplicaciones moviles de la mano de PhoneGap*. Obtenido de <http://www.desarrolloweb.com/manuales/aplicaciones-moviles-phonegap.html>
- González M, M. M. (2016 de 6 de 1). *El trueque y la moneda, Origen, Recuperado* . Obtenido de <http://www.portalplanetasedna.com.ar/trueque.htm>
- González M, M. M. (s.f.). *El trueque y la moneda, Origen*. Recuperado el 15 de 1 de 2015, de <http://www.portalplanetasedna.com.ar/trueque.htm>
- Heinemeier, D. (10 de 6 de 2015). *Rails Guides*. Obtenido de [http://guides.rubyonrails.org/getting\\_started.html](http://guides.rubyonrails.org/getting_started.html)
- JavaTechig*. (14 de 12 de 2015). Obtenido de <http://javatechig.com/sencha-touch/sencha-touch-user-interface-development>
- Json. (8 de 12 de 2015). Obtenido de <http://www.json.org/json-es.html>



- Kniberg, H. (4 de 1 de 2016). *Kanban vs Scrum* . Obtenido de <https://www.crisp.se/file-uploads/Kanban-vs-Scrum.pdf>
- Librosweb. (6 de 12 de 2015). *Librosweb*. Obtenido de ¿Qué es JavaScript? : [http://librosweb.es/libro/javascript/capitulo\\_1.html](http://librosweb.es/libro/javascript/capitulo_1.html)
- M, R. (2009). *Lenguajes De Programación*. Recuperado el 2 de 9 de 2015, de <http://altenwald.org/2009/02/02/lenguajes-de-programacion/>
- M., R. (9 de 2 de 2015). *Los 5 mejores Sistemas operativos para celulares*. Obtenido de <http://iphoneandord.com/los-5-mejores-sistemas-operativos-para-celulares/>
- Mariño, M. (15 de 1 de 2015). *Trueque social en tiempos de crisis*. Obtenido de <http://blogs.20minutos.es/140-y-mas/2012/06/28/trueque-social-en-tiempos-de-crisis/>
- Mariño, M. (15 de 1 de 2015). *Trueque social en tiempos de crisis*. Obtenido de <http://blogs.20minutos.es/140-y-mas/2012/06/28/trueque-social-en-tiempos-de-crisis/>
- Mario, B. (13 de 12 de 2015). *GitHub*. Obtenido de <https://github.com/brianmario/mysql2>
- Matz. (29 de 11 de 2001). en An Interview with the Creator of Ruby. Obtenido de en An Interview with the Creator of Ruby
- Matz, e. A. (2001). Interview with the Creator of Ruby. *TecnoMagazine*, 29.
- Mejia, A. (7 de 12 de 2015). *Adrian Mejia's blog*. Obtenido de <http://adrianmejia.com/blog/2011/08/11/ruby-on-rails-architectural-design/>
- Mint. (6 de 12 de 2015). *Mint*. Obtenido de <https://www.mint.com/barter-system-history-the-past-and-present>

- Orenfarhi. (6 de 12 de 2015). *mobiForge*. Obtenido de <https://mobiforge.com/news-comment/what-does-javascript-bring-mobile-html5-and-css3-party>
- PhoneGap. (19 de 8 de 2015). *PhoneGap Documentation*. Obtenido de <http://docs.phonegap.com/>
- Pichler, P. (s.f.). *Pichler Consulting*. Recuperado el 19 de 08 de 2015, de <http://www.romanpichler.com/tools/product-canvas/>
- Pichler, R. (18 de 8 de 2015). *The Product Canvas*. Obtenido de <http://www.romanpichler.com/tools/product-canvas/>
- Plataformatec. (13 de 12 de 2015). *GitHub*. Obtenido de <https://github.com/plataformatec/devise>
- Rubenfa. (6 de 10 de 2015). *Patrones de diseño: Qué son y por qué debes usarlos*. . Obtenido de <http://www.genbetadev.com/metodologias-de-programacion/patrones-de-diseno-que-son-y-por-que-debes-usarlos>.
- Ruby. (1 de 9 de 2015). *Acerca de Ruby*. Obtenido de <https://www.ruby-lang.org/es/about/>
- Ruby on Rails. (6 de 12 de 2015). Obtenido de <http://api.rubyonrails.org/classes/ActionController/Base.html>
- Rubydoc. (14 de 10 de 2015). Obtenido de <http://www.rubydoc.info/docs/rails/3.0.0/ActiveRecord/Base>
- Rubyonrails. (20 de 10 de 2015). *Action controller*. Obtenido de [http://guides.rubyonrails.org/action\\_controller\\_overview.html](http://guides.rubyonrails.org/action_controller_overview.html)
- Rubyonrailsthissur. (14 de 10 de 2015). *Ruby on Rails Tips*. Obtenido de <https://rubyonrailsthissur.wordpress.com/2012/07/18/naming-conventions-while-creating-tables-in-rails/> 14-10-2015

Sencha Touch. (7 de 12 de 2015). Obtenido de

<https://www.sencha.com/products/touch/#overview>

Sim, P. (10 de 10 de 2015). *Active Record - Object relational mapping in Rails*.

Obtenido de <https://github.com/rails/rails/tree/master/activerecord>

Thomasdbs. (13 de 12 de 2015). *GitHub*. Obtenido de

<https://github.com/carrierwaveuploader/carrierwave>

tutorialspoint. (6 de 10 de 2015). *Ruby on Rails - Frameworks*. Obtenido de

<http://www.tutorialspoint.com/ruby-on-rails/rails-framework.htm>

## **ANEXOS**