



UNIVERSIDAD CENTRAL DEL ECUADOR

FACULTAD DE INGENIERÍA, CIENCIAS FÍSICAS Y MATEMÁTICA

CARRERA DE INGENIERÍA INFORMÁTICA

**“DESARROLLO DE UNA API REST CON SUS APLICACIONES WEB Y
MÓVIL PARA LA VENTA DE ROPA ONLINE DE LA EMPRESA ROOSMAN”**

TRABAJO DE GRADUACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE
INGENIERO INFORMÁTICO

AUTORES:

SANTOS HERNÁNDEZ WAGNER DAVID

SERRANO PARREÑO JAIRO ADRIÁN

TUTOR: ING. DARWIN RODOLFO CAINA AYSABUCHA, MSc

QUITO-13 DE MARZO

2017

AUTORIZACIÓN DE LA AUTORÍA INTELECTUAL

Nosotros, WAGNER DAVID SANTOS HERNANDEZ Y JAIRO ADRIÁN SERRANO PARREÑO en calidad de autores del trabajo de titulación, modalidad proyecto integrador: DESARROLLO DE UNA API REST CON SUS APLICACIONES WEB Y MÓVIL PARA LA VENTA DE ROPA ONLINE DE LA EMPRESA ROOSMAN, autorizamos a la Universidad Central del Ecuador hacer uso de todos los contenidos que nos pertenecen o parte de los que contiene esta obra, con fines estrictamente académicos o de investigación

Los derechos que como autores nos corresponden, con excepción de la presente autorización, seguirán vigentes a nuestro favor, de conformidad con lo establecido en los artículos 5, 6, 8, 19 y además pertinentes de la Ley de Propiedad Intelectual y su Reglamento.

Así mismo autorizamos a la Universidad Central del Ecuador para que realice la digitalización y publicación de este trabajo de investigación en el repositorio virtual, de conformidad a lo dispuesto en el Art. 144 de la Ley orgánica de Educación Superior.

En la ciudad de Quito, 13 de marzo del 2017.



David Wagner Santos Hernandez

CC.1717123929

Teléfono: 025109164

Celular: 0996800629

wagnersants1@hotmailmail.com



Jairo Adrián Serrano Parreño

CC. 1724249493

Teléfono: 022851240

Celular: 0987726931

jairoserrano89@hotmail.com

CERTIFICACIÓN DEL TUTOR

Yo, Darwin Rodolfo Caina Aysabucha en calidad de tutor del trabajo de titulación: DESARROLLO DE UNA API REST CON SUS APLICACIONES WEB Y MÓVIL PARA VENTA DE ROPA ONLINE DE LA EMPRESA ROOSMAN, elaborado por los estudiantes David Wagner Santos Hernández y Jairo Adrián Serrano Parreño de la Carrera de Informática, Facultad de Ingeniería Ciencias Físicas y Matemática de la Universidad Central del Ecuador, considero que los mismos reúnen los requisitos y méritos necesarios en el campo metodológico y en el campo epistemológico, para ser sometidos a la evaluación por parte del jurado examinador que se designe, por lo que lo APRUEBO, a fin de que el proyecto integrador sea habilitado para continuar con el proceso de titulación determinado por la Universidad Central del Ecuador.

Certifico que este documento fue revisado en el programa antiplagio URKUND.

En la ciudad de Quito, a los 20 días del mes de febrero de 2017.



Ing. Darwin Rodolfo Caina Aysabucha, MSc

CC: 1803221389

Teléfono: 0984551699

Correo electrónico: drcaina@uce.edu.ec

Urkund Analysis Result

Analysed Document: TESIS_10022017.docx (D25611867)
Submitted: 2017-02-10 07:15:00
Submitted By: jairoserrano89@hotmail.com
Significance: 1 %

Sources included in the report:

Titulacion-II-04-01-2017-11-49-WILMER ALBERTO ZAMBRANO GARCIA.docx (D24710842)
<http://www.desarrolloweb.com/articulos/que-es-angularjs-descripcion-framework-javascript-conceptos.html>

Instances where selected sources appear:

2

INFORME DE LOS REVISORES



UNIVERSIDAD CENTRAL DEL ECUADOR
FACULTAD DE INGENIERÍA, CIENCIAS FÍSICAS Y MATEMÁTICA
DIRECCIÓN CARRERA DE INGENIERÍA INFORMÁTICA

Quito, 01 de marzo de 2017.
Oficio 051-2017-DC.

Señores:

Ing. Jorge Morales

Ing. Wagner Lucero

DOCENTES REVISORES DE PROYECTO

Presente.-

Estimados Docentes:

Una vez que está culminado el proyecto INTEGRADOR titulado "**Desarrollo de una API REST con sus aplicaciones Web y móvil para la Venta de Ropa Online de la empresa Roosman**", elaborado por el (los) postulante (s) SANTOS HERNÁNDEZ DAVID WAGNER y SERRANO PARREÑO JAIRO ADRIAN, me permito solicitar de manera comedida **procedan a evaluar el proyecto** señalado de acuerdo al formato remitido, en un plazo máximo de 10 días y entregar en la Dirección de la Carrera para continuar con el trámite respectivo.

Agradezco su atención.

Atentamente,

Ing. Boris Herrera Flores., MSc.

**DIRECTOR DE LA CARRERA DE INGENIERÍA
EN INFORMÁTICA.**



Anexo: formato para evaluación.
Paola Burbano.

CALIFICACIÓN DE TRIBUNAL



UNIVERSIDAD CENTRAL DEL ECUADOR
FACULTAD DE INGENIERÍA, CIENCIAS FÍSICAS Y MATEMÁTICA
DIRECCIÓN CARRERA DE INGENIERÍA INFORMÁTICA

UNIDAD DE TITULACIÓN

RESULTADO DEL PROYECTO INTEGRADOR

CARRERA DE: INGENIERÍA INFORMÁTICA

FECHA: 10 de marzo de 2017.

POSTULANTE (S): SANTOS HERNÁNDEZ DAVID WAGNER y SERRANO PARREÑO JAIRO ADRIAN

PROYECTO: "Desarrollo de una API REST con sus aplicaciones Web y móvil para la Venta de Ropa Online de la empresa Roosman".

CALIFICACIÓN:

TRIBUNAL	PROFESOR (A)	NOTA SOBRE CUARENTA	
		NUMEROS	LETRAS
REVISOR	Ing. Jorge Morales	39,00	Treinta y Nueve Coma Cero
REVISOR	Ing. Wagner Lucero	39,00	Treinta y Nueve Coma Cero
PROMEDIO		39,00	Treinta y Nueve Coma Cero
Ing. Boris Herrera MSc. DIRECTOR DE LA CARRERA			

Dr. Mario Guerra Burbano
SECRETARIO ABOGADO.



Elaborado por: Paola Burbano.

Teléfono: (02) 2558-833 ext. 218 / 219

DEDICATORIA

A mis Padres Rosa y Manuel por su

apoyo incondicional, comprensión y amor

A mi hermano Efraín y su esposa Paola, por sus consejos,
palabras de aliento y motivación constante que me han brindado

A la vida por darme la oportunidad de seguir mis sueños y tener

la maravillosa familia y amigos que tengo.

Wagner

Primero a Dios,

por levantarme cada día y ayudarme a cumplir esta meta.

A mis padres y hermanos que han sido mi apoyo,

y han estado ahí para mí cuando lo necesitaba.

A mi novia que ha compartido conmigo

cada tristeza y alegría en este camino.

Y a mis amigos que me apoyaron y con los que festejamos

los logros de cada uno como si fuesen propios.

Jairo

AGRADECIMIENTOS

Los autores expresan sus agradecimientos a:

Nuestro tutor, Ing. Darwin Caina, por sus valiosas orientaciones. Sus conocimientos, su manera de trabajar, su persistencia, su paciencia y su motivación hacia este proyecto, ya que sin su ayuda no se habría podido lograr.

A nuestros amigos y familiares por el apoyo brindado.

A la empresa ROOSMAN por su participación e inclusión en el desarrollo del proyecto.

CONTENIDO

AUTORIZACIÓN DE AUTORÍA INTELECTUAL	II
CERTIFICACIÓN DEL TUTOR	III
DESIGNACIÓN DE TRIBUNAL	V
CALIFICACIÓN DE TRIBUNAL	VI
INTRODUCCIÓN	1
MARCO TEÓRICO	3
1.1. REST	3
1.1.1. Uso correcto de URIs	3
1.1.2. Uso correcto de HTTP.....	4
1.1.3. Implementar <i>Hypermedia</i>	5
1.2. Aplicaciones SPA o páginas Web SPA (<i>Single Page Applications</i>).....	6
1.2.1. Arquitectura básica de una SPA	6
1.3. Lumen <i>micro-framework</i> de Laravel.....	8
1.4. AngularJS	9
1.5. IONIC.....	9
1.6. Bases de datos No Sql	10
1.6.1. Ventajas de los sistemas NoSQL.....	11
1.6.2. Bases de datos documentales.....	12
1.7. Levantamiento de Requerimientos con la Técnica JAD (<i>Joint Application Design</i>)	13
1.7.1. Desarrollo de taller JAD.....	13
1.8. Metodologías Ágiles	15
1.9. SCRUM.....	16
1.9.1. El Equipo Scrum (<i>Scrum Team</i>).....	17
1.9.2. El Dueño de Producto (<i>Product Owner</i>)	17
1.9.3. El Equipo de Desarrollo (<i>Development Team</i>).....	18
1.9.4. El <i>Scrum Master</i>	19
1.9.5. Eventos de <i>Scrum</i>	19
1.9.6. El <i>Sprint</i>	20
METODOLOGÍA	21
1.10. Requerimientos y Metodología de desarrollo.....	21
1.11. Selección de herramientas	25
DESARROLLO	28
1.11.1. Capas del sistema	28
1.12. Desarrollo del API REST	28
1.12.1. Configuración de entorno.....	30
1.12.2. <i>Scaffolding</i> del Proyecto.....	31

1.12.3.	Instalación de JWT	34
1.12.4.	Diseño de la Base de Datos	34
1.12.5.	Instalación de drivers para conexión con MongoDB	40
1.12.6.	Creación de CRUD principales	40
1.13.	Descripción del proceso de compra.....	41
1.13.1.	Tareas programadas (<i>cron</i>).....	44
1.14.	Desarrollo de la Aplicación Web.....	45
1.14.1.	Diseño de los <i>Wireframes</i>	46
1.14.2.	<i>Scaffolding</i> del Proyecto Web	46
1.14.3.	Instalación de los diferentes <i>plugins</i> para usar en el proyecto.....	49
1.14.4.	Creando las rutas que se conectarán al API.....	50
1.14.5.	Realización de la pantalla de inicio	52
1.14.6.	Creación del resto de pantallas basado en los <i>wireframes</i>	55
1.15.	Desarrollo de la aplicación móvil.....	55
1.15.1.	Diseño de los <i>wireframe</i> de la aplicación móvil.....	56
1.15.2.	<i>Scaffolding</i> del Proyecto Web	57
1.15.3.	Instalación de los diferentes <i>plugins</i> para usar en el proyecto móvil	59
1.15.4.	Creación de las rutas que se conectarán al API.....	60
1.15.5.	Creación de la pantalla de inicio para la aplicación móvil	62
1.15.6.	Creación del resto de pantallas basado en los <i>wireframes</i>	66
RESULTADOS.....		67
1.16.	Perfil administrador.....	67
1.16.1.	Manejo de catálogos.....	67
1.16.2.	Reporte de facturas	70
1.16.3.	Configuración general	71
1.17.	Perfil Cliente.....	72
1.17.1.	Navegación y selección de productos.....	72
1.17.2.	Carrito de compras	76
1.17.3.	Correo de compra	77
1.17.4.	Historial de Pagos.....	77
CONCLUSIONES		79
RECOMENDACIONES		80
GLOSARIO.....		81
BIBLIOGRAFÍA.....		86
ANEXOS.....		87

LISTAS DE TABLAS

Tabla 1.1. Ventajas y Desventajas de las SPAs	7
Tabla 1.2. Diferencias entre metodologías ágiles y no ágiles.....	16
Tabla 3.1 Campos del documento usuario.....	36
Tabla 3.2 Campos documento Setup	36
Tabla 3.3 Campos documento categoría.....	36
Tabla 3.4 Campos del documento secciones	37
Tabla 3.5 Campos del documento de impuestos.....	37
Tabla 3.6 Campos del documento productos.....	38
Tabla 3.7 Campos del documento carrito_compras	39
Tabla 3.8 Campos del documento codigo_descuento	39
Tabla 3.9 Campos del documento de facturas	40
Tabla 3.10 Campo del documento detalle_factura.....	40

LISTA DE FIGURAS

Figura 1.1. Ciclo de vida de las aplicaciones Web tradicionales y SPA	8
Figura 1.2. Bases de datos No SQL de documentos	12
Figura 2.1 Jerarquía para manejo de productos	22
Figura 2.2 Código fuente de API en Bitbucket.....	27
Figura 3.1 Gráfica de las capas del sistema	28
Figura 3.2. Dashboard de Trello	29
Figura 3.3 Tareas del Sprint.....	29
Figura 3.4. Archivo de configuración Homestead.yaml.....	31
Figura 3.5. Scaffolding del Proyecto	32
Figura 3.6. Diagrama de base de datos.....	35
Figura 3.7. Proceso general de compra	42
Figura 3.8. Proceso Agregar productos al carrito de compra.	43
Figura 3.9. Wireframes de la pantalla home	46
Figura 3.10. Scaffolding del Proyecto Web.....	47
Figura 3.11. Archivo values.js	50
Figura 3.12. Archivo methods.js	52
Figura 3.13. Archivo home.html, que será usado para la pantalla de inicio.....	54
Figura 3.14. Archivo states.js	54
Figura 3.15. home.js	55
Figura 3.16. Wireframe de la pantalla de login.....	57
Figura 3.17. Scaffolding del proyecto móvil	58
Figura 3.18. Archivo values.js con la variable global apiUrl.	60
Figura 3.19. Archivo methods.js	61
Figura 3.20. login.html	62
Figura 3.21. states.js	63
Figura 3.22. login.js.....	66
Figura 4.1. Dashboard de Administrador	67
Figura 4.2. Lista de Categorías	68
Figura 4.3 Lista de Secciones	68
Figura 4.4. Lista de Impuestos	69
Figura 4.5. Lista de Productos	69
Figura 4.6. Lista de Códigos de Descuentos.....	70
Figura 4.7. Reporte de Facturas	71
Figura 4.8. Parámetros de Configuración.....	72
Figura 4.9. Detalle producto (categoría mujer).....	74
Figura 4.10. Carrito de compras.....	76
Figura 4.116. Mail con información sobre factura generada.....	77
Figura 4.12. Facturas Generadas	77
Figura 9.1. Pantalla home o de inicio.....	III
Figura 9.2. Pantalla de Categorías.....	III
Figura 9.3. Lista de los productos dentro de una sección y categoría.....	IV
Figura 9.4. Dashboard del administrador	IV
Figura 9.5. Lista de productos.....	V
Figura 9.6. Creación de un nuevo producto.....	V

Figura 9.7. Lista de las categorías	VI
Figura 9.8. Creación de nueva categoría	VI
Figura 9.9. Creación de nueva sección	VII
Figura 9.10. Creación de nueva sección	VII
Figura 9.11. Lista de impuestos.	VIII
Figura 9.12. Creación de nuevo impuesto.	VIII
Figura 9.13. Lista de códigos de descuento.	IX
Figura 9.14. Creación de nuevo código de descuento.	IX
Figura 9.15. Confirmación del Pago	X
Figura 9.16. Parámetros configurables.	X
Figura 9.17. Administración de la pantalla de inicio.	XI
Figura 9.18. Reporte de facturas.	XI
Figura 9.19. Pantalla de login.	XII
Figura 9.20. Creación de nuevo usuario	XII
Figura 9.21. Pantalla home o de inicio	XIII
Figura 9.22. Productos por categoría.....	XIII
Figura 9.23. Productos por sección	XIV
Figura 9.24. Menú lateral del usuario.....	XIV
Figura 9.25. Pantalla de perfil del usuario	XV
Figura 9.26. Pantalla de búsqueda.....	XV
Figura 9.27. Carrito de Compra	XVI
Figura 9.28. Pantalla de Historial de pago.....	XVI
Figura 9.29. Configuración del usuario	XVII

RESUMEN

TEMA: “DESARROLLO DE UN API REST CON SUS APLICACIONES WEB Y MÓVIL PARA VENTA DE ROPA ONLINE DE LA EMPRESA ROOSMAN”

Autores: David Wagner Santos Hernández

Jairo Adrián Serrano Parreño

Tutor: Darwin Rodolfo Caina Aysabucha

Al implementar sistemas informáticos orientados a automatizar el proceso de venta de una empresa, se aumenta el alcance de la misma generando mayores ganancias, además de permitir la expansión de la empresa en otros mercados del país.

Para esto se analizó los procesos que se deseaba automatizar, mejorar e implementar. Una vez que se definió estos requerimientos, se desarrolló un API REST con el cual se maneja toda la lógica de negocio, permitiendo realizar una aplicación Web y una aplicación móvil con lo cual se busca impulsar el crecimiento de la empresa. Para el almacenamiento de datos se utilizó bases de datos no relacionales (NoSQL) y para el desarrollo Web y aplicación móvil se manejó AngularJS, permitiendo la reutilización de código para ambos desarrollos y al establecer un *core* común, la escalabilidad del sistema es mucho más simple.

PALABRAS CLAVES: APLICACIONES WEB/API REST/ LUMEN/ NO SQL/
MONGODB/ ANGULARJS/ IONIC/ SCRUM/ JWT

ABSTRACT

TOPIC: “DEVELOPING OF A API REST WITH ITS APLICATION WEB AND MOBILE FOR SELLING CLOTHES ONLINE OF THE COMPANY ROOSMAN”

Authors: David Wagner Santos Hernández

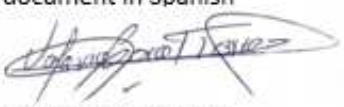
Jairo Adrián Serrano Parreño

Tutor: Darwin Rodolfo Caina Aysabucha

By implementing informatics systems oriented to automate a company’s selling process, the scope of that company is improved, generating higher earnings. Furthermore, allowing the company an expansion opportunity in other national markets.

For this purpose the processes required to be automated, improve and implemented were analyzed. Once the requirements were defined, an API REST was developed, which manage the whole business logic, allowing the developing of a web and mobile application, both oriented to improve and boost the company growth. For data storage, a non-relational database (NoSQL) was used, and for the web and mobile application development an AngularJS was managed, allowing reused of the code for both. And, by establishing a common core the system scalability is much simpler.

KEYWORDS: WEB APLICATION/API REST/ LUMEN/ NO SQL/ MONGODB/
ANGULARJS/ IONIC/ SCRUM/ JWT

I certify that the above and foregoing is a true and correct translation of the original document in Spanish

Valeria Coral Torres
IELTS Certified 15CO009598CORV001A
ID: 1723207237

INTRODUCCIÓN

A partir del año 2000, REST (*Representational State Transfer*, Transferencia de Estado Representacional) revolucionó por completo la ingeniería de software, permitiendo crear aplicaciones y servicios que pueden ser usadas por cualquier dispositivo que entienda HTTP. Esta arquitectura es más simple y convencional que las que se han usado en los últimos diez años como SOAP (*Simple Object Access Protocol*) y XML-RPC.

Esta arquitectura de desarrollo de proyectos y servicios Web, fue definida por Roy Fielding, quien se destaca por ser uno de los principales desarrolladores de los protocolos Web, incluyendo HTTP y URI. Roy Fielding en su disertación doctoral “*Architectural Styles and the Design of Network-based Software Architectures*”, Define una metodología para los estilos arquitectónicos de alto nivel que expresan las principales ideas detrás de un enfoque arquitectónico. Siendo esta arquitectura un estilo de alto nivel, puede ser implementado mediante la utilización de diferentes tecnologías. El API es un conjunto de funciones o procedimientos utilizados por los programas informáticos para acceder a los servicios del sistema operativo, bibliotecas de software, u otros sistemas; y se caracterizan por ser de tipo SOAP y REST.

El de tipo SOAP es un protocolo estándar que define el cómo los objetos en diferentes procesos, pueden comunicarse mediante un intercambio de datos XML.

El de tipo REST es en cambio una forma sencilla de enviar y recibir datos entre el cliente y el servidor, que no dispone de muchos estándares, se caracteriza además porque puede enviar y recibir datos como JSON, XML e incluso texto sin formato.

Este conjunto de funciones o procedimientos (APIs) comienza a desarrollarse gracias al comercio electrónico. A finales del año 2000 *eBay* (sitio destinado a la subasta de productos a través de Internet) diseña su interfaz de programación de aplicaciones (API), en su intención

por impulsar el área de las soluciones de comercio electrónico, a la vez que también fomenta la de las APIs. Pero, *eBay* no fue la única en ese empeño ya que en el año 2002, el gigante *Amazon* (compañía estadounidense de comercio electrónico) hizo lo propio.

Desde el año 2006 hasta el 2015 la implementación de las APIs ha tenido un crecimiento exponencial, a tal punto que en la actualidad se estima hay cerca de 16,784 APIs publicadas¹.

La utilización de las APIs es tan variado, ya que puede ser aplicado como referencia para apps móviles, incrementar el ecosistema de clientes y socios, lograr un alcance masivo para transacciones y contenidos, así como imputar nuevos modelo de negocio, liderar la innovación interna, entre otros. Y pueden ser aplicadas en los ámbitos industriales, comerciales, sociales, financieros, y en actividades puntuales como compras, pago, música, comida y demás.

Definidas las API como un conjunto de funciones y procedimientos; y la REST como un arquitectura de alto nivel que puede ser utilizados por cualquier cliente o dispositivo; nos hemos propuesto, desarrollar un API REST para la venta de ropa online de la empresa ROOSMAN, que aún no dispone de un sistema similar y desea incorporar esta solución tecnológica para ser más competitivo, más eficiente, optimizar sus costos, tener un mayor posicionamiento de la empresa en un mercado nacional, regional y local, de esta forma mantener un mayor nivel de satisfacción y cercanía al cliente.

La actividad de ROOSMAN está centrada en la comercialización de ropa con diseños exclusivos para damas y caballeros, para lo cual la implementación de este sistema en su actividad comercial, constituyen en una herramienta válida para el cumplimiento de los objetivos de ROOSMAN y su visión.

¹ ProgrammableWeb, "API Directory", Recuperado el 10/11/2016 de:
<https://www.programmableweb.com/apis/directory>

MARCO TEÓRICO

1.1. REST²

Representational State Transfer (REST), es un tipo de arquitectura de desarrollo Web que usa el protocolo HTTP para su funcionamiento, en tal sentido, para poder usar REST no necesita de la implementación de ningún otro protocolo adicional sino el mismo que usa la WEB, a diferencia de otros tipos de arquitectura.

En el desarrollo de servicios Web o una aplicación Web y sobre todo en una API REST se dan tres niveles de calidad que se describen en el modelo “*Richardson Maturity Model*” siendo estos:

1. Uso correcto de URIs
2. Uso correcto de HTTP.
3. Implementar *Hypermedia*.

HTTP que es un protocolo sin estado, cuando es utilizado por REST nos posibilita una mayor escalabilidad, y permite alternar con distintas tecnologías para servir determinados recursos de una misma API, por lo tanto temas como el almacenamiento de variables de sesión, ya no constituyen una preocupación técnica.

1.1.1. Uso correcto de URIs

Las URL (*Uniform Resource Locator*), al ser un tipo de URI, (*Uniform Resource Identifier*), esta permite localizar y reconocer de forma única el recurso, además de poder acceder a él se puede compartir su ubicación. Cada página, información en una sección, archivo, cuando se habla de REST, se los nombra como recursos, por lo tanto es la información a la que se desea acceder, modificar o borrar, independientemente del formato que tenga.

² Asier Marqués, “*Conceptos sobre APIs REST*”, Recuperado el 10/11/16 de: <http://asiermarques.com/2013/conceptos-sobre-apis-rest/>

Existen varias reglas que se deben utilizar para ponerle nombre a la URI de un recurso:

- Los nombres de URI no deben implicar una acción, por lo tanto debe evitarse usar verbos en ellos.
- Deben ser únicas, no debemos tener más de una URI para identificar un mismo recurso.
- Deben ser independiente de formato.
- Deben mantener una jerarquía lógica.
- Los filtrados de información de un recurso no se hacen en la URI.

1.1.2. Uso correcto de HTTP

Al momento de desarrollar APIs REST los aspectos claves que hay que priorizar son:

- **Métodos HTTP:**

Para manipular los recursos, HTTP nos proporciona los métodos con los cuales debemos operar siendo estos los siguientes:

- GET: Para consultar y leer recursos
- POST: Para crear recursos
- PUT: Para editar recursos
- DELETE: Para eliminar recursos.
- PATCH: Para editar partes concretas de un recurso.

- **Códigos de estado:**

En el desarrollo de una API se debe evitar crear herramientas propias en lugar de utilizar las que ya han sido creadas, pensadas y testeadas; ya que al crear estas herramientas se cae en uno de los errores más frecuentes en el desarrollo de las APIs, que ocurre mayormente con los códigos de error y estado.

Algunos de estos códigos pueden ser:

- 100: Informan al navegador de algunas acciones que se van a realizar:
- 200: Indican que la petición del navegador se ha recibido, procesado y respondido correctamente.
- 300: Indican que el navegador debe realizar alguna acción adicional para que la petición se complete (como por ejemplo redirigirse a otra página).
- 400: Indican que se ha producido un error cuyo responsable es el navegador:
- 500: Indican que se ha producido un error cuyo responsable es el servidor

En la realización de una operación, es necesario saber si dicha operación ha sido exitosa o ha fallado, de haber ocurrido esto nos interesa saber el por qué ha fallado; ya que HTTP tiene un listado de códigos de estado muy amplio que cubre todas las posibles sucesos, que se deben añadir en las respuestas cuando las operaciones han sido exitosas o se produjo algún error.

- **Aceptación de tipos de contenido:**

HTTP permite especificar en qué formato se desea recibir el recurso en orden de preferencia y para ello utiliza el *header Accept*. Al cliente se devolverá el *header Content-Type*, para que se conozca en que formato se devuelve la respuesta.

1.1.3. Implementar *Hypermedia*.

Considerando que la finalidad de *Hypermedia* es conectar mediante vínculos las aplicaciones clientes con las APIs y que además se añade información extra al recurso sobre su posible conexión a otros recursos relacionados con él, en los casos que sean necesarios.

La *Hypermedia* es tan útil para el cliente ya que evita hacer mantenimientos en cada uno de los URLs de los recursos si en un futuro estas cambian, ya que los clientes no tienen conocimiento de las URLs de dichos recursos. También es útil cuando se desea automatizar procesos entre APIs y así evitar la interacción humana.

1.2. Aplicaciones SPA o páginas Web SPA (*Single Page Applications*)³.

En el desarrollo de aplicaciones Web se da una tendencia importante hacia las denominadas SPA. Cuyo principal objetivo es conseguir una mejora importante en la satisfacción del usuario, ya que se obtiene una reducción de los tiempos de espera o latencia entre vistas (similar a las aplicaciones nativas).

Habitualmente la lógica de negocio (el código ejecutable) de aplicaciones Web se realiza íntegramente en el lado del servidor y se confía en la propia naturaleza del sistema de URLs el mostrar una “vista de aplicación” u otra. Por ejemplo en un *e-commerce* el detalle de producto estará en una URL concreta, mientras que la pantalla de registro estará bajo otra URL totalmente diferente.

Para el navegador, cada URL es completamente independiente del resto; Aunque tengan los mismos estilos y/o plantillas, estos tienen que volver a ser procesados desde cero. Para la gran mayoría de páginas Web dinámicas, implica que al cambiar entre vistas sufra el problema de la latencia en la Web.

1.2.1. Arquitectura básica de una SPA

En esencia una SPA es la interfaz de la aplicación Web implementada casi íntegramente en el navegador, aunque como toda página Web tiene una base importante de HTML y CSS. Y

³ Javier Tejero, “SPA, un paradigma de Arquitectura de Aplicaciones Web en auge”, Recuperado el 10/11/16 de: <https://cink.es/blog/2013/10/07/spa-un-paradigma-de-arquitectura-de-aplicaciones-Web-en-auge/>

que para realizar páginas o aplicaciones SPA se pueden usar varios *frameworks* de *JavaScript* que nos ayudan a realizarlo.

Ventajas	Desventajas
UI más rápidas	Son aplicaciones más difíciles de construir
Son más interactivas	Necesita grandes conocimientos de <i>JavaScript</i>
Pueden trabajar en <i>offline</i>	Al basarse <i>JavaScript</i> es difícil elegir
Son perfectas para realizar aplicaciones móviles HTML5	

Tabla 0.1. Ventajas y Desventajas de las SPAs
Fuente: Autoría Propia

Todas las vistas de la interfaz de la aplicación están contenidas en la SPA, y que al realizar la primera consulta al servidor esta todas las vistas de la interfaz, solo posponiendo recursos pesados tales como: Grandes cantidades de datos, Imágenes, videos, entre otros.

En una aplicación de este tipo el tiempo de respuesta es mucho más rápido que el de una aplicación Web tradicional, esto se debe entre otros aspectos, a que al recargar una página no se envía todo el archivo HTML sino únicamente la información necesaria para actualizar el contenido.

Con las SPAs se elimina por completo uno de los principales cuellos de botella. El problema de la latencia, así podemos conseguir que la aplicación Web alcance la velocidad de cualquier aplicación nativa en lo que se refiere al tiempo de espera al cambiar de vistas. Esto se aprecia notablemente en cualquier dispositivo, especialmente cuando se utilizan dispositivos móviles.

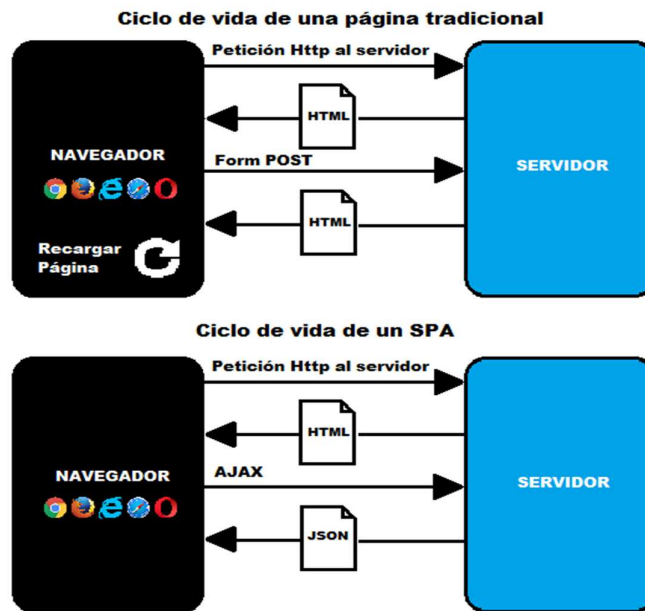


Figura 0.1. Ciclo de vida de las aplicaciones Web tradicionales y SPA
Fuente: Autoría Propia

1.3. Lumen *micro-framework* de Laravel

Lumen nace como la necesidad de poder contar con un *framework* que sea más sencillo, más liviano, una mayor velocidad de ejecución en las peticiones y con características más específicas. Ya que otros *frameworks* de PHP como Laravel o *Symfony*, contaban con muchas características y componentes extras para desarrollar Aplicaciones Web, tanto la lógica de negocio como la vista o *Front-End*.

Lumen⁴ es un *micro-framework* para PHP creado por Taylor Otwell (mismo creador de Laravel), que comparte muchos componentes de Laravel, pero Lumen es una versión más liviana y esta es orientada a la creación de APIs y micro servicios, pero también se puede usar para crear sitios Web.

⁴ Duilio Palacios, "Lumen, la versión micro framework de Laravel", Recuperado el 10/11/16 de: <https://styde.net/lumen-la-version-micro-framework-de-laravel-php/>

1.4. AngularJS⁵

Es un *framework* MVC (Modelo, Vista, Controlador) de *JavaScript* desarrollado por la empresa Google, la cual es de código libre y ayuda al Desarrollo *Web Front-End* que permite crear aplicaciones SPA.

Los *frameworks* que implementan el patrón MVC y ayudaron a separar conceptos para no tener que crear una función según nuestra necesidad en el desarrollo.

- Vistas: Será el HTML y todo lo que representa datos o información para el usuario final.
- Controladores: Se encargará de la lógica de la aplicación y sobre todo de las llamadas "Factorías" y "Servicios" para obtener datos de los servidores o de la memoria local en HTML5.
- Modelo de la vista: En AngularJS el "Modelo" es algo más que el MVC tradicional, es decir, las vistas son algo más que el modelo de datos. El "Scope" que es el modelo en AngularJS, es la información que es útil para el programador pero que no forma parte de la lógica de negocio.

Además del patrón principal, descrito hasta ahora se tiene los llamados módulos que AngularJS propone para que el desarrollador tenga un orden al escribir el código siendo este más limpio y gracias a esto se facilite su desarrollo y el mantenimiento.

1.5. IONIC

Es una herramienta, gratuita y de código abierto, para el desarrollo de aplicaciones híbridas basadas en HTML5, CSS y JS.

⁵ Alberto Basalo, Miguel Ángel Alvarez, "Qué es AngularJS", Recuperado el 10/11/16 de: <http://www.desarrolloweb.com/articulos/que-es-angularjs-descripcion-framework-javascript-conceptos.html>

IONIC se construye sobre Apache Cordova, que es otro *framework* que nos ayuda en el desarrollo de aplicaciones híbridas igualmente basado en HTML5, CSS y JS. El cual cuenta con APIs *JavaScript* que ayuda a interactuar con las características nativas del móvil, como es: Acceder a la cámara, emails, agenda de contactos, realizar llamadas, etc.

Se aprovecha lo mejor de Apache Cordova para usarlo en IONIC, se considera que IONIC es el SDK del *Front-End* para el desarrollar aplicaciones híbridas móviles con HTML5. Además IONIC usa AngularJS como su *framework JavaScript*, con el fin brindar todos los beneficios que se tiene de AngularJS al desarrollo de aplicaciones híbridas.

IONIC también cuenta con sus propios componentes para CSS y JS, para ayudar en el desarrollo de las aplicaciones híbridas.

Las aplicaciones híbridas usan componentes *Web view* (navegadores que ejecutan HTML, CSS y *JavaScript*), pero en lugar de ser ejecutado en un navegador Web, este se empaqueta usando diferentes *asset* de los lenguajes nativos, que nos proporciona Apache Cordova, para poder generar aplicaciones nativas que puedan ser ejecutadas en los diferentes sistemas operativos de los dispositivos móviles.

1.6. Bases de datos No Sql⁶

Las bases de datos NoSQL (*Not only SQL* – No sólo SQL) son estructuras que nos permiten almacenar información en aquellas situaciones en las que las bases de datos relacionales generan ciertos problemas, debido principalmente a problemas de escalabilidad y rendimiento de las bases de datos relacionales, donde se dan cita miles de usuarios concurrentes y con millones de consultas diarias. Además las bases de datos NoSQL son sistemas de almacenamiento de información que no cumplen con el esquema entidad-relación. Tampoco

⁶ ACENS, “Bases de datos NoSQL. Qué son y tipos que nos podemos encontrar”, Recuperado el 10/11/16 de: <https://www.acens.com/wp-content/images/2014/02/bbdd-nosql-wp-acens.pdf>

utilizan una estructura de datos en forma de tabla donde se van almacenando los datos sino que para el almacenamiento hacen uso de otros formatos como clave–valor, mapeo de columnas, grafos, entre otros.

Dependiendo de la forma en la que almacenen la información, se puede encontrar varios tipos distintos de bases de datos NoSQL. Los tipos más utilizados son:

- Bases de datos clave – valor
- Bases de datos en grafo
- Bases de datos orientadas a objetos
- Autenticación basada en *tokens*
- Bases de datos documentales

1.6.1. Ventajas de los sistemas NoSQL

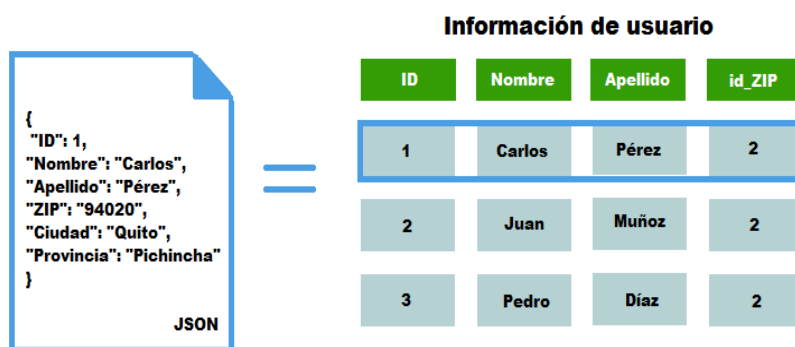
Esta forma de almacenar la información ofrece ciertas ventajas sobre los modelos relacionales. Entre las ventajas más significativas podemos destacar:

- Se ejecutan en máquinas con pocos recursos: Estos sistemas, a diferencia de los sistemas basados en SQL, se pueden montar en máquinas con un costo más reducido.
- Escalabilidad horizontal: Para mejorar el rendimiento de estos sistemas simplemente se consigue añadiendo más nodos, con la única operación de indicar al sistema cuáles son los nodos que están disponibles.
- Pueden manejar gran cantidad de datos: Esto es debido a que utiliza una estructura distribuida, en muchos casos mediante tablas *Hash*.
- No genera cuellos de botella: El principal problema de los sistemas SQL es que necesitan transcribir cada sentencia para poder ser ejecutada, y cada sentencia compleja requiere además de un nivel de ejecución aún más complejo, lo que

constituye un punto de entrada en común, que ante muchas peticiones puede ralentizar el sistema.

1.6.2. Bases de datos documentales

Este tipo de bases de datos NoSQL almacena la información como un documento, generalmente utilizando para ello una estructura simple como JSON o XML y donde se utiliza una clave única para cada registro como se muestra en la Figura 1.2.



*Figura 0.2. Bases de datos No SQL de documentos
Fuente: acens white papers*

Este tipo de implementación permite, además de realizar búsquedas por clave-valor, realizar consultas más avanzadas sobre el contenido del documento. Son las bases de datos NoSQL más versátiles. Se pueden utilizar en gran cantidad de proyectos, incluyendo muchos que tradicionalmente funcionarían sobre bases de datos relacionales. Algunos ejemplos de este tipo son MongoDB o CouchDB.

1.7. Levantamiento de Requerimientos con la Técnica JAD (*Joint Application Design*)⁷

Es un concepto de diseño de sistemas interactivos donde participan grupos de discusión afines al proyecto. JAD fue diseñado para que los desarrolladores de sistemas y usuarios de diferentes orígenes y opiniones juntos en un entorno productivo y creativo. Las reuniones eran una forma de obtener los requisitos de calidad y especificaciones.

JAD es una técnica para determinar rápidamente los requerimientos de un sistema por medio de la lluvia de ideas de los grupos afines al proyecto. Para lo cual se crea un grupo compuesto de los principales usuarios, administradores y analistas de sistemas. Una vez creado el grupo se realizan reuniones con la finalidad de recopilar datos, debatir ideas, conciliar las diferencias, analizar las necesidades, y obtener soluciones alternativas.

JAD puede ser utilizado para realizar diferentes análisis como: de viabilidad, de costos, de riesgos, de implementación, de factibilidad, entre otros. Generalmente, el diseño de especificaciones tales como diseño de *wireframes*, diseño de *Front-End* diseño de pruebas, diagramas de capas, diagramas de flujo, casos de pruebas, entre otros. Suelen estar contempladas en las sesiones JAD

1.7.1. Desarrollo de taller JAD

Para el desarrollo de talleres JAD hay que contar por lo mínimo de tres partes que son:

- El patrocinador del proyecto.- Es quien presupuesta el proyecto, el dueño del sistema. Tienen el lugar más alto en la organización, de modo que ellos pueden tomar las decisiones y proporcionar los recursos necesarios y apoyar para el proyecto. Las Responsabilidades del Patrocinador de proyecto:

⁷ Michael Martinez ,” Joint Application Design (JAD)”, Recuperado el 10/11/16 de: <https://prezi.com/twuzzym3th5p/joint-application-design-jad/>

- Asegurar que los clientes correctos son parte del grupo
 - Asegurar que hay suficiente personal de soporte técnico en el proyecto
 - Ayudar en la selección de casos de la prueba
 - Ayudar en la definición del alcance y funcionalidad
 - Evaluar si el sistema es eficaz o no.
- Líder del Proyecto.- Tiene que estar comprometido al proyecto, tener un conocimiento de fondo del área comercial y sistemas de información actuales relacionados. Ellos necesitan ser entusiastas y objetivos y no permitirle a ningún solo individuo dominar el grupo. Responsabilidades del Líder de proyecto:
 - Asegurar que todos los roles de su equipo estén ocupados
 - Asegurar que las reuniones se planifiquen.
 - Asegurar que las agendas se planifican y se sigan.
 - Asegurar que se asignan las tareas y se cumplen, y que el listado de tareas se ejecutan en la secuencia prevista con su línea de tiempo.
 - Clientes.- Son los que conocen cómo funcionara el sistema y cómo se usa. Ellos ayudarán al equipo a comprender las tareas manipuladas por el sistema. Responsabilidades:
 - Definir la información con la que el proceso tiene que tratar.
 - Crear casos de uso para su prueba.
 - Analizar los obstáculos al éxito en el ambiente actual.

1.8. Metodologías Ágiles ⁸

Las metodologías ágiles son una serie de técnicas para la gestión de proyectos que han surgido como contraposición a los métodos clásicos de gestión. Aunque surgieron en el ámbito del desarrollo de software, también han sido exportadas a otro tipo de proyectos.

Todas las metodologías que se consideran ágiles cumplen con el manifiesto ágil que no es más que una serie de principios que se agrupan en cuatro valores:

1. Los individuos y su interacción, por encima de los procesos y las herramientas.
2. El software que funciona, frente a la documentación exhaustiva.
3. La colaboración con el cliente, por encima de la negociación contractual.
4. La respuesta al cambio, por encima del seguimiento de un plan.

Lo que se desea es minimizar el impacto de las tareas que no son totalmente imprescindibles para conseguir el objetivo del proyecto. Se pretende aumentar la eficiencia de las personas involucradas en el proyecto y, como resultado de ello, minimizar el costo.

Para enumerar las principales diferencias de una Metodología Ágil respecto de las Metodologías Tradicionales la tabla 1.2 recoge estas diferencias que no se refieren sólo al proceso en sí, sino también al contexto de equipo y organización que es más favorable a cada uno de estas filosofías de procesos de desarrollo de software.

⁸ Patricio Letelier, Carmen Penadés, "Metodologías ágiles para el desarrollo de software: eXtreme Programming (XP)", Recuperado el 10/11/16 de: <http://www.cyta.com.ar/ta0502/v5n2a1.htm>

Metodología Ágil	Metodología Tradicional
Pocos Artefactos. El modelado es prescindible, modelos desechables.	Más Artefactos. El modelado es esencial, mantenimiento de modelos
Pocos Roles, más genéricos y flexibles	Más Roles, más específicos
No existe un contrato tradicional, debe ser bastante flexible	Existe un contrato prefijado
Cliente es parte del equipo de desarrollo (además in-situ)	El cliente interactúa con el equipo de desarrollo mediante reuniones
Orientada a proyectos pequeños. Corta duración (o entregas frecuentes), equipos pequeños (< 10 integrantes) y trabajando en el mismo sitio	Aplicables a proyectos de cualquier tamaño, pero suelen ser especialmente efectivas/usadas en proyectos grandes y con equipos posiblemente dispersos
La arquitectura se va definiendo y mejorando a lo largo del proyecto	Se promueve que la arquitectura se defina tempranamente en el proyecto
Énfasis en los aspectos humanos: el individuo y el trabajo en equipo	Énfasis en la definición del proceso: roles, actividades y artefactos
Basadas en heurísticas provenientes de prácticas de producción de código	Basadas en normas provenientes de estándares seguidos por el entorno de desarrollo
Se esperan cambios durante el proyecto	Se espera que no ocurran cambios de gran impacto durante el proyecto

Tabla 0.2. Diferencias entre metodologías ágiles y no ágiles

Fuente: <http://www.cyta.com.ar>

1.9. SCRUM⁹

Es un marco de trabajo de procesos que ha sido usado para gestionar el desarrollo de productos complejos desde principios de los años 90. *Scrum* no es un proceso o una técnica

⁹ Ken Schwaber y Jeff Sutherland, “*La Guía de Scrum*”, Recuperado el 10/11/16 de: <http://www.scrumguides.org/docs/scrumguide/v1/Scrum-Guide-ES.pdf>

para construir productos; en lugar de eso, es un marco de trabajo dentro del cual se pueden emplear varias técnicas y procesos. Muestra la eficacia relativa de las prácticas de gestión de producto y las prácticas de desarrollo, de modo que podamos mejorar.

El marco de trabajo *Scrum* consiste en los Equipos *Scrum*, roles, eventos, artefactos y reglas asociadas. Cada componente dentro del marco de trabajo sirve a un propósito específico y es esencial para el éxito de *Scrum* y para su uso. Las estrategias específicas para usar el marco de trabajo *Scrum* son diversas.

1.9.1. El Equipo Scrum (*Scrum Team*)

El Equipo *Scrum* consiste en un Dueño de Producto (*Product Owner*), el Equipo de Desarrollo (*Development Team*) y un *Scrum Master*. Los Equipos *Scrum* son auto organizados y multifuncionales. Los equipos auto organizados eligen la mejor forma de llevar a cabo su trabajo y no son dirigidos por personas externas al equipo. Los equipos multifuncionales tienen todas las competencias necesarias para llevar a cabo el trabajo sin depender de otras personas que no son parte del equipo. El modelo de equipo en *Scrum* está diseñado para optimizar la flexibilidad, la creatividad y la productividad.

Los Equipos *Scrum* entregan productos de forma iterativa e incremental, maximizando las oportunidades de obtener retroalimentación. Las entregas incrementales de producto aseguran que siempre estará disponible una versión potencialmente útil y funcional del producto.

1.9.2. El Dueño de Producto (*Product Owner*)

El Dueño de Producto es el responsable de maximizar el valor del producto y del trabajo del Equipo de Desarrollo. El cómo se lleva a cabo esto podría variar ampliamente entre distintas organizaciones, Equipos *Scrum* e individuos. El Dueño de Producto es la única persona responsable de gestionar la Lista del Producto. La gestión de la Lista del Producto incluye:

- Expresar claramente los elementos de la Lista del Producto
- Ordenar los elementos en la Lista del Producto para alcanzar los objetivos y misiones de la mejor manera posible
- Optimizar el valor del trabajo desempeñado por el Equipo de Desarrollo
- Asegurar que la Lista del Producto es visible, transparente y clara para todos, y que muestra aquello en lo que el equipo trabajará a continuación
- Asegurar que el Equipo de Desarrollo entiende los elementos de la Lista del Producto al nivel necesario.

El Dueño de Producto podría hacer el trabajo anterior, o delegarlo en el Equipo de Desarrollo. Sin embargo, en ambos casos el Dueño de Producto sigue siendo el responsable de dicho trabajo.

El Dueño de Producto es una única persona, no un comité. El Dueño de Producto podría representar los deseos de un comité en la Lista del Producto, pero aquellos que quieran cambiar la prioridad de un elemento de la Lista deben hacerlo a través del Dueño de Producto.

Para que el Dueño de Producto pueda hacer bien su trabajo, toda la organización debe respetar sus decisiones. Las decisiones del Dueño de Producto se reflejan en el contenido y en la priorización de la Lista del Producto. No está permitido que nadie pida al Equipo de Desarrollo que trabaje con base en un conjunto diferente de requerimientos, y el Equipo de Desarrollo no debe actuar con base en lo que diga cualquier otra persona

1.9.3. El Equipo de Desarrollo (*Development Team*)

El Equipo de Desarrollo consiste en los profesionales que desempeñan el trabajo de entregar un Incremento de producto “Terminado”, que potencialmente se pueda poner en producción, al final de cada *Sprint*. Solo los miembros del Equipo de Desarrollo participan en la creación del Incremento.

Los Equipos de Desarrollo son estructurados y empoderados por la organización para organizar y gestionar su propio trabajo. La sinergia resultante optimiza la eficiencia y efectividad del Equipo de Desarrollo.

1.9.4. El Scrum Master

El responsable de asegurar que *Scrum* es entendido y adoptado. Los *Scrum Masters* hacen esto asegurándose de que el Equipo *Scrum* trabaja ajustándose a la teoría, prácticas y reglas de *Scrum*.

Ayuda a las personas externas al Equipo *Scrum* a entender qué interacciones con el Equipo pueden ser de ayuda y cuáles no, además ayuda a todos a modificar estas interacciones para maximizar el valor creado por el Equipo.

1.9.5. Eventos de Scrum

En *Scrum* existen eventos predefinidos con el fin de crear regularidad y minimizar la necesidad de reuniones no definidas en *Scrum*. Todos los eventos son bloques de tiempo (*time-boxes*), de tal modo que todos tienen una duración máxima. Una vez que comienza un *Sprint*, su duración es fija y no puede acortarse o alargarse. Los demás eventos pueden terminar siempre que se alcance el objetivo del evento, asegurando que se emplee una cantidad apropiada de tiempo sin permitir desperdicio en el proceso.

Además del propio *Sprint*, que es un contenedor del resto de eventos, cada uno de los eventos de *Scrum* constituye una oportunidad formal para la inspección y adaptación de algún aspecto. Estos eventos están diseñados específicamente para habilitar las vitales transparencia e inspección. La falta de alguno de estos eventos da como resultado una reducción de la transparencia y constituye una oportunidad perdida para inspeccionar y adaptarse.

1.9.6. El *Sprint*

El corazón de *Scrum* es el *Sprint*, es un bloque de tiempo (*time-box*) de un mes o menos durante el cual se crea un incremento de producto “Terminado”, utilizable y potencialmente desplegable. Es más conveniente si la duración de los *Sprints* es consistente a lo largo del esfuerzo de desarrollo. Cada nuevo *Sprint* comienza inmediatamente después de la finalización del *Sprint* previo.

Los *Sprints* contienen y consisten de la Reunión de Planificación del *Sprint* (*Sprint Planning Meeting*), los *Scrums* Diarios (*Daily Scrums*), el trabajo de desarrollo, la Revisión del *Sprint* (*Sprint Review*), y la Retrospectiva del *Sprint* (*Sprint Retrospective*).

Durante el *Sprint*:

- No se realizan cambios que puedan afectar al Objetivo del *Sprint* (*Sprint Goal*)
- Los objetivos de calidad no disminuyen
- El alcance puede ser clarificado y renegociado entre el Dueño de Producto y el

Equipo de Desarrollo a medida que se va aprendiendo más.

Cada *Sprint* puede considerarse un proyecto con un horizonte no mayor de un mes. Al igual que los proyectos, los *Sprints* se usan para lograr algo. Cada *Sprint* tiene una definición de qué se va a construir, un diseño y un plan flexible que guiará la construcción y el trabajo y el producto resultante.

Los *Sprints* están limitados a un mes calendario. Cuando el horizonte de un *Sprint* es demasiado grande la definición de lo que se está construyendo podría cambiar, la complejidad podría elevarse y el riesgo podría aumentar. Los *Sprints* habilitan la predictibilidad al asegurar la inspección y adaptación del progreso al menos en cada mes calendario. Los *Sprints* también limitan el riesgo al costo de un mes calendario.

METODOLOGÍA

1.10. Requerimientos y Metodología de desarrollo

Para la metodología de desarrollo se realizó el levantamiento de requerimientos usando la técnica JAD (*Joint Application Design*), Conformación del equipo *scrum* en los cuales se definieron el dueño del producto y el equipo de desarrollo, se desarrollaran reuniones para definir el Sprint con el equipo del trabajo y dueño del producto, se realizaron *scrum* diario entre el equipo de desarrollo y revisiones de Sprint. Los cuáles serán detallados uno a uno más adelante el planteamiento y su desarrollo.

➤ Levantamiento de requerimientos usando JAD

Para el levantamiento de los primeros requerimientos se usó la Técnica JAD, la cual consiste en reuniones con el patrocinador o dueño del producto. En la cual se va a reunir personas que puedan aportar ideas y que las define el patrocinador, también se integra un grupo de desarrolladores, los cuales van a hablar y discutir sobre los requerimientos que debe tener el software que se desea desarrollar.

En dichas reuniones se realizó un informe sobre los temas tratados y sobre el levantamiento de requerimientos, el cual se encuentra detallado en el anexo A.

La empresa desea auto administrar la aplicación Web, para lo cual va a tener una sección exclusiva de administrador en la cual va a poder administrar algunas características del sistema:

- Categorías
- Secciones
- Productos
- Impuestos
- Código de Descuento
- Visualizar Facturas

- Confirmar Depósito
- Pantalla de Inicio
- Reportes

Para esto se acordó una jerarquía en el tratamiento de los productos como se puede observar en la siguiente figura 2.1, estando el producto contenido en las secciones y estas a su vez contenidas en las categorías:

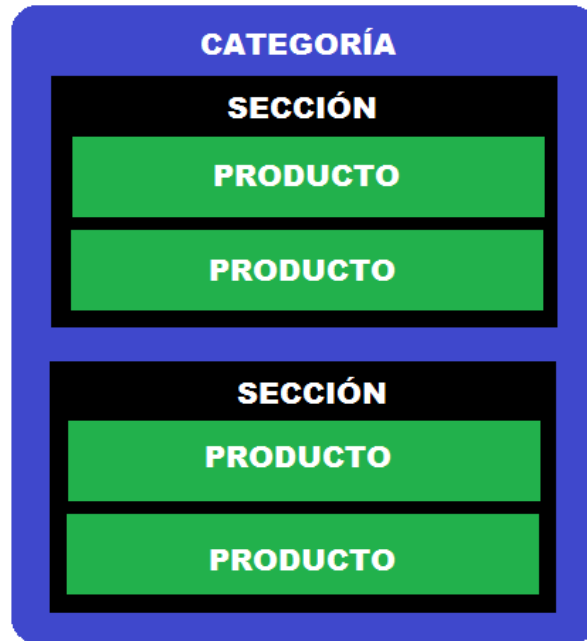


Figura 0.1 Jerarquía para manejo de productos
Fuente: Autoría Propia

Y en base a esto definir las funcionalidades a desarrollar en el API, las cuales van a ser:

- Creación, Edición, Borrado, Consulta de Categorías
- Creación, Edición, Borrado, Consulta de Secciones
- Creación, Edición, Borrado, Consulta de Productos
- Creación, Edición, Borrado, Consulta de Impuestos
- Creación, Edición, Borrado, Consulta de Códigos de Descuento
- Consultas de todas las facturas generadas, por fechas y por cliente

- Confirmación de Pagos para realizar la factura del cliente
- Administración de las imágenes que van a aparecer en la pantalla de inicio
- Reportes que va a poder generar el administrador

La parte a la que accedera los usuarios son:

- *Login* a la aplicación
- Registro para la aplicación
- Recuperación de contraseña
- Visualización de la pantalla de inicio
- Visualización de la pantalla de categorías
- Visualización de los productos según su sección
- Detalle del producto
- Carrito de Compras
- Perfil de Usuario
- Historial de pagos
- Configuración de la cuenta del usuario

Para el desarrollo de la aplicación móvil, se desea que cuente con las siguientes funcionalidades que son:

- *Login* a la aplicación
- Registro para la aplicación
- Recuperación de contraseña
- Visualización de los productos nuevos al ingresar
- Visualización de las categorías
- Visualización de todos los productos de las categorías
- Visualización de las secciones que pertenecen a una categoría

- Visualización de los productos de las secciones
- Carrito de compras
- Detalle del producto
- Perfil de usuario
- Historial de pagos
- Configuración de la cuenta del usuario

Se deja constancia que aquí solo se realizó el levantamiento de los requerimientos técnicos o la lógica del negocio, que es cómo va a funcionar el software, aun no se ha realizado ningún levantamiento de requerimiento de la estética de los proyecto, del cómo se va a visualizar las aplicaciones.

- Conformación del Equipo *Scrum*

La conformación del equipo *Scrum* será de la siguiente forma:

- Dueño del Producto: Sra. Rosa Hernández
- Equipo de Desarrollo: Wagner Santos y Jairo Serrano
- *Scrum Master*: Ing. Darwin Caina

Cada una de las partes del *Scrum* cumplirá con su función designada y contribuirá al desarrollo del proyecto

- Reunión y realización de los *Sprint*

Para la realización de las reuniones el equipo de desarrollo analiza y plantea las actividades que pueden ir en el Sprint que se va a realizar, las cuales se va a presentar al dueño del producto y ayudará a ver cuáles son prioridades y cuáles no, para poder así realizar la lista para comenzar los Sprint.

En las reuniones de los *Sprints*, que será del equipo de desarrollo con el dueño del Producto, se tratarán temas en base a los requerimientos y en base al diseño *Front-End* que tendrá el proyecto en sus diferentes aplicaciones, aquí se discutirá la parte visual o estética y las funcionalidades que van a poseer cada aplicación, en las cuales el equipo de desarrollo expondrá que es viable en función al tiempo del *Sprint* y que no; para poder llegar a un consenso y poder comenzar con el desarrollo del *Sprint*.

- *Srum Diary* entre el equipo de Desarrollo

Se realizarán reuniones pequeñas entre el equipo de desarrollo, para analizar las actividades que se han realizado, cuales están en proceso de desarrollo y actividades pendientes, para ver cómo avanza el desarrollo del proyecto y poder reorganizar el tiempo, para poder cumplir con la meta planteada en el *Sprint*.

Esta reunión no tomará más de quince minutos, ya que los miembros del equipo de desarrollo solo exponen las actividades que se están realizando y si es que ocurrió alguna novedad en alguna actividad del *Sprint* se la informa aquí para poder buscar una solución a ese problema.

1.11. Selección de herramientas

Ya definido las partes con las que va a contar la aplicación Web y móvil, ahora se analizará el lenguaje de programación y las herramientas que se utilizará para comenzar el desarrollo de cada de las partes descritas. Para el desarrollo del API se eligió el lenguaje de programación PHP que es un lenguaje diseñado para la *Web*, actualmente se encuentra su versión 7.1.0, también se utilizará el *framework* Lumen que actualmente se encuentra en su versión 5.3 , que permita manejar numerosas peticiones y brinde la mayor velocidad en respuesta a dichas peticiones.

Para la autenticación por medio de *tokens* se utilizará JWT (*Json Web Tokens*), que es el que ayudará a realizar el *token* en base a la información del usuario, este es un *plugin* de libre distribución y se puede encontrar para diferentes lenguajes de programación, y cuenta con una vasta documentación del uso y la implementación.

Para el desarrollo del *Front-End* se va a realizar una aplicación Web y una aplicación móvil. Para esto se utilizará los lenguajes de programación HTML, CSS y JavaScript, para ambas aplicaciones. Para el desarrollo de la aplicación Web se va a utilizar AngularJS, un *framework* de JavaScript, con el cual se desarrollara una aplicación Web SPA, que cuenta con numerosas ventajas. Actualmente se encuentra en su versión 1.5.8, que es su versión estable, pero también cuenta con su versión 2.1.0 que aún se encuentra en fase de desarrollo.

Para el desarrollo de la aplicación móvil se va a utilizar IONIC, que es un *framework* para el desarrollo de aplicaciones móviles híbridas, que pueden ser compiladas para varios sistemas operativos móviles, el cual usa *Web views* (HTML, CSS y JS), para poder empaquetarlas y compilarse en lenguajes nativos, IONIC también usa AngularJS como su *framework* de JS, con el cual permitirá notablemente con la reutilización de código que se realice en la aplicación Web, para poder usarla en el desarrollo de la aplicación móvil.

Para la base de datos, se ha optado por usar una base de datos No SQL orientada a documentos, por lo cual se ha decidido usar MongoDB que actualmente está en la versión 3.4. Al realizar el desarrollo del API con esta base de datos, se permite aplicar metodologías de desarrollo ágil, ya que no es necesario seguir un esquema de entidad-relación, ni tampoco definir la estructura de llevará el documento. MongoDB también ayuda a entornos que van a requerir escalabilidad.

Para el servidor donde se alojarán los diferentes proyectos se ha decidido usar el sistema operativo Ubuntu en su versión 16.0, que es una distribución Linux.

Para el entorno de desarrollo y para que el equipo de desarrollo cuente con una misma versión del servidor, se usará *Vagrant* y *VirtualBox*, para montar una máquina virtual servidor, el cual tendrá toda el software necesario para que el equipo de desarrollo pueda realizar los diferentes proyectos, en este caso la máquina virtual que se usará es la de *Homestead* que actualmente está en la versión 0.5.0, la cual ya viene instalado todo lo que necesario.

Para llevar una misma versión del proyecto se usará *Git*, que es un programa para manejar el versionamiento, actualmente se encuentra en su versión 2.11.0, el cual ayudará a mantener siempre una sola versión de los proyectos y será más fácil el control de las versiones. El repositorio donde estarán alojados los proyectos será de *Bitbucket*, se muestra el código fuente en dicho repositorio en la figura 2.2.

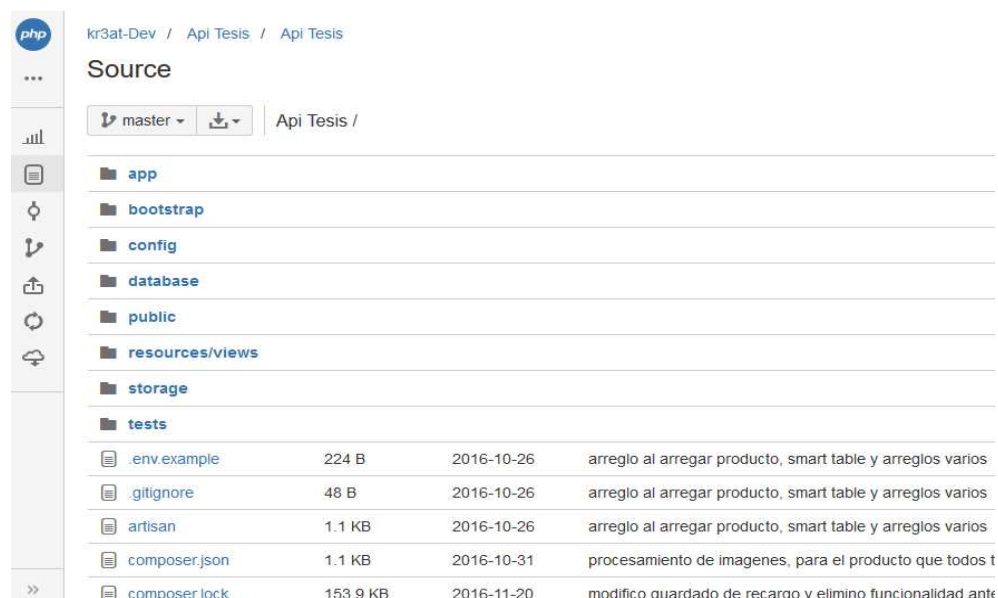


Figura 0.2 Código fuente de API en Bitbucket
Fuente: Autoría Propia

Para la comprobación de los recursos desarrollados en el API se utilizará la herramienta *Postman*, que es una extensión de google *chrome* con la cual podemos probar y realizar diferentes peticiones a las URIs, ya sean del tipo Get, Post, Put, Delete entre otras.

DESARROLLO

1.11.1. Capas del sistema

Las capas que se definen en el sistema son: la persistencia en la cual utilizamos MongoDB, la lógica del negocio la cual manejaremos con el API REST y la vista del cliente para lo cual utilizamos AngularJS tanto en el desarrollo Web como en la aplicación móvil. Una vista clara de esta arquitectura está mostrado en la figura 3.1.

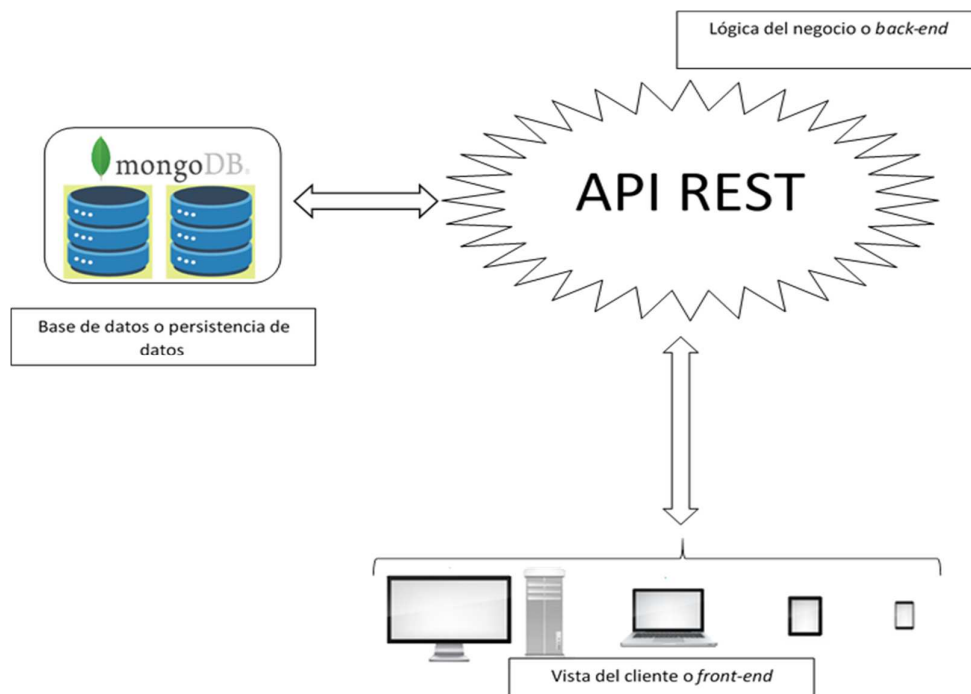


Figura 0.1 Gráfica de las capas del sistema
Fuente: Autoría Propia

1.12. Desarrollo del API REST

Con los requerimientos ya obtenidos por parte de la Empresa Auspiciante ROOSMAN, se tiene que organizar los diferentes *Sprints* que se van a realizar a lo largo del proyecto, por lo tanto se realiza la reunión con el Equipo de Desarrollo y el encargado, para lo cual el Dueño del Producto, ha designado a una persona del Equipo de Desarrollo.

En esta primera reunión se analiza las herramientas a utilizar para el desarrollo y para el control de los *Sprints* de las distintas tareas a realizar en el presente proyecto. Para esto último se ha decidido usar *Trello*, una herramienta *Web* en la cual se puede crear tableros, crear fichas con actividades y su tiempo de duración facilitando el control de tareas y su seguimiento.

Para el control de los *Sprints* en *Trello* se ha creado ya un tablero para el desarrollo del API, como se ve en la figura 3.2 y en la figura 3.3 una vista de tareas para el *Sprint*.



Figura 0.2. Dashboard de Trello
Fuente: Autoría Propia



Figura 0.3 Tareas del Sprint
Fuente: Autoría Propia

En la reunión realizada se ha hecho una estimación de que el proyecto tendrá una duración de 4 meses, por lo tanto se realizarán cuatro *Sprint*, cada uno con una duración de un mes para poder realizar el desarrollo del API con todas las funcionalidades deseadas, esta ha sido una estimación aún muy temprana, por lo que puede estar sujeta a cambios o variaciones, según lo decida el equipo de desarrollo.

1.12.1. Configuración de entorno

Para la instalación del entorno de desarrollo se usará *Homestead* que es una máquina virtual que trae instalado y optimizadas herramientas desarrollo de una aplicación, facilitando el uso de *Git*, bases de datos, gestor de paquetes, entre otros.

Un pre requisitos para usar *Homestead* es tener instalado previamente *VirtualBox*, *Vagrant* y *Git*. A continuación se definen los pasos del proceso de instalación de herramientas para el entorno de desarrollo que se requiere:

1. Instalación de *Virtualbox* versión 5.0
2. Instalación de *Vagrant* versión 1.8
3. Instalación de *Git* versión 2.11
4. Con *Vagrant* instalado se procede con la descarga de *Homestead* versión 0.5.0
5. Configuración del archivo *Homestead.yaml*.
6. Inicialización de la maquina *homestead* con *vagrant*

Ya instalado *Homestead* se debe configurar el archivo *Homestead.yaml*, como se menciona en la lista de pasos anteriores, el cual contiene la configuración para poder sincronizar las carpetas locales con las del servidor virtual y la dirección que van a tener para poder ingresar desde el navegador Web.

Homestead.yaml
<pre> --- ip: "192.168.10.10" memory: 2048 cpus: 1 provider: virtualbox authorize: C:\cygwin\home\pc\.ssh\id_rsa.pub keys: - C:\cygwin\home\pc\.ssh\id_rsa folders: - map: C:\Code to: /home/vagrant/Code sites: - map: ventasapi.com to: /home/vagrant/Code/API-tesis/public - map: roosman.com to: /home/vagrant/Code/ventaropa-Web databases: - homestead </pre>

Figura 0.4. Archivo de configuración Homestead.yaml

Una vez ya terminado de modificar el archivo se ejecutará la máquina virtual con el comando “*vagrant up*”, el cual se encargará de hacer todo para iniciar la máquina virtual y sincronizar los archivos locales con los de la máquina virtual.

1.12.2. Scaffolding del Proyecto

Inicializada la máquina virtual de *Homestead*, hay que ubicarse en la carpeta que está sincronizada con el equipo local y ejecutar el siguiente comandando:

composer create-project laravel/lumen apiVenta --prefer-dist

Esta acción tomará unos minutos a que *composer*, el manejador de paquetes para php, descargue y cree todas las carpetas para tener el proyecto listo. Una vez terminado todo ya se

puede ver en la máquina local todas las carpetas y archivos que se crearon, y está listo para comenzar el desarrollo del API.



Figura 0.5. Scaffolding del Proyecto
Fuente: Autoría Propia

Como se muestra en la figura 3.5, la carpeta app (1) es donde se encuentra la mayoría de los archivos y subcarpetas que usa el *framework* para poder realizar el desarrollo. Los archivos .php que están dentro de la app como: Category.php, DiscountCode.php, DisplayScreen.php, entre otros.

La carpeta *Controllers* (2) es donde están los controladores con sus respectivos métodos, que son los que ayudarán a responder las peticiones que sean solicitadas al API.

La carpeta *Middleware* (3) es donde estarán los archivos *middleware*, que son los que se encargan de filtrar las peticiones HTTP en una aplicación *Web*. En este caso se usará el *middleware* para la autenticación por medio de *tokens*, usando el *plugin* jwt.

El archivo *routes.php* (4), contenida en la carpeta HTTP, es el archivo el cual contiene las rutas o URIs, con el cual se realiza la petición al API, ya que en él están definidas las rutas, el tipo de petición, el controlador y el método que usará cuando sea llamada esa ruta.

Los archivos modelos (5), son lo que contienen la información del documento con el cual se conectará a la base de datos para poder guardar, mostrar o borrar la información.

La carpeta *bootstrap* (5) es donde se encuentra el archivo *App.php*, que es el archivo de configuración inicial para que inicie Lumen, en donde se declaran todos los *middleware*, los alias, y donde se definirá los archivos de *services provider*, que ayudará a hacer uso de los *plugins* instalados.

La carpeta *config* (6) es donde se encontrarán archivos de configuración, los cuales van a ser el archivo para el envío de correo, configuración de la base de datos, autenticación por *tokens* usando jwt, y demás.

La carpeta *databases* (7) se utiliza para poder hacer migraciones, *factories*, los cuales no se usará en este proyecto.

La carpeta *public* (8) es la que contendrá todos los archivos que van a ser públicos del proyecto, en los cuales estarán el *index.php* *.htaccess* y las imágenes.

La carpeta *resources* (9) es donde estarán los recursos que se utilizará para el envío de correos, ya que contendrá las plantillas para el envío de estos.

La carpeta *vendor* (10) es donde se encontrarán los paquetes de terceros o *plugins*, que son necesarios para el correcto funcionamiento del *framework* de Lumen y también donde estarán los *plugins* que se utilizarán, tales como la autenticación de *tokens* (jwt), manejos de consultas de MongoDB, entre otros.

1.12.3. Instalación de JWT

Para poder instalar el paquete de JWT se utilizó información en la Web oficial (<https://jwt.io>), para poder encontrar un paquete que sea compatible con Lumen, obteniendo así el paquete jwt-auth de *tymon designs*, el cual es un paquete de código abierto, que se lo puede encontrar en el repositorio de GitHub y cuenta con una gran documentación.

Una vez descargado con *composer*, se procedió a la configuración de los archivos para la integración con el actual proyecto y contar con una autenticación basado en *tokens*.

1.12.4. Diseño de la Base de Datos

Para el diseño de la base de datos se tiene en cuenta que MongoDB es una base de datos orientada a documentos, por lo cual no es necesario seguir un esquema fijo, este tipo de base de datos tampoco cuenta con los *joins* clásicos del SQL tradicional, para lo cual se va a usar relaciones embebidas o referenciales.

Realizando un análisis de las funcionalidades que se desea presentar en el sistema tanto la parte del administrador como la parte del usuario, se estima que van a ser un total de diez documentos, los cuales se crearán para poder almacenar toda la información y estos serán:

1. Categorías
2. Secciones
3. Productos
4. Impuestos
5. Códigos de descuento

6. Carrito de compras
7. Detalle factura
8. Factura
9. *Setup*
10. Usuario

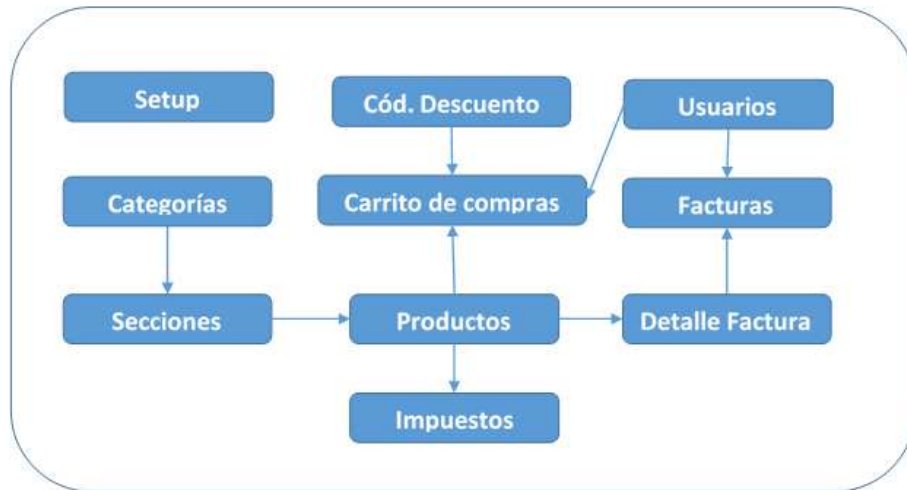


Figura 0.6. Diagrama de base de datos
Fuente: Autoría Propia

Como se puede apreciar en la figura 3.6 se muestran conexiones entre los distintos documentos, estas son similares a relaciones entre tablas en bases relacionales pero en bases NoSql se manejan como referencias.

En base a la estructura que posee la empresa se definen niveles para clasificar los productos de la empresa, por esto se ha clasificado en categorías, secciones y finalmente productos. A continuación se describen los documentos de la base de datos diseñada.

- **Usuarios:** Aquí se define la información de usuarios, tener este registro es indispensable para realizar las compras.

Usuarios	
nombre:	Almacena el nombre de usuario.
apellido:	Apellidos del usuario.
email:	Email del usuario.
password:	Contraseña de la cuenta registrada.
superAdmin:	Al poseer un registro en este campo se presentará el menú de administrador.

Tabla 0.1 Campos del documento Usuario

Fuente: Autoría Propia

- **Setup:** Este documento contiene la configuración de temas relacionados con la empresa.

Setup	
nombre_empresa:	Nombre de la empresa
direccion:	Dirección de la empresa
telefono:	Teléfono de la empresa
nombre_cuenta:	Nombre que se utilizará para realizar los pagos.
num_cuenta:	Número de cuenta en el cual la empresa recibirá los pagos.
Campos para redes sociales:	
facebook:	Enlace que enviará directamente a la página de <i>facebook</i> de la empresa.
instagram:	Se enviará a la página de la empresa publicada en <i>instagram</i> .
Enlaces para descargar la aplicación que se desarrollará posteriormente:	
android_store:	Para mostrar la aplicación en el <i>Play Store</i> .
apple_store:	Para mostrar la aplicación para descarga desde <i>appleStore</i> .
validez_facturas:	Se definirá la validez de las facturas una vez el usuario haya hecho un compromiso de compra.

Tabla 0.2 Campos documento Setup

Fuente: Autoría Propia

- **Categorías:** En este documento se clasifica el tipo de consumidor de los productos de esta forma la categoría puede tomar valores como por ejemplo hombre, mujer, junior, entre otros.

Categoría	
nombre:	Nombre que definirá la categoría.
descripcion:	Almacena una breve descripción de la categoría.
id_seccion:	Arreglo que contiene una referencia al documento de secciones.

Tabla 0.3 Campos documento Categoría

Fuente: Autoría Propia

- **Secciones:** En este documento se clasifica los productos en base al tipo de consumidor de manera agrupada, para nuestro caso los ejemplos serían camisas, pantalones, sacos y demás.

Secciones	
nombre:	Nombre de las distintas secciones
descripcion:	Breve descripción de las distintas secciones.
id_categoria:	Identificador de la categoría que permitirá la referencia desde este documento.
tallas:	Arreglo con objetos que contienen definición de las diferentes tallas para los productos.
productos:	Almacena un arreglo que contiene identificadores del documento de productos el cual permitirá la referencia entre estos documentos.

Tabla 0.4 Campos del documento Secciones
Fuente: Autoría Propia

- **Impuestos:** Contiene los distintos impuestos aplicados a los productos de la empresa:

Impuestos	
nombre:	Nombre del impuesto a aplicar.
descripcion:	Descripción del impuesto.
porcentaje:	Valor del porcentaje que se va agregar al valor inicial del producto.
productos:	Arreglo con los identificadores de los productos que permiten la referencia entre estos documentos.

Tabla 0.5 Campos del documento de Impuestos
Fuente: Autoría Propia

- **Productos:** Documento que contiene los distintos productos que comercializa la empresa.

Productos	
nombre:	Nombre que se da al producto ingresado.
descripcion:	Definición específica del producto.
id_seccion:	Referencia al documento de secciones.

tallas:	Arreglo que define las distintas tallas que puede tener el producto.
colores:	Se almacena un arreglo de objetos que definen colores en base al arreglo de tallas en el cual se guarda un arreglo de colores y cantidad por cada talla ingresada en el arreglo “tallas” presente en este documento.
imagen_path:	Arreglo con las referencias a la locación de las imágenes que se ingresan por producto, se definirá como sugerencia un orden en este arreglo, siendo el orden: Imagen frontal, trasera y varias.
stock:	Número entero que está ligado a la suma de las cantidades de los productos por color definidos en el arreglo colores.
impuestos:	Almacena un arreglo con la referencia a los impuestos que posee el producto.
recargo:	Valor adicional al precio e impuestos del producto según considere el administrador de la empresa, en el análisis se consideró este campo para ingresar valores de envío de productos.
precio:	Precio inicial del producto sin impuestos.
precio_venta:	Valor del producto con impuestos incluidos.
fabricante:	Datos del fabricante del producto.
status:	Estado del producto que se utilizará para el borrado lógico.

Tabla 0.6 Campos del documento Productos

Fuente: Autoría Propia

- **Carrito_compras:** Documento dinámico que crea un único objeto por cliente en el cual se registran los productos que el usuario desea adquirir. Una vez hecha la promesa de compra por parte del cliente se eliminarán los productos ingresados al carrito de compras.

Carrito_compras	
user_id:	Almacena la referencia al usuario al que le pertenece el carrito de compras.
producto:	Arreglo con objetos embebidos de los productos agregados al producto, cada objeto en este arreglo tiene la siguiente estructura:
producto_id:	Campo que referencia al producto seleccionado.
descripcion:	Descripción del producto.
precio:	Almacena el precio bruto del producto.
cantidad:	Cantidad de este producto seleccionado por el cliente.
recargo:	Valor de recargo que está presente en el producto.
subTotalProd:	Precio por cantidad de productos seleccionados.

subTotalRecargo:	Almacena el recargo por cantidad de productos seleccionados.
impuestos:	Arreglo de objetos en el cual posee nombre y porcentaje de impuestos.
talla:	Almacena los objetos de las distintas tallas escogidas por el cliente.
color:	Código de color seleccionado.
imagenPrincipal:	Toma el valor de la locación de la primera imagen ingresada para el producto.
verificarStock:	Bandera que identifica si el producto se encuentra en stock.
disponibles:	Número que representa la disponibilidad de un producto en base al <i>stock</i> y la cantidad seleccionada por el cliente.
no_disponibles:	Cantidad de productos que no se encuentran disponibles en el <i>stock</i> .

*Tabla 0.7 Campos del documento Carrito_compras
Fuente: Autoría Propia*

- **Codigo_descuento:** Documento que define descuentos al valor total de la compra por parte del cliente. Este código tendrá un tiempo de validez y número de usos que será definido por parte del administrador.

Código_descuento	
cod:	Almacena el nombre/razón del descuento.
_id:	Código que se utilizará para agregar el descuento a la compra por parte del cliente.
descuento:	Porcentaje de descuento que se realizará al total de la factura.
caduca:	Fecha de caducidad del código.
estado:	Estado del código, este puede ser “Activo”, “Inactivo” y “Usado”, que permite el control del código de descuento.
limite:	Almacena el límite de usos del código de descuento.
usos:	Número de veces que se ha utilizado ya el código de descuento.

*Tabla 0.8 Campos del documento Codigo_descuento
Fuente: Autoría Propia*

- **Facturas:** Documento que almacena los datos de la compra, como datos de usuario, de valores a pagar, descuentos y estado del envío del producto.

Facturas	
id_shopping_cart:	Almacena la referencia al carrito de compra.

user_id:	Usuario que registró la compra.
Datos de facturación:	Almacenan datos generales para la emitir la factura: nombre, <i>identity_number</i> , direccion, <i>location</i> , telefonoFijo, celular
subtotalProducts:	Suma de precios de productos sin impuestos.
discountValue:	Porcentaje de descuento que se aplica a la factura.
subtotalDiscounts:	Almacena el valor de descuentos a descontar al valor total de la factura.
subtotalsTaxes:	Arreglo que almacena la suma de valores de impuestos dividido por cada uno de los impuestos.
subtotalSurcharges:	Suma de valores de recargo de cada producto.
total:	Total a pagar por parte del cliente.
payment:	Verifica que se haya realizado el pago.
review_product:	Valor que muestra si realizó la promesa de compra.
updated_at:	Fecha de la actualización de la factura.
created_at:	Fecha de creación de la factura.

Tabla 0.9 Campos del documento de Facturas

Fuente: Autoría Propia

- **Detalle_factura:** Documento que almacena los productos por factura generada.

Detalle_factura	
invoice_id:	Referencia a las facturas creadas.
productos:	Datos de productos que son tomados directamente del carrito de compras.

Tabla 0.10 Campo del documento Detalle_factura

Fuente: Autoría Propia

1.12.5. Instalación de drivers para conexión con MongoDB

Una de las consideraciones que se debe realizar al instalar los drivers de MongoDB es que el repositorio oficial proporciona la versión más actualizada y es recomendable realizar la instalación de esta forma, para esto se debe agregar el repositorio oficial al servidor.

1.12.6. Creación de CRUD principales

El primer paso en el desarrollo del API como tal es la creación de los CRUD (“*Create, Read, Update and Delete*”), el acrónimo de "Crear, Leer, Actualizar y Borrar" que son las

funciones básicas de la base de datos. Antes de realizar la creación de estas funciones hay que definir referencias que se van a manejar según la lógica que se definió para el proyecto.

- **Setup:** Realiza funciones leer y actualizar todo el documento.
- **Usuarios:** Crea, lee, actualiza y elimina todos los campos de los usuarios.
- **Categorías:** Realiza todas las funciones básicas y adicional en las mismas se considera la relación con el documento de 'Secciones', para esto se añade al método *delete* la restricción de eliminar solo categorías que no se encuentran referenciados.
- **Secciones:** Se añade a las funciones del CRUD, una restricción en la eliminación similar al anterior, verificando si no tiene productos referenciados en la sección, además de la eliminación de la referencia en la categoría.
- **Impuestos:** Verifica que no existan productos referenciados al momento de eliminar, esto en adición a las funciones básicas del CRUD.
- **Códigos de descuento:** Se manejan las funciones básicas del CRUD, con la particularidad de que el método actualizar va a realizar la verifica del número de usos del código para el cambio de estado a 'USADO' al llegar al límite.
- **Productos:** Se maneja un borrado lógico en el método *delete*, y una verificación para la actualización considerando más o menos impuestos, ya que el sistema soporta más de un impuesto por producto.

1.13. Descripción del proceso de compra

Para realizar la compra se ha definido el siguiente proceso general que se va a definir por medio de un diagrama de flujo como se muestra en la figura 3.7.

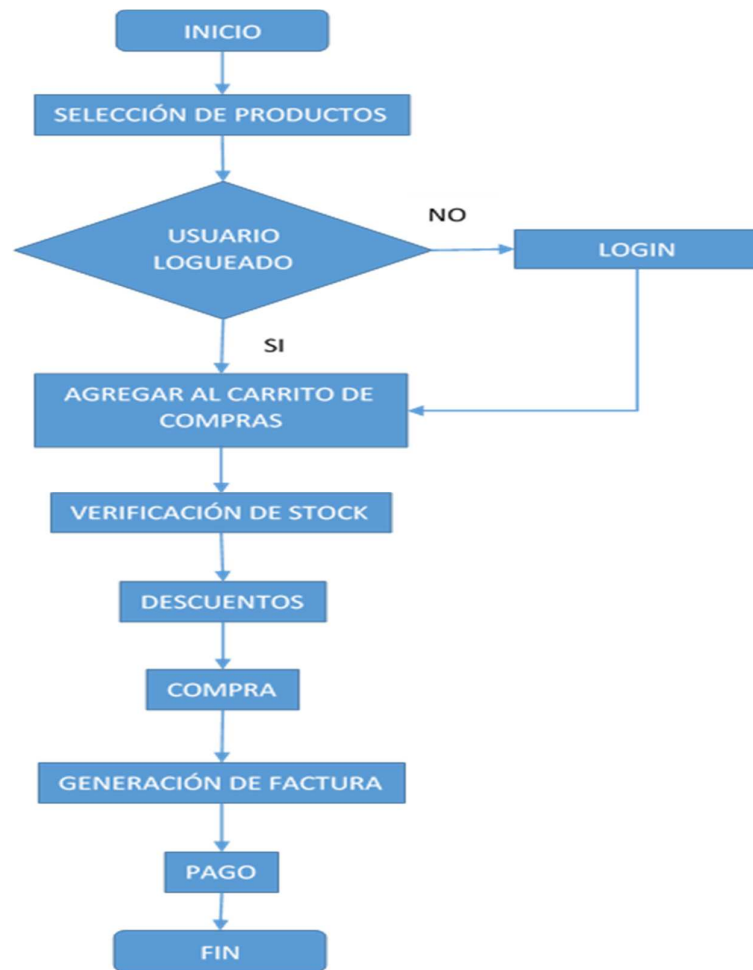
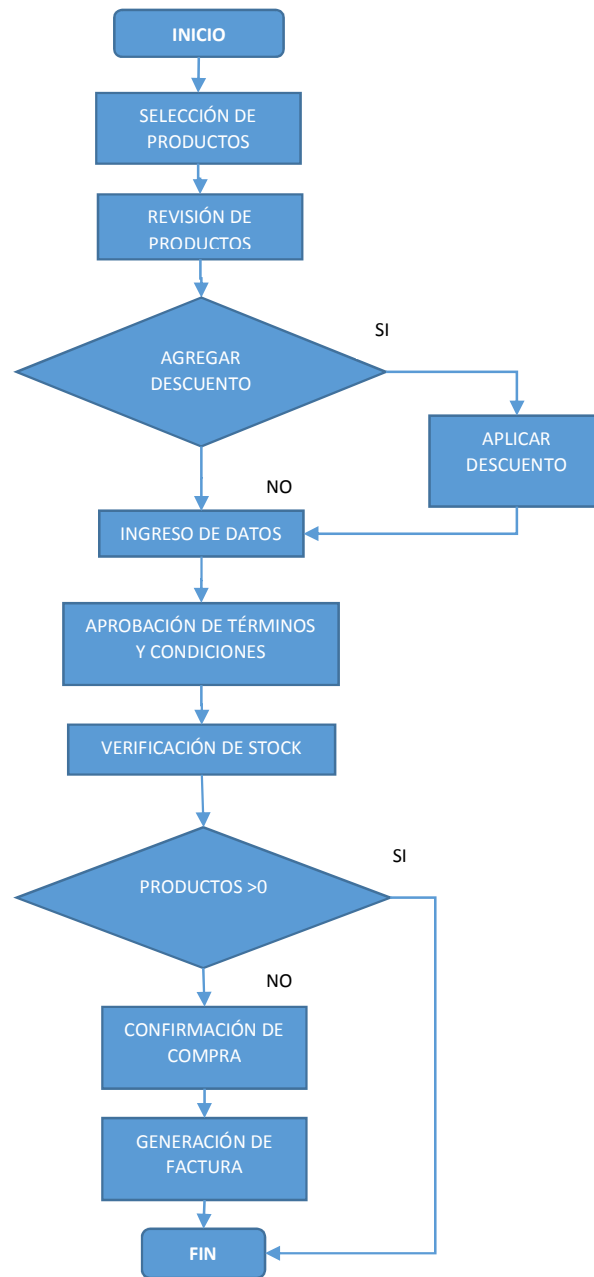


Figura 0.7. Proceso general de compra
Fuente: Autoría Propia

Como se observa en la figura 3.7, si un usuario que no está registrado en el sistema ha ingresado, este usuario puede observar y ver el detalle de todos los productos pero en el momento de agregar el producto al carrito de compras, el sistema pedirá registrar una cuenta nueva o ingresar a una ya existente para poder agregar productos en el carrito.

Una vez el cliente haya ingresado con un usuario registrado empieza el proceso para agregar productos al carrito de compras, al ser este un proceso importante se lo define en la figura 3.8.



*Figura 0.8. Proceso Agregar productos al carrito de compras.
Fuente: Autoría Propia*

Como se muestra en la figura 3.8 una vez agregado los productos el sistema permite el ingreso de códigos de descuento, al mismo tiempo se lleva un control del número de usos del código de descuento y la fecha de caducidad del mismo, a partir de aquí se ingresarán los datos de facturación y se aceptan los términos y condiciones, esto define el tipo de pago y demás

condiciones puestas directamente por la empresa auspiciante como son: devoluciones, precio en envíos, entre otros. A partir de este proceso se genera internamente una pre-factura para almacenar la información de factura como los productos seleccionados.

Luego se realizara la verifica del *stock* de los productos, el cual consiste en comparar la cantidad solicitada por el cliente y la disponible en el sistema, si la cantidad solicitada excede la disponibilidad que se encuentra el sistema, se le mostrará al cliente que el productos no está dispone al momento, pero permitirá continuar la compra si existieran más productos los cuales si se disponga el *stock*.

Al confirmar la compra, se eliminan los productos del carrito de compras y pasan formalmente a una factura, todo este proceso de compra se le informará al usuario vía *email* para que esté al tanto del proceso en que se encuentra su pedido, y se registrará en su perfil la factura generada para que pueda ser revisada en cualquier momento.

Una vez completado el proceso de compra por parte del cliente, el administrador deberá verificar el pago en el sistema, una vez el cliente ha hecho su promesa de compra tendrá un número de días configurado por el administrador, en el cual el cliente debe hacer efectivo el pago, caso contrario se eliminará la factura vencida y los productos regresarán al *stock*.

1.13.1. Tareas programadas (*cron*)

El sistema maneja tareas programadas debido a que se tienen que ejecutar procesos automáticos tomando en cuenta la fecha, para esto se han realizado dos procesos, los cuáles se ejecutarán de manera automática y estos son:

- Caducidad de códigos de descuento.
- Caducidad de facturas.

Como se definió en el apartado anterior los códigos de descuento tienen fecha de caducidad, por esto la tarea programada realiza una comparación de fechas para inhabilitar códigos caducados.

Del mismo modo se realiza una comprobación para las facturas, las cuales serán eliminadas si no se ha realizado el pago por parte del cliente y al mismo tiempo incrementar el *stock* con los productos que pertenecían a facturas eliminadas.

1.14. Desarrollo de la Aplicación Web

Para el desarrollo de la Aplicación Web se usará AngularJS y de igual manera se usará *Trello* para poder realizar el control de los diferentes *Sprints*. Para el desarrollo de la aplicación *Web*, se utilizará el mismo entorno de desarrollo de *Homestead*.

En la primera reunión del equipo de desarrollo se estima que se realizarán dos *Sprints*, cada Sprint con una duración de un mes.

Para el desarrollo de las vistas y los controladores correspondientes a las vistas se define un pequeño proceso el cual se seguirá para hacer el desarrollo de las mismas el cual contará de los siguientes tres pasos:

1. Crear el html de la vista basado en el *wireframe*.
2. Definir la ruta que va a usar y el nombre del controlador que se le asignará
3. Se creará el controlador que se conectará con la vista, que ha sido creado anteriormente en el paso 1.

De tal forma se procederá en los presentes *Sprints* para realizar la vista con su respectivo controlador, ya que el proceso de desarrollo para la aplicación *Web* será repetitivo, el cual trata sobre la crear la vista, definición de su ruta y asignación de su controlador, la única diferencia será entre los controlados de cada vista, ya que algunos serán para listar, o guardar, o crear, o modificar la información; y van a tener que realizar diferentes peticiones al API.

1.14.1. Diseño de los Wireframes

Para el diseño de la aplicación Web primero lo que se realizó fue el diseño de los *wireframes* con los que va a contar la aplicación Web.

Los *wireframes* son bocetos de cómo se desea que luzca la aplicación Web, lo cual ayuda a ver cómo quedará la aplicación, los *wireframes* puede tener pequeñas modificaciones en posiciones de los diferentes componentes, pero siempre tiene que tener semejanza al diseño original. Un ejemplo de esto es el *wireframe* de la pantalla de inicio que vemos en la figura 3.9.



Figura 0.9. Wireframes de la pantalla home
Fuente: Autoría Propia

El resto de los *wireframes* se puede encontrar en anexo B.

1.14.2. Scaffolding del Proyecto Web

Para realizar el *Scaffolding* del Proyecto se utilizó el *seed* de AngularJS, el cual se encuentra disponible en *GitHub*, una vez realizada la descarga se modifica el nombre de la

carpeta y se sincroniza con la máquina virtual servidor por medio de *vagrant*. También se modificó la ubicación de carpetas, como *css*, *img*, *lib* y demás. Se realizó un pequeño cambio en el archivo *bower.json*, para poder guardar los *plugins* de terceros en la carpeta *lib* como se puede observar en la figura 3.10, en la carpeta *app* es donde estarán los diferentes archivos *javascript* (.js).

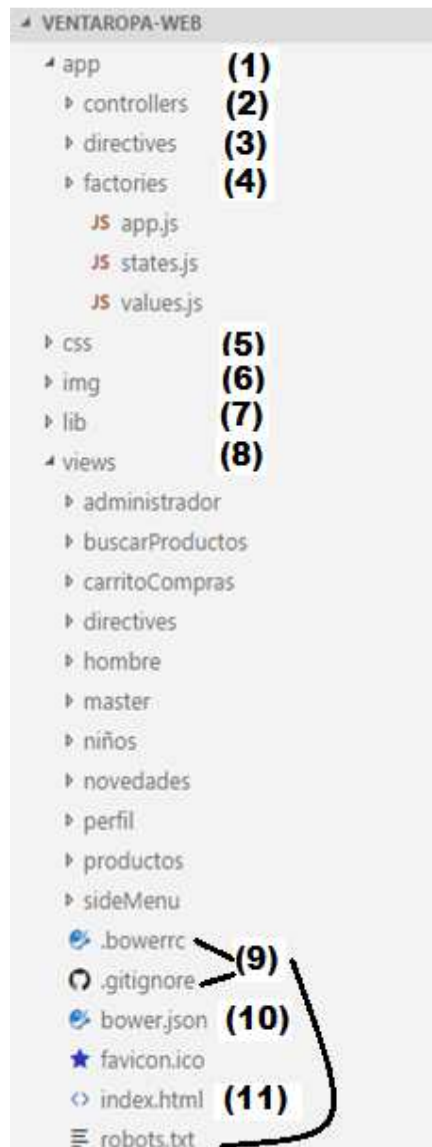


Figura 0.10. Scaffolding del Proyecto Web
Fuente: Autoría Propia

Como se muestra en la figura 3.10, la carpeta *app* (1) es donde se encuentra las subcarpetas donde estarán ubicados los archivos *javascripts*, que ayudaran en el manejo de directivas, controladores y llamadas de métodos. Los archivos *.js* que están dentro de la *app* como: *app.js*, *states.js* y *values.js*, son archivos que nos ayudan en la configuración de angular.

La carpeta *controller* (2) es donde estarán los controladores que se conectaran con las diferentes vistas para poder realizar diferentes acciones y proporcionar información.

La carpeta *directives* (3) es donde estarán las directivas de angular, que según la propia documentación de AngularJS, las directivas son marcadores en un elemento DOM (como un atributo, un nombre de elemento, un comentario o una clase *css*), que indica al compilador HTML de AngularJS para adjuntar el comportamiento especificado a este DOM.

La carpeta *factories* (4) es donde se están empaquetadas todos los recursos del API, para poder realizar las diferentes peticiones para obtener los datos que se encuentran en la base de datos. El archivo *app.js* es donde estarán definidos los diferentes módulos que se van a usar en AngularJS, y los cuales pueden ser propios o *plugins* de terceros, el archivo *states.js* es donde se definirán las rutas de la aplicación *Web*, en donde se asignará la vista y el controlador que va a usar.

La carpeta *css* (5) es donde estarán los archivos de estilos para la aplicación *Web*.

La carpeta *img* (6) es donde estarán las diferentes imágenes que se usarán en la aplicación *Web*.

La carpeta *lib* (7) es donde estarán todos los archivos de los *plugins* instalados.

La carpeta *views* (8) es donde están todos los archivos *html*, los cuales están organizados en diferentes carpetas, que serán usados para las vistas de la aplicación *Web*.

Los ficheros `.bowerrc`, `.gitignore`, `robots.txt` (9) son archivos de configuración que ya vienen creados en el *seed* de AngularJS

EL archivo `bower.json` (10) es donde están todas las dependencias que se usan para la aplicación Web y con el cual podemos actualizar o descargar con el manejador de paquetes *Bower*.

El archivo `index.html` (11) es el archivo principal de la aplicación Web en donde definimos todos los archivos javascript (`.js`) y los archivos de estilos (`.css`) que van a ser utilizados, así como el *body* en donde se cargara todo el contenido manejado por AngularJS.

1.14.3. Instalación de los diferentes *plugins* para usar en el proyecto.

Para el desarrollo de la aplicación se va usar algunos *plugins* que son desarrollados por terceros para poder darle una mayor funcionalidad al proyecto.

Para la instalación de los paquetes se va a usar *bower*, que es un manejador de paquetes para *javascript*.

Los *plugins* a instalarse van a ser:

- AngularJS versión 1.5.8
- *bootstrap* versión 3.3.7
- *angular-resource* versión 1.5.8
- *angular-ui-router* versión 0.3.1
- *angular-animate* versión 1.5.8
- *angular-sanitize* versión 1.5.8
- *angular-touch* versión 1.5.8
- *jquery* versión 3.1.0
- *angular-bootstrap* versión 2.0.1
- *angucomplete-alt* versión 2.4.2

- *angular-loading-bar* versión 0.9.0
- *angular-smart-table* versión 2.1.8
- *pdfmake* versión 0.1.20
- *angular-color-picker* versión 2.7.1

Para lo cual en el archivo `bower.json` se lo va a declarar en la parte de *dependencies*. También se puede encontrar mucha más información de cada *plugin*, sus funciones y su documentación en los repositorios oficiales de cada uno, que se encuentran en *GitHub*.

1.14.4. Creando las rutas que se conectaran al API

En el archivo `values.js`, se coloca la variable global, que contendrá la dirección en donde se encuentra el API.

<i>values.js</i>
<pre>(function () { angular.module('values', []).value('apiUrl', 'http://ventasapi.com/'); })();</pre>

Figura 0.11. Archivo values.js
Fuente: Autoría Propia

- Breve explicación del código

En la figura 3.11, se observa que se define un *values* con el nombre de `apiUrl` y con el valor de `http://ventasapi.com/`, que es la dirección en donde se encuentra alojado el API. Este valor podrá ser accedido por cualquiera de los controladores inyectando el valor de `apiUrl`, para poder ser llamado en cualquier lugar.

Para crear el archivo `js` que contendrá todas las rutas o URIs del API desarrollada, se debe ubicar en la carpeta *factory* y desde esta crear un nuevo archivo que se llamara *methods.js*

<i>methods.js</i>
<pre>(function () { angular.module('factories.methods', []) .factory('Methods', ['\$resource', '\$log', 'apiUrl', '\$timeout', '\$rootScope', '\$http',</pre>

```

function ($resource, $log, apiUrl, $timeout, $rootScope, $http) {
  return {
    flashMessage: function (data, type) {
      $rootScope.flashMessage = {data:data,type:type};
      $timeout(function () {
        $rootScope.flashMessage = {
          data: false,
          type: false
        };
      }, 3000);
    },
    displayScreen: function () {
      return $resource(apiUrl + 'displayScreen/:id', { id: '@id' }, {
        save: {
          method: 'POST',
          headers: {
            "authorization": "Bearer " + localStorage.getItem('token'),
          }
        },
        get: {
          method: 'GET',
          headers: {
            "authorization": "Bearer " + localStorage.getItem('token'),
          }
        },
        update: {
          method: 'PUT',
          headers: {
            "authorization": "Bearer " + localStorage.getItem('token'),
          }
        },
        query: {
          method: 'GET',
          headers: {
            "authorization": "Bearer " + localStorage.getItem('token'),
          },
          isArray: true
        },
        delete: {
          method: 'DELETE',
          headers: {
            "authorization": "Bearer " + localStorage.getItem('token'),
          }
        }
      });
    },
  },

```

```
};
});
})();
```

Figura 0.12. Archivo methods.js
Fuente: Autoría Propia

- Explicación del código del archivo methods.js

Como se puede ver en la figura 3.12, se define el nombre *methods*, para que desde cualquier controlador del proyecto se puede acceder a los métodos que tiene este archivo.

En este archivo se escribe el nombre del método y la función que va a realizar, aquí se usa la variable global *apiUrl* y se le concatena el resto de la dirección del recurso que se desea obtener, también se define si la url va a necesitar algún parámetro como puede ser un id.

También se definen los métodos que van a poder ser usados, como se observa en la figura 3.10, el método *save*, corresponde a una petición *post* que se hará al servidor para lo cual se agrega una cabecera a la petición, que contiene el *token* que está almacenado en la memoria del navegador, dicho *token* será usado en el servidor para poder comprobar si el usuario tiene permiso para ingresar a ese recurso o Uri del API.

1.14.5. Realización de la pantalla de inicio

Para la realización de la pantalla de inicio, se usará el proceso que fue definido al principio del capítulo.

1.- Se crea el archivo html que contendrá la vista de la pantalla de inicio, la cual se muestra en la figura 3.13

```
home.html
<div class="row">
  <div class="banner">
    <a href="#/{{pantallaHome.banner1.nombreCat}}/{{pantallaHome.banner1.nombre}}"
      ng-if="pantallaHome.banner1.tipo == 'seccion'">
      
```

```

</a>
<a href="#/{{pantallaHome.banner1.nombre}}"
  ng-if="pantallaHome.banner1.tipo == 'categoria'">
  
</a>
</div>
</div>
<br>
<div class="row">
  <div class="col-lg-4 col-md-4 col-sm-4">
    <div class="banner">
      <a href="#/{{pantallaHome.display1.nombreCat}}/{{pantallaHome.display1.nombre}}"
        ng-if="pantallaHome.display1.tipo == 'seccion'">
        
      </a>
      <a href="#/{{pantallaHome.display1.nombre}}"
        ng-if="pantallaHome.display1.tipo == 'categoria'">
        
      </a>
    </div>
  </div>
  <div class="col-lg-4 col-md-4 col-sm-4">
    <div class="banner">
      <a href="#/{{pantallaHome.display2.nombreCat}}/{{pantallaHome.display2.nombre}}"
        ng-if="pantallaHome.display2.tipo == 'seccion'">
        
      </a>
      <a href="#/{{pantallaHome.display2.nombre}}"
        ng-if="pantallaHome.display2.tipo == 'categoria'">
        
      </a>
    </div>
  </div>
  <div class="col-lg-4 col-md-4 col-sm-4">
    <div class="banner">
      <a href="#/{{pantallaHome.display3.nombreCat}}/{{pantallaHome.display3.nombre}}"
        ng-if="pantallaHome.display3.tipo == 'seccion'">
        
      </a>
      <a href="#/{{pantallaHome.display3.nombre}}"
        ng-if="pantallaHome.display3.tipo == 'categoria'">
        
      </a>
    </div>
  </div>
</div>

```

Figura 0.13. Archivo home.html, que será usado para la pantalla de inicio

Fuente: Autoría Propia

2.- Se define la ruta que usará la vista y el nombre del controlador que será usada para pasar datos a la vista.

La definición de la ruta y el nombre del controlador se la hará en el archivo states.js definido en la figura 3.14.

```
states.js
(function () {
  angular.module('states', [])
    .config(['$stateProvider', '$urlRouterProvider',
      function ($stateProvider, $urlRouterProvider) {
        $urlRouterProvider.otherwise('/home');
        $stateProvider
          .state('m', { // master
            abstract: true,
            templateUrl: 'views/master/master.html',
            controller: 'MasterCtrl as m'
          })
          .state('m.home', {
            url: "/home",
            views: {
              'content': {
                templateUrl: 'views/master/home.html',
                controller: "HomeCtrl"
              }
            }
          })
      })
    });
})();
```

Figura 0.14. Archivo states.js

Fuente: Autoría Propia

Este archivo es el que manejará las rutas de la aplicación Web, aquí se define la url que va a tener, el archivo html que va a usar, el cual ya se ha definido en la figura 3.13, la url y el controlador que va a usar la vista para poder mostrar los datos y realizar las acciones que estén programadas en la vista.

3.- Creación del controlador que se conectará con la vista

Aquí se creará el controlador Home Ctrl, que será el que se conecte con la vista, pero a su vez también podrá realizar peticiones al API.

```
home.js
(function () {
  angular.module('controllers.home', [])
    .controller('HomeCtrl', ['$scope', '$log', 'Methods', function ($scope, $log, Methods) {
      $scope.pantallaHome = {};
      Methods.displayScreen().query().$promise.then(
        function (data) {
          if(data){
            $scope.pantallaHome = data[0];
          }
        },
        function (error) {
          console.log(error);
        }
      );
    }]);
})();
```

Figura 0.15. home.js
Fuente: Autoría Propia

En el HomeCtrl se puede ver en la figura 3.15 se usa el método *displayScreen*, que se encuentra definido en la figura 3.12, el cual va a realizar una petición al API y con la respuesta de la petición se mostrara los datos en pantalla.

1.14.6. Creación del resto de pantallas basado en los *wireframes*

Para la creación del resto de las pantallas que aún se tiene que realizar en la aplicación *Web*, se usará el mismo proceso que ya se definió antes, y del cual se puede observar un ejemplo en la sección anterior de cómo usar el proceso.

1.15. Desarrollo de la aplicación móvil

Para el desarrollo de la aplicación móvil se va a reutilizar código que ya fue desarrollado en la aplicación *Web*, ya que IONIC usa el mismo *Framework* de *JavaScript* que es AngularJS.

Se usará algunos archivos como: *Methods*, *values* y algunos controladores.

IONIC nos proporciona también una vasta documentación, componentes html y css, funciones de *JavaScript*.

Igual que en los proyectos anteriores se usará *Trello* para el control de los diferentes *Sprints* que se puedan realizar en el desarrollo de la aplicación móvil. Para mostrar esto se define un pequeño proceso para la realización de las pantallas o vistas de la aplicación móvil.

1. Crear el html de la vista basado en el *wireframe*.
2. Definir la ruta que va a usar y el nombre del controlador que se le asignará.
3. Se creará el controlador que se conectará con la vista, que ha sido creado anteriormente en el paso 1.

1.15.1. Diseño de los *wireframe* de la aplicación móvil

Para el diseño de la aplicación móvil primero lo que se realizó fué el diseño de los *wireframes*. Estos son bocetos de cómo se desea que luzca la aplicación móvil, lo cual ayuda a una mejor visualización de cómo quedará la aplicación, los *wireframes* puede tener pequeñas modificaciones en posiciones de los diferentes componentes, pero siempre tiene que tener semejanza al diseño original como se muestra en la figura 3.16. El resto de los *wireframes* de la aplicación móvil se puede encontrar en anexo C.



Figura 0.16. Wireframe de la pantalla de login
Fuente: Autoría Propia

1.15.2. Scaffolding del Proyecto Web

Para realizar el *scaffolding* del proyecto se usó el CLI de IONIC, que son los comandos que IONIC proporciona. En la carpeta del proyecto se inicia el nuevo proyecto ingresando el comando:

“ionic start ventaMovil tabs”

Con el comando anterior el CLI de IONIC comienza con la creación de las diferentes carpetas necesarias y realiza la instalación de todos los componentes necesarios para poder iniciar el proyecto, solo toca esperar a que todo se instale correctamente y comenzar la configuración para comenzar con el desarrollo.

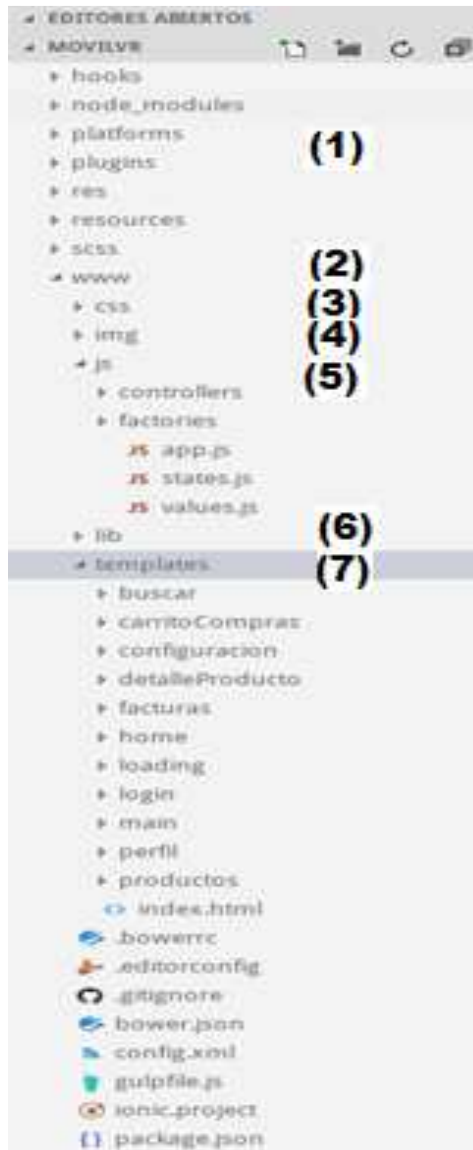


Figura 0.17. Scaffolding del proyecto móvil
Fuente: Autoría Propia

Como se puede observar en la figura 3.17, son los archivos y carpetas que fueron creados por el CLI de IONIC.

Las carpetas (1) *hooks*, *node_modules*, *platforms*, *plugins*, *resources*, *scss*. Son carpetas que usa IONIC al momento de hacer la compilación de los *Web views* a código nativo, ya sea para *Android* o *IOS*.

La carpeta sobre la cual se trabaja es *www* (2), la cual contiene los *css*, *img*, *js*, *lib*, *templates* y el archivo *index.html*.

La carpeta *css* (3) es la que tiene el archivo de estilo *css*.

La carpeta *img* (4) es donde están almacenadas imágenes que usa la aplicación móvil.

La carpeta *js* (5) es donde estarán los archivos *js*, que al igual que en el proyecto *Web*, serán los controladores, archivos de rutas y variables globales.

La carpeta *lib* (6) es donde estarán *plugins* que se instalaron.

La carpeta *templates* (7) es donde estarán los archivos *html*, que serán las vistas de la aplicación móvil.

1.15.3. Instalación de los diferentes *plugins* para usar en el proyecto móvil

Al igual que el proyecto *Web*, se van a usar algunos *plugins* que con desarrollados por terceros para poder darle una mayor funcionalidad al proyecto.

Para la instalación de los paquetes se va a usar *bower*, que es un manejador de paquetes para *javascript*

Los *plugins* a instalar van a ser:

- *AngularJS-resource* versión 1.5.8
- *AngularJS-messages* versión 1.5.8
- *Angucomplete-alt* versión 2.5.0
- *Pdfmake* versión 0.1.20
- *ngCordova* versión 0.1.27-alpha

En el archivo `bower.json` se los va a declarar en la parte de *dependencies*. También se puede encontrar mucha más información de cada *plugin*, sus funciones y su documentación en los repositorios oficiales de cada uno, que se encuentran en *GitHub*.

1.15.4. Creación de las rutas que se conectaran al API

Al igual que en el proyecto *Web*, se creará un archivo que contendrá todas las rutas o Uris, las cuales servirán para poder conectarnos al API y realizar las diferentes peticiones.

En esta parte vamos a reutilizar el código que ya fue desarrollado en la aplicación *Web*.

<i>values.js</i>
<pre>(function () { angular.module('values', []).value('apiUrl', 'http://ventasapi.com/'); })();</pre>

Figura 0.18. Archivo values.js con la variable global apiUrl.

Fuente: Autoría Propia

- Breve explicación del código

En la figura 3.18, se observa que se define un *values* con el nombre de *apiUrl* y con el valor de `http://ventasapi.com/`, que es la dirección en donde se encuentra alojado el API. Este valor podrá ser accedido por cualquiera de los controladores inyectando el valor de *apiUrl*, para poder ser llamado en cualquier lugar.

Para crear el archivo `.js` que contendrá todas las rutas o Uris del API desarrollada, se debe ubicar en la carpeta *factory* y desde esta crear un nuevo archivo que se llamara *methods.js*

<i>methods.js</i>
<pre>(function () { angular.module('factories.methods', []) .factory('Methods', ['\$resource', '\$log', 'apiUrl', '\$timeout', '\$rootScope', '\$http', function (\$resource, \$log, apiUrl, \$timeout, \$rootScope, \$http) { login: function (user) { return \$resource(apiUrl + 'auth/login').save(user).\$promise; }, register: function (user) {</pre>

```

        return $resource(apiUrl + 'register').save(user).$promise;
    },
    productMobil : function () {
        return $resource(apiUrl + 'mobilProduct/:id',{id:'@id'},{
            query: {
                method: 'GET',
                headers: {
                    "authorization": "Bearer " + localStorage.getItem('token'),
                },
                isArray: true
            }
        });
    },
    productByCategoryMobile: function () {
        return $resource(apiUrl + 'mobileProductCategory/:id', { id: '@id' }, {
            get: {
                method: 'GET',
                headers: {
                    "authorization": "Bearer " + localStorage.getItem('token'),
                },
                isArray: true
            },
        },
    )
    },
};
})();

```

Figura 0.19. Archivo methods.js
Fuente: Autoría Propia

- Explicación del código del archivo methods.js

Como se puede ver en la figura 3.19, se define el nombre *methods*, para que desde cualquier controlador del proyecto, se puede acceder a los métodos que tiene este archivo.

En este archivo se escribe el nombre del método y la función que va a realizar, aquí se usa la variable global *apiUrl* y se le concatena el resto de la dirección del recurso que se desea obtener, también se define si la url va a necesitar algún parámetro como puede ser un id.

También se definen los métodos que van a poder ser usados, como se observa en la figura 3.10, el método *get*, corresponde a una petición *get* que se hará al servidor para lo cual se agrega una cabecera a la petición, que contiene el *token* que está almacenado en la memoria del navegador, dicho *token* será usado en el servidor para poder comprobar si el usuario tiene permiso para ingresar a ese recurso o Uri del API.

1.15.5. Creación de la pantalla de inicio para la aplicación móvil.

Para la realización de la pantalla de inicio se usará el pequeño proceso que se definió anteriormente.

1.- se crea primero el archivo html de la vista según el *wireframe* definido en la figura 3.20.

```
Login.html
<div class="loginLayouy">
  
  <div class="padding">
    <span class="input-label">Correo Electronico</span>
    <input class="loginInput" type="email" placeholder="Correo Electronico"
      ng-model="userLogin.email" required>
    <br>
    <span class="input-label">Password</span>
    <input class="loginInput" type="password" placeholder="Password"
      ng-model="userLogin.password" required>
    <br>
    <button class="button button-block button-stable"
      ng click="login(userLogin)">
      Login
    </button>
    <button class="button button-block button-stable"
      ng-click="modal.show()">
      ¿No tienes una cuenta? Registrate Aqui!
    </button>
  </div>
</div>
```

Figura 0.20. login.html
Fuente: Autoría Propia

2.- Se define la ruta que usará la vista y el nombre del controlador que será usada para pasar datos a la vista.

La definición de la ruta y el nombre del controlador se la hará en el archivo states.js que se muestra a continuación en la figura 3.21.

```
states.js
(function () {
  angular.module('states', [])
    .config(['$stateProvider', '$urlRouterProvider',
      function ($stateProvider, $urlRouterProvider) {
        $urlRouterProvider.otherwise('/home');
        $stateProvider
          .state('login',{
            url: '/login',
            templateUrl: 'templates/login/login.html',
            controller: 'LoginCtrl'
          })
          .state('salir',{
            url: '/salir',
            controller: 'LogOutCtrl'
          })
          .state('m', {
            abstract: true,
            templateUrl: 'templates/main/main.html',
            controller: 'MainCtrl'
          })
          .state('m.home', {
            url: '/navegar',
            views: {
              'menuContent': {
                templateUrl: 'templates/home/home.html',
                controller: 'MainCtrl'
              }
            }
          })
      }
    ]);
})();
```

Figura 0.21. states.js
Fuente: Autoría Propia

Este archivo es el que manejará las rutas de la aplicación, aquí se define la url que va a tener, el archivo html, que en este caso ya fue creado como se puede observar en la figura 3.20, la url y el controlador que va a usar la vista para poder mostrar los datos y realizar las acciones que estén programadas en la vista.

3.- Creación del controlador que se conectará con la vista

Aquí se creará el controlador LoginCtrl definido en la figura 3.22, que será el que se conecte con la vista, pero a su vez también podrá realizar peticiones al API.

```
login.js
(function () {
  angular.module('starter.controllers.login', [])
    .controller('LoginCtrl',
      function ($scope, $ionicPopup, $ionicLoading, Methods, $http, $state, $timeout,
        $ionicModal) {
        $scope.user = angular.fromJson(localStorage.getItem('user'));
        if ($scope.user !== null) {
          $state.go('m.home');
        }

        $scope.userLogin = {};
        $scope.login = function (userLogin) {
          Methods.login(userLogin).then(function (data) {
            if (data.error) {
              $scope.errors = [];
              $scope.errors.push("Usuario o Password incorrectos");
              var alertPopup = $ionicPopup.alert({
                title: 'Error ',
                template: $scope.errors
              });
            } else {
              $ionicLoading.show({
                template: 'Loading...',
                duration: 400
              }).then(function () {
                localStorage.setItem('token', JSON.stringify(data.success.token));
                localStorage.setItem('user', JSON.stringify(data.success.user));
                $state.go('m.home');
                $scope.userLogin = {};
              });
            }
          });
        }
      }
    );
  });
```

```

    }
    }, function (error) {
        console.log(error);
    });
}
$scope.registration = {};
$ionicModal.fromTemplateUrl('templates/login/register.html', {
    scope: $scope,
    animation: 'slide-in-up'
}).then(function (modal) {
    $scope.modal = modal;
});
$scope.openModal = function () {
    $scope.modal.show();
};
$scope.closeModal = function () {
    $scope.modal.hide();
};
$scope.$on('$destroy', function () {
    $scope.modal.remove();
});
$scope.register = function () {
    Methods.register($scope.registration)
        .then(function (data) {
            if (data.errors) {
                $scope.registrationErrors = [];
                $scope.registrationMessages = {};
                angular.forEach(data.errors, function (value, key) {
                    $scope.registrationMessages[key] = value;
                });
                var alertPopup = $ionicPopup.alert({
                    title: 'Error ',
                    template: "Error los datos proporcionados son incorrectos"
                });
            } else {
                console.log(data.message.length);
                if (data.message.length == 68) {
                    var alertPopup = $ionicPopup.alert({
                        title: 'Error ',
                        template: data.message
                    });
                } else {
                    $scope.registration = {}
                    $scope.modal.hide();
                }
            }
        })
    }
}

```



```
        console.log(data)
    }, function (err) {
        innerScope.registrationErrors = [];
        innerScope.registrationErrors.push(err);
    });
};
});
})();
```

Figura 0.22. login.js
Fuente: Autoría Propia

Este controlador es el que se encargará de hacer la autenticación de los usuarios, para eso usará el método *login*, el cual ya fue definido como se puede observar en la figura 3.17 *methods.js*, que es donde están las rutas o Uris del API.

1.15.6. Creación del resto de pantallas basado en los *wireframes*

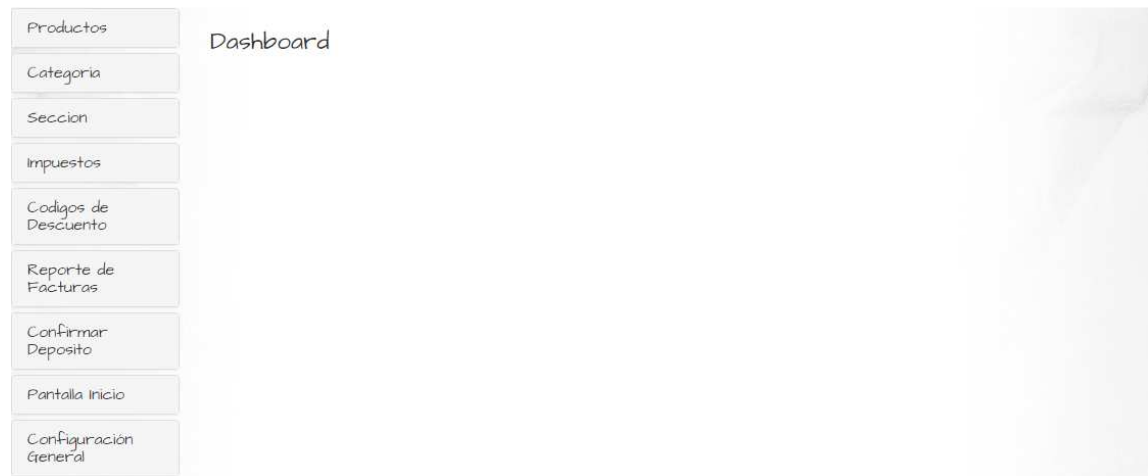
Para la creación del resto de las pantallas que aún se tiene que realizar en la aplicación móvil se usará el mismo proceso que ya se definió antes, tanto en el desarrollo *Web* como móvil, y del cual se puede observar un ejemplo en la sección anterior de cómo usar el proceso.

RESULTADOS

Los resultados obtenidos de la implementación del sistema desarrollado está dividido entre el perfil ‘Administrador’ y el perfil ‘Cliente’, en los cuales se detallan los beneficios obtenidos para la empresa auspiciante.

1.16. Perfil administrador

Una vez se realiza la autenticación como administrador se presenta en el menú una opción llamada ‘Administrador’ que no se mostrará para el perfil de ‘Cliente’, esta permite una serie de acciones que muestran en la Figura 4.1.



*Figura 0.1. Dashboard de Administrador
Fuente: Autoría Propia*

1.16.1. Manejo de catálogos

El usuario administrador es quien alimentará inicialmente la base de datos y será el responsable de manipular la información de los productos que se mostrará en el perfil ‘Cliente’, para esto el sistema presenta las facilidades de ingreso, edición y borrado de esta información a través de las siguientes interfaces:

- **Categorías:** Este es el primer catálogo a ingresar ya que a partir de este se crearán las secciones y luego los productos.

Lista de Categorías

Nueva Categoría

Nombre	Descripción	Acciones
Hombre	Ropa de hombre	 
Mujer	Ropa de mujer	 
Junior	Ropa de niño	 

Figura 0.2. Lista de Categorías

Fuente: Autoría Propia

- **Secciones:** Esta interface muestra el listado de secciones referenciados a el catalogo descrito anteriormente.

Lista de Secciones

Nueva Sección


Categoría	Nombre	Descripción	Acciones
Hombre	Pantalones	Pantalones	 
Hombre	Camisas	Camisas	 
Mujer	Pantalones	Pantalones	 
Mujer	Blusas	Blusas	 

Figura 0.3 Lista de Secciones

Fuente: Autoría Propia

- **Impuestos:** Desde esta interface se maneja el catálogo de los impuestos que serán aplicadas a los productos, como en el Ecuador se maneja el IVA que por el momento es de 14%, este es el único impuesto ingresado.

Lista de Impuestos

Nuevo Impuestos			
Nombre	Descripción	Porcentaje	Acciones
IVA	Impuesto al valor agregado	14	 

Figura 0.4. Lista de Impuestos

Fuente: Autoría Propia

- **Productos:** En esta lista se manejará el ingreso, edición y borrado lógico de productos en el sistema, ya que es el catálogo más extenso, tenemos la opción adicional de realizar búsquedas.

Lista de Productos



















Nuevo Producto					
1 2 3					
Buscar _					
Categoría	Sección	Nombre	Descripción	Estado	Acciones
Hombre	Pantalones	Pantalon Blanco	Pantalon Blanco semi-tubo, tela stretch	ACTIVE	 
Hombre	Pantalones	Pantalon medio tono	Pantalon de corte recto con el color envejecido	ACTIVE	 
Hombre	Pantalones	Pantalon Negro	Pantalon Negro semi-tubo con tela stretch	ACTIVE	 
Hombre	Pantalones	Pantalon Claro	Pantalon de tono claro corte semi-tubo en tela stretch	ACTIVE	 
Hombre	Pantalones	Pantalon Nudy	Pantalon de bota tubo con tela stretch semi desgastado	ACTIVE	 
Hombre	Pantalones	Pantalon con desgastes	Pantalon con desgastes semi-tubo con tela confort	ACTIVE	 
Hombre	Pantalones	Pantalon skinny	Pantalon skinny con rotos tela confort	ACTIVE	 
Hombre	Pantalones	Pantalon Nudy corte tubo	Pantalon Nudy corte tubo oscuro con tela confort	ACTIVE	 
Hombre	Pantalones	Pantalon oscuro parche	Pantalon oscuro con parche, con saltos de pintura, corte semi-tubo, tela confort	ACTIVE	 

Figura 0.5. Lista de Productos

Fuente: Autoría Propia

- **Códigos de descuento:** Desde esta interface se manejarán los descuentos que recibirán los usuarios y que se aplicará en las compras realizadas por los mismos.



Código	Nombre	% de descuento	Fecha de caducidad	Estado	Acciones
Navidad	Navidad	25	2016-12-30	Activo	 
007rm	Inauguración	20	2017-02-05	Activo	 
001rm	saldo enero	10	2017-02-05	Activo	 

Figura 0.6. Lista de Códigos de Descuentos
Fuente: Autoría Propia

1.16.2. Reporte de facturas

Desde esta interface se podrá consultar una la lista de facturas por fecha, así también por usuario en un intervalo de fechas, adicionalmente al seleccionar la factura buscada se puede visualizar el detalle de la misma en formato PDF.

FACTURAS POR FECHA

Fecha inicio

Fecha fin

Factura	Cliente	Fecha	Valor
58900b694a8bac33060c447	Jairo Adrian Serrano Parreño	2017-01-31 03:59:10	178.99
58900d924a8bac070d22c19e	Ximena Denisse Aguirre	2017-01-31 04:09:13	18.00
58900e144a8bac33060c449	Jairo Adrian Serrano Parreño	2017-01-31 04:10:04	26.74
5893de494a8bac070d22c1a2	Jairo Adrian Serrano Parreño	2017-02-03 01:35:40	84.00
58947ab54a8bac53380eb8fe	Rodolfo	2017-02-03 12:54:01	151.99

Figura 0.7. Reporte de Facturas
Fuente: Autoría Propia

1.16.3. Configuración general

Desde esta interface se editará la información general de la empresa, como la dirección, teléfono, redes sociales, entre otros.

Configuración


Nombre de la empresa	Dirección
<input type="text" value="ROOSMAN"/>	<input type="text" value="C.C. Nuevo Amanecer"/>
Teléfono	Ruc
<input type="text" value="2954338"/>	<input type="text" value="171723929001"/>
Email	Nombre de cuenta
<input type="text" value="roosmacia@gmail.com"/>	<input type="text" value="roosman"/>
Número de cuenta	Validez Facturas
<input type="text" value="3020298404"/>	<input type="text" value="4"/>
Facebook	Instagram
<input type="text" value="http://www.facebook.com"/>	<input type="text" value="http://www.instagram.com"/>
Android Store	Apple Store
<input type="text" value="http://www.playstore.com"/>	<input type="text" value="http://www.macstore.com"/>
Subir logo de la Empresa	Logo de la Empresa
<div>Subir Imagen</div> <div>Drop Files here to upload</div>	
<input type="button" value="Guardar"/>	

Figura 0.8. Parámetros de Configuración
Fuente: Autoría Propia

1.17. Perfil Cliente

Esta será la interface que se presentará al usuario final, desde aquí se realizará la navegación a través de los productos existentes, la compra de los mismos, revisión de las facturas generadas y configuración del perfil.

1.17.1. Navegación y selección de productos

Al seleccionar una sección se listaran los productos que se encuentran disponibles en aquella sección, como se puede observar en la Figura 4.9, y podrá seleccionar el producto que desee.

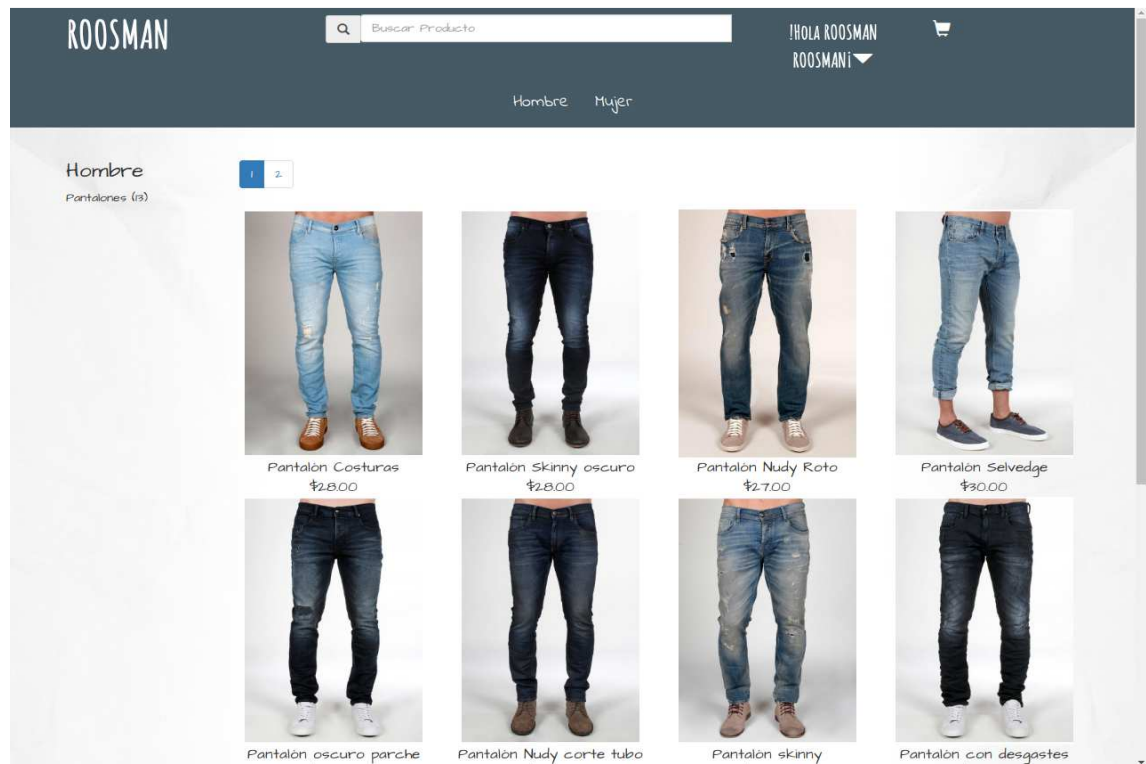


Figura 0.9 Lista de productos existentes (sección pantalones de hombre)
Fuente: Autoría Propia

Al seleccionar un producto se abrirá la ventana que se muestra en la Figura 4.9 y 4.10 correspondientes a las categorías de hombre y mujer, en la cual podrá seleccionar la talla y el color que desee, para poder agregar el producto al carrito de compras.



Figura 0.10 Detalle producto (categoría hombre)
Fuente: Autoría Propia



Figura 0.11. Detalle producto (categoría mujer)
Fuente: Autoría Propia

En la aplicación móvil el listado de productos se lo visualiza como se muestra en la Figura 4.12, en el cual también puede ingresar a las diferentes categorías y secciones que se poseen.



Figura 0.12 Listado de productos en la aplicación móvil
Fuente: Autoría Propia

Al seleccionar sobre el producto que se desea, se mostrara la pantalla que se puede observar en la Figura 4.13, la cual es el detalle del producto, aquí podrá seleccionar la talla y el color deseado para agregar el producto al carrito de compras.



Figura 0.13 Detalle producto en la aplicación móvil
Fuente: Autoría Propia

1.17.2. Carrito de compras

En esta interface se mostrará una lista de los productos seleccionados y el detalle del valor a pagar en la factura.

The screenshot shows a web shopping cart interface. At the top, it says 'Carrito de Compras'. Below this, there's a table with columns: 'Producto', 'Precio unitario', 'Envío x producto', 'Cantidad', 'SubTotal Producto', and 'SubTotal Envío'. A single product is listed: 'Pantalón con costuras en el bolsillo, corte semi-tubo, tela stretch | 28 | color [blue dot]'. Its unit price is \$24.56, shipping is 2, quantity is 1 (in a dropdown), subtotal is \$24.56, and shipping subtotal is \$2.00. Below the table, there's a section 'Agregar Código de Descuento'. To the right, a 'Resumen del Pedido' (Order Summary) box shows: Subtotal Productos: 24.56, Descuento: 2.96, IVA (4%): 3.44, Subtotal Envío: 2.00, and Total: \$27.54. A note below says 'Usted ha usado un código de Descuento'. A 'Comprar' button is at the bottom right.

Producto	Precio unitario	Envío x producto	Cantidad	SubTotal Producto	SubTotal Envío
Pantalón con costuras en el bolsillo, corte semi-tubo, tela stretch 28 color ●	\$24.56	2	1	\$24.56	\$2.00

Agregar Código de Descuento

Resumen del Pedido

Subtotal Productos	24.56
Descuento	2.96
IVA (4%)	3.44
Subtotal Envío	2.00
Total	\$27.54

Usted ha usado un código de Descuento

Comprar

Figura 0.14. Carrito de compras
Fuente: Autoría Propia

Por otro lado en la aplicación móvil, el carrito de compras se muestra cómo se puede observar en la Figura 4.15

The screenshot shows a mobile shopping cart application interface. At the top, there's a hamburger menu icon and the title 'Carrito Compras'. Below this, a 'Resumen del Pedido' (Order Summary) box shows: Subtotal Productos: 24.56, IVA (4%): 3.44, Subtotal Envío: 2.00, and Total: \$30.00. A 'Comprar' button is below the summary. Below the summary, there's a section 'Agregar Código de Descuento'. At the bottom, a product card shows a photo of black pants, description 'Pantalon Negro semi-tubo con tela stretch', price '\$24.56', shipping '\$2.00', size 'Talla: 30', color (black swatch), and quantity '1' (in a dropdown).

Carrito Compras

Resumen del Pedido

Subtotal Productos	24.56
IVA (4%)	3.44
Subtotal Envío	2.00
Total	\$30.00

Comprar

Agregar Código de Descuento



Descripcion Producto : Pantalon Negro semi-tubo con tela stretch
Precio: \$24.56
Recargo: \$2.00
Talla: 30
Color:
Cantidad: 1

Figura 0.15 Carrito de compra aplicación móvil
Fuente: Autoría Propia

1.17.3. Correo de compra

Al realizar la compra se realiza un envío de correo que contendrá el detalle de compra una vez se ha realizado la compra



Figura 0.16. Mail con información sobre factura generada
Fuente: Autoría Propia

1.17.4. Historial de Pagos

Desde esta interface el cliente puede obtener una lista de todas las compras realizadas, con la opción de filtrarlas por el campo que el considere.

FACTURAS GENERADAS			
Buscar _			
Factura	Cliente	Fecha	Valor
58947ab54a8bac53380eb8fe	Rodolfo	2017-02-03 12:54:01	15199
58947e6b4a8bac37e95055c2	Rodolfo	2017-02-03 13:06:53	20.00

Figura 0.17. Facturas Generadas
Fuente: Autoría Propia

Por otro lado, en la aplicación móvil, las facturas generadas están listadas para que el usuario pueda descargarlas y visualizarlas desde el dispositivo móvil.



Figura 0.18 Facturas Generadas en la aplicación móvil
Fuente: Autoría Propia

Al pulsar sobre una de estas se mostrara el detalle de la factura seleccionada el cual se nos abrirá en el lector de PDF que posea nuestro dispositivo móvil.



Figura 0.19 Visualización de la factura desde el dispositivo móvil
Fuente: Autoría Propia

CONCLUSIONES

- Luego de analizar los procesos que se han implementado para la empresa *Roosman*, se pudo verificar un eficiente manejo de información no solamente en la facilitación de las ventas y la llegada al usuario final, si no que adicionalmente se lleva un registro de información que podría ser utilizado en el futuro para el análisis de mercado.

- El uso de un API permite que el desarrollo del *Front-End Web* como la aplicación móvil sea mucho más simple ya que para estos no utilizan ningún tipo de lógica de negocio y al trabajar con AngularJS se han demostrado que se puede reutilizar los mismos métodos en los dos proyectos.

- Al manejar bases no relacionales como MongoDB se han tenido ventajas en el sentido de que al acceder a un documento se obtiene toda la información del mismo permitiendo realizar las consultas de una manera más ágil, pero una de las desventajas notables fue la integridad referencial que al no poseer esto en MongoDB se tuvo que realizar validaciones por medio de programación para controlar este tipo de relaciones referenciales.

- El uso del API REST ayuda a crear un ambiente más escalable y veloz, ya que los cambios realizados en la aplicación *Web* se reflejan inmediatamente en la aplicación móvil, y viceversa; con el manejo de autenticación por *Tokens* es más veloz y fluido ya que no crea secciones en la parte del servidor y todos pueden acceder de forma más ágil a los recursos que desean.

RECOMENDACIONES

Al usar el sistema presentado se recomienda llevar un estándar en cuanto a las imágenes que se van a cargar tanto en productos como en las pantallas generales de categorías y secciones con el fin de tener un ambiente más agradable para el usuario final.

Si se desea continuar el proyecto se propone completar los métodos de pago incluyendo métodos como tarjetas de crédito, *paypal* o débito bancario que en este caso representaba un costo monetario adicional que no se encontraba contemplado en el presupuesto del proyecto.

Adicional a esto como se mencionó anteriormente se puede implementar la parte estadística en el API, para hacer análisis de mercado como son los productos más vendidos, productos menos vendidos, días de mayor número de ventas, entre otros.

Se recomienda la posibilidad de difundir la aplicación de esta herramienta a los negocios pequeños y medianos para que puedan obtener un mayor crecimiento, desarrollo; mediante la ampliación de esta herramienta en su mercado.

Se recomienda que para realizar la compilación de la aplicación móvil para sistemas operativos móviles con IOS, se debe de contar con una computadora MAC, que disponga de un sistema operativo *Yosemite*, para realizar la compilación de la aplicación móvil

GLOSARIO

API, *Application Programming Interface*, es un conjunto de subrutinas, funciones y procedimientos (o métodos, en la programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro *software* como una capa de abstracción.

Bitbucket, es un servicio de alojamiento basado en *Web*, para los proyectos que utilizan el sistema de control de revisiones *Mercurial* y *Git*.

Bootstrap, es un *framework* desarrollado y liberado por *Twitter* que tiene como objetivo facilitar el diseño *Web*.

MVC, El modelo–vista–controlador (MVC) es un patrón de arquitectura de *software*, que separa los datos y la lógica de negocio de una aplicación de la interfaz de usuario y el módulo encargado de gestionar los eventos y las comunicaciones.

Bower, es una herramienta que nos permite encontrar proyectos y herramientas de *javascript* y administrarlos dentro de nuestros proyectos, dicho de forma fácil nos permite descargar y actualizar de forma sencillas paquetes.

Composer, es un gestor de dependencias en proyectos, para programación en PHP. Eso quiere decir que nos permite gestionar (declarar, descargar y mantener actualizados) los paquetes de software en los que se basa nuestro proyecto PHP.

Controllers, son los archivos que poseen cierto código que ayudara a poder conectar la vista con el modelo, para poder mandar la información necesaria a la vista, generalmente son usados en los modelos MVC.

CSS, es un lenguaje de hojas de estilos creado para controlar el aspecto o presentación de los documentos electrónicos definidos con HTML y XHTML.

Dashboard, Es una interfaz donde el usuario puede administrar el equipo y/o *software*.

eBay, es un sitio destinado a la subasta de productos a través de Internet. Es uno de los pioneros en este tipo de transacciones, habiendo sido fundado en el año 1995.

Framework, es un entorno o ambiente de trabajo para desarrollo; dependiendo del lenguaje normalmente integra componentes que facilitan el desarrollo de aplicaciones como el soporte de programa, bibliotecas, plantillas y más.

Front-End, En diseño *Web* (o desarrollo *Web*) hace referencia a la visualización del usuario navegante.

Git, es un *software* de control de versiones diseñado por Linus Torvalds, pensando en la eficiencia y la confiabilidad del mantenimiento de versiones de aplicaciones cuando éstas tienen un gran número de archivos de código fuente.

GitHub, es un servicio de alojamiento o repositorio basado en *Web*, para los proyectos que utilizan el sistema de control de revisiones *Mercurial* y *Git*.

Homestead, es un ambiente para desarrollo local que en solo unos pocos pasos nos configura todo lo necesario para llevar a cabo nuestros proyectos en Laravel y prácticamente cualquier proyecto en PHP.

HTML, es un lenguaje de marcado que se utiliza para el desarrollo de páginas de Internet. Se trata de la sigla que corresponde a *HyperText Markup Language*,

HTTP, Abreviatura de la forma inglesa *Hypertext Transfer Protocol*, ‘protocolo de transferencia de hipertextos’, que se utiliza en algunas direcciones de Internet.

JavaScript, es un lenguaje de programación, que se utiliza principalmente del lado del cliente (es decir, se ejecuta en nuestro ordenador, no en el servidor) permitiendo crear efectos atractivos y dinámicos en las páginas *Web*.

JSON, *JavaScript Object Notation* - Notación de Objetos de *JavaScript* es un formato ligero de intercambio de datos.

JWT, (*JSON Web Tokens*) es la autenticación por medio de *Tokens* y usa el estándar para este tipo de autenticación es utilizar JSON Web Tokens (JWT).

Laravel, es un *framework* de código abierto para desarrollar aplicaciones y servicios Web con PHP.

Lumen, es un *micro-framework* para PHP creado por Taylor Otwell que comparte muchos de los componentes de su “hermano mayor” Laravel.

Middleware, son compontes que algunos viene incluidos en los *frameworks* para estar a la escucha de peticiones y realizar intercambios de información.

PHP, es un lenguaje de código abierto muy popular especialmente adecuado para el desarrollo *Web* y que puede ser incrustado en HTML.

Plugin, es aquella aplicación que, en un programa informático, añade una funcionalidad adicional o una nueva característica al software.

Postman, es una extensión gratuita para el navegador Google *Chrome* que permite probar servicios *Web* fácilmente, basta con indicar la url, el método HTTP (*POST*, *GET*, etc.) y los parámetros de la petición.

REST, es un estilo de arquitectura software para sistemas hipermedia distribuidos como la *World Wide Web*.

Scaffolding, es un término utilizando en programación para designar la construcción automática de aplicaciones a partir del esquema de la base de datos o sistemas de carpetas.

Scope, es un objeto que hace referencia al modelo de la aplicación. Es un contexto de ejecución para las expresiones.

Scrum, es un proceso de la Metodología Ágil que se usa para minimizar los riesgos durante la realización de un proyecto, pero de manera colaborativa.

Seed, es un término usado en informática, para cuando se desea llenar tablas de bases de datos con una semilla que tenemos para la población de la tabla.

SOAP, es un protocolo estándar que define cómo dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos XML.

SQL, es un lenguaje específico del dominio que da acceso a un sistema de gestión de bases de datos relacionales que permite especificar diversos tipos de operaciones en ellos.

Stock, Conjunto de mercancías o productos que se tienen almacenados en espera de su venta o comercialización.

Symfony, es un completo *framework* diseñado para optimizar el desarrollo de las aplicaciones Web basado en el patrón Modelo Vista Controlador. Para empezar, separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación Web.

Tokens, es un dispositivo electrónico que se le da a un usuario autorizado de un servicio computarizado para facilitar el proceso de autenticación.

Trello, es un gestor de tareas que permite el trabajo de forma colaborativa mediante tableros (*board*) compuestos de columnas (llamadas listas) que representan distintos estados.

URI, son las siglas en inglés de *Uniform Resource Identifier* (en español identificador uniforme de recursos), que sirve para identificar recursos en Internet.

Vagrant, es una herramienta para desarrolladores que facilita la creación de entornos virtuales para desarrollo.

VirtualBox, es una herramienta de virtualización de código abierto multiplataforma disponible para Windows, Linux y Mac OS X u otros sistemas operativos, que permite crear unidades de disco virtuales donde podemos instalar un sistema operativo invitado dentro del que utilizamos normalmente en nuestro equipo

Wireframe, para un sitio *Web*, también conocido como un esquema de página o plano de la pantalla, es una guía visual que representa el esqueleto o estructura visual de un sitio *Web*.

XML, es una adaptación del SGML (*Standard Generalized Markup Language*), un lenguaje que permite la organización y el etiquetado de documentos. Esto quiere decir que el XML, no es un lenguaje en sí mismo, sino un sistema que permite definir lenguajes de acuerdo a las necesidades.

XML-RPC, es un protocolo de llamada a procedimiento remoto que usa XML para codificar los datos y HTTP como protocolo de transmisión de mensajes.

BIBLIOGRAFÍA

1. MARQUÉS, A. 2013 “*Conceptos sobre APIs REST*”, 11 abril 2013, de <http://asiermarques.com/2013/conceptos-sobre-apis-rest/>
2. PALACIOS, D. 2015 “*Lumen, la versión micro framework de Laravel*”, 14 abril 2015, de <https://styde.net/lumen-la-version-micro-framework-de-laravel-php/>
3. BASALO, A. , ÁLVAREZ, M. 2014 “*Qué es AngularJS*”, 28 agosto 2014, de <http://www.desarrolloweb.com/articulos/que-es-angularjs-descripcion-framework-javascript-conceptos.html>
4. ACENS, 2014 “*Bases de datos NoSQL. Qué son y tipos dque nos podemos encontrar*”, febrero 2014, de <https://www.acens.com/wp-content/images/2014/02/bbdd-nosql-wp-acens.pdf>
5. LETELIER, P., PENADÉS, M., 2006 “*Métodologías ágiles para el desarrollo de software*”, 15 enero de 2006, de <http://www.cyta.com.ar/ta0502/v5n2a1.htm>
6. SCHWABER, K., SUTHERLAND, J., 2013 “*La Guía de Scrum*”, Julio del 2013, de <http://www.scrumguides.org/docs/scrumguide/v1/Scrum-Guide-ES.pdf>
7. BBVAOpen4U, 2016 “*API REST: qué es y cuáles son sus ventajas en el desarrollo de proyectos*”, 23 de marzo 2016, de <https://bbvaopen4u.com/es/actualidad/API-rest-que-es-y-cuales-son-sus-ventajas-en-el-desarrollo-de-proyectos>
8. TEJERO, J., 2013 “*SPA, un paradigma de Arquitectura de Aplicaciones Web en auge*”, 7 de octubre 2013, de <https://cink.es/blog/2013/10/07/spa-un-paradigma-de-arquitectura-de-aplicaciones-Web-en-auge/>

ANEXOS

Anexo A. Informe JAD	I
Anexo B. Wireframes de la aplicación Web.	III
Anexo C. Wireframes de la aplicación móvil.	XII

Informe JAD

Quito 10 Septiembre del 2016

El presente informe trata sobre la reunión realizada en entre la Empresa ROOSMAN, la Señora Rosa Hernandez representante, y el equipo de Desarrollo. Para la toma de requerimientos que va a realizar el equipo de Desarrollo para la aplicación que desea la Empresa ROOSMAN

La Empresa ROOSMAN desea una aplicación web y una aplicación móvil para la venta de ropa en línea.

Para lo siguiente la empresa se desea:

- Que los productos estén clasificados primero por categorías, pero que cada categoría posea secciones.
- Que se puede crear nuevas categorías y nuevas secciones.
- Que se pueda actualizar los productos ya sea el nombre, precio, etc
- Que los sistemas maneja impuestos, que pueden ser de varios tipos
- Que exista la posibilidad de crear códigos de descuento para los clientes y los puedan ingresar en el sistema
- Que tenga un método de pago alternativo a las tarjetas de crédito
- Que se pueda configurar parámetros como sea el nombre, dirección, etc. Que se puedan modificar o actualizar estos parámetros
- Que se pueda cambiar las imágenes de los productos o categorías o secciones.
- Que se pueda buscar las facturas por cliente o por fecha
- Que se pueda imprimir las facturas
- Que se maneje un stock individual por producto
- Que cada cliente posea un carrito de compras
- Que se pueda enviar por mail las facturas a los clientes
- Que los clientes puedan ver sus facturas en el sistema
- Que se puedan buscar productos
- Que la aplicación pueda tener los link de sus redes sociales
- Que se pueda subir imágenes para los productos

La empresa no le interesa que lenguaje de programación, servidor, sistema operativo, etc; que use el equipo de desarrollo.

La empresa desea preferiblemente que la aplicación móvil sea para celular con sistema operativo Android, pero si se puede también tener una aplicación para IOS mucho mejor.

Para lo cual el equipo de desarrollo plantea lo siguiente a la empresa ROOSMAN

- Dividir en dos partes la aplicación, una para los usuarios y otra para el administrador
- Crear CRUD (create, recover, update, delete) de las siguientes opciones: productos, categorías, sección, impuestos, códigos de descuento.
- Analizar un método de pago alternativo
- Pantalla de configuración de parámetros básico para la aplicación
- Análisis de lenguaje y herramientas para el desarrollo

- Reportes para la parte de facturas
- Diseño de posibles wireframes para el uso

Por lo cual la empresa ROOSMAN acepta lo establecido por parte del equipo de desarrollo, y espera futuras reuniones para ver el avance de las aplicaciones web y móvil

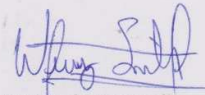
Con lo cual el equipo de desarrollo se compromete a mostrar avances de las aplicaciones a la Empresa ROOSMAN, para así analizar posibles cambios que se deban realizar.

Para la finalización de este documentó constan las firmas de la empresa y un representante del equipo de desarrollo.



ROOSMAN
Sra. Rosa Hernandez

170716 448-7



Equipo de Desarrollo
Sr. Wagner Santos

171712392-9

Anexo B. Wireframes de la aplicación Web.

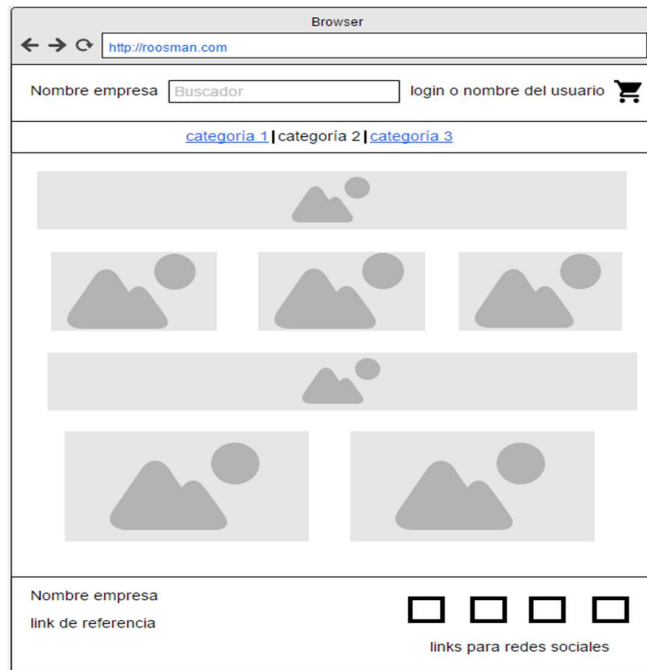


Figura 0.1. Pantalla home o de inicio

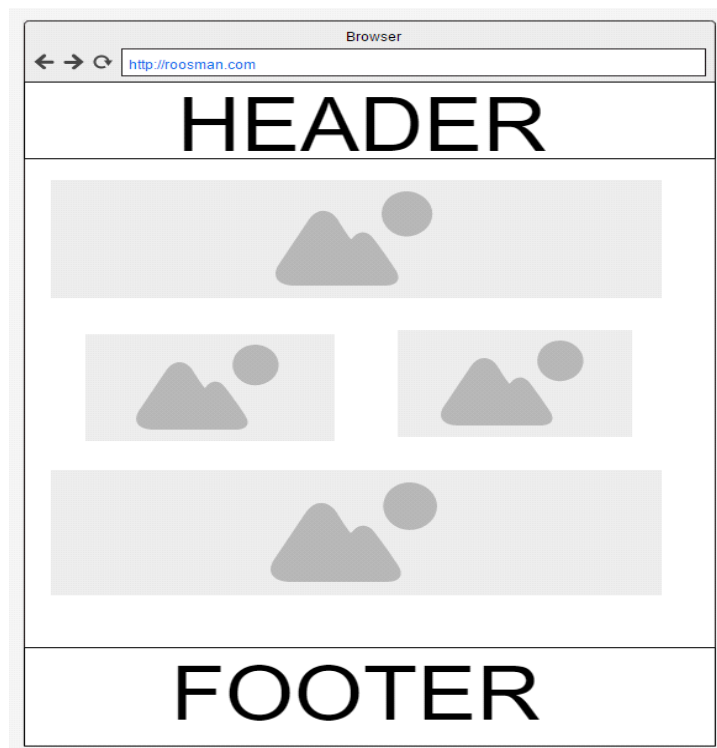


Figura 0.2. Pantalla de Categorías

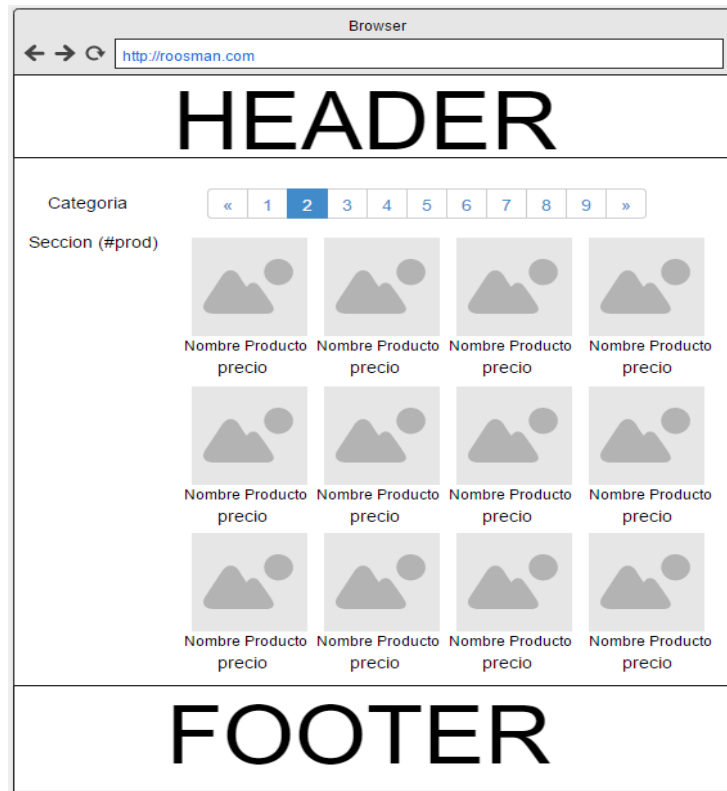


Figura 0.3. Lista de los productos dentro de una sección y categoría



Figura 0.4. Dashboard del administrador

Mozzila

← → ↻

http://roosman.com

HEADER

Lista de Categorías

Nuevo Producto

Productos

Categorías

Secciones

Impuestos

Código de descuento

▼ Nombre	▼ Categoría	▼ Seccion	▼ accion
Cell 1	Cell 2	Cell 3	[edit][delete]
Cell 4	Cell 5	Cell 6	[edit][delete]
Cell 7	Cell 8	Cell 9	[edit][delete]
Cell 10	Cell 11	Cell 12	[edit][delete]

FOOTER

Figura 0.5. Lista de productos

Browser

← → ↻

http://roosman.com

HEADER

Agregar Producto

Productos

Categorías

Secciones

Impuestos

Códigos de descuento

categoria

Options ▼

sección

Options ▼

nombre

descripción

fabricante

impuesto

☒ impuesto

precio

precio de venta

recargo

tallas

Options ▼

stock

colores de talla seleccionada

agregar color

color

cantidad

quitar color

subir imagenes

guardar

cancelar

FOOTER

Figura 0.6. Creación de un nuevo producto

V

Browser

← → ↻ <http://roosman.com>

HEADER

Lista de Categorías

[Nueva Categoría](#)

Productos	▼ Nombre	▼ Descripción	▼ Acciones
Categorías	Cell 1	Cell 2	[edit] [delete]
Secciones	Cell 4	Cell 5	[edit] [delete]
Impuestos	Cell 7	Cell 8	[edit] [delete]
Códigos de descuento	Cell 10	Cell 11	[edit] [delete]

FOOTER

Figura 0.7. Lista de las categorías

Browser

← → ↻ <http://roosman.com>

HEADER

Nueva Categorías

Productos	Nombre	<input type="text"/>
Categorías	Descripción	<input type="text"/>
Secciones	Imagen para banner	<input type="button" value="subir imagen"/>
Impuestos	Imagen para display	<input type="button" value="subir imagen"/>
Códigos de descuento		
Confirmar deposito		
Configuración General		
Pantalla Home	<input type="button" value="guardar"/>	<input type="button" value="cancelar"/>
Reporte de Facturas		

FOOTER

Figura 0.8. Creación de nueva categoría

Moqzilla

← → ↻ <http://roosman.com>

HEADER

Lista de Secciones

[Nuevo Sección](#)

Productos
Categorías
Secciones
Impuestos
Código de descuento

▼ Nombre	▼ Categoría	▼ accion
Cell 1	Cell 2	[edit][delete]
Cell 4	Cell 5	[edit][delete]
Cell 7	Cell 8	[edit][delete]
Cell 10	Cell 11	[edit][delete]

FOOTER

Figura 0.9. Creación de nueva sección

Browser

← → ↻ <http://roosman.com>

HEADER

Nueva sección

Productos
Categorías
Secciones
Impuestos
Códigos de descuento
Confirmar deposito
Configuración General
Pantalla Home
Reporte de Facturas

Nombre

Descripción

categorías ▼

Tallas [agregar talla](#)

Talla

[borrar talla](#)

Imagen para banner [subir imagen](#)

Imagen para display [subir imagen](#)

[guardar](#) [cancelar](#)

FOOTER

Figura 0.10. Creación de nueva sección

Browser

← → ↻ <http://roosman.com>

HEADER

Lista de impuestos

Nueva impuesto

Productos	▼ Nombre	▼ Descripción	▼ Porcentaje	▼ Acciones
Categorías	Cell 1	Cell 2	cell 3	[edit] [delete]
Secciones	Cell 4	Cell 5	cell 6	[edit] [delete]
Impuestos	Cell 7	Cell 8	cell 9	[edit] [delete]
Códigos de descuento	Cell 10	Cell 11	cell 12	[edit] [delete]

FOOTER

Figura 0.11. Lista de impuestos.

Browser

← → ↻ <http://roosman.com>

HEADER

Nueva impuesto

Nombre

Descripción

Porcentaje

guardar cancelar

FOOTER

Figura 0.12. Creación de nuevo impuesto.

Browser

← → ↻

http://roosman.com

HEADER

Lista de códigos de Descuento

Nueva código de Descuento

Productos

Categorías

Secciones

Impuestos

Códigos de descuento

▼ Nombre	▼ % Descuento	▼ fecha caducidad	▼ Acciones
Cell 1	Cell 2	cell 3	[edit] [delete]
Cell 4	Cell 5	cell 6	[edit] [delete]
Cell 7	Cell 8	cell 9	[edit] [delete]
Cell 10	Cell 11	cell 12	[edit] [delete]

FOOTER

Figura 0.13. Lista de códigos de descuento.

Browser

← → ↻

http://roosman.com

HEADER

Nueva Código de Descuento

Productos

Categorías

Secciones

Impuestos

Códigos de descuento

Confirmar deposito

Configuración General

Pantalla Home

Reporte de Facturas

codigo

% de descuento

fecha caducidad

estado

cantidad de uso

limite de uso

Options ▼

guardar

cancelar

FOOTER

Figura 0.14. Creación de nuevo código de descuento.

← → ↻

http://roosman.com

HEADER

Confirmar Pago

Productos

Categorías

Secciones

Impuestos

Códigos de descuento

Confirmar deposito

Código de Factura

Email Address

buscar

▼ Nombre	▼ cantidad	▼ fecha compra	▼ confirmación
Cell 1	Cell 2	cell 3	[x]

FOOTER

Figura 0.15. Confirmación del Pago

← → ↻

http://roosman.com

HEADER

Configuración

Productos

Categorías

Secciones

Impuestos

Códigos de descuento

Confirmar deposito

Configuración General

nombre de la empresa

Email Address

dirección

Email Address

telefono

Email Address

número de cuenta

Email Address

nombre de la cuenta

Email Address

Días validez factura

Email Address

facebook

Email Address

instagram

Email Address

android store

Email Address

apple store

Email Address

logo de la empresa

subir imagen

guardar

FOOTER

Figura 0.16. Parámetros configurables.

X

← → ↻

http://roosman.com

Browser

Productos

Categorías

Secciones

Impuestos

Códigos de descuento

Confirmar deposito

Configuración General

Pantalla Home

Diseño de la pantalla home

Banner 1

Display 1

Display 2

Display 3

Banner 2

Display 4

Display 5

Banner 1

Options

Banner 2

Options

Display 1

Options

Display 2

Options

Display 3

Options

Display 4

Options

Display 5

Options

guardar

cancelar

Figura 0.17. Administración de la pantalla de inicio.

← → ↻

http://roosman.com

Browser

Productos

Categorías

Secciones

Impuestos

Códigos de descuento

Confirmar deposito

Configuración General

Pantalla Home

Reporte de Facturas

Facturas por Fecha

Q

Buscar cliente

Fecha Inicio

4/22/2012

▼

Fecha Fin

4/22/2012

▼

Buscar

▼ Factura	▼ Cliente	▼ Fecha	▼ valor
Cell 1	Cell 2	Cell 3	Cell 4
Cell 5	Cell 6	Cell 7	Cell 8
Cell 9	Cell 10	Cell 11	Cell 12
Cell 13	Cell 14	Cell 15	Cell 16

Figura 0.18. Reporte de facturas.

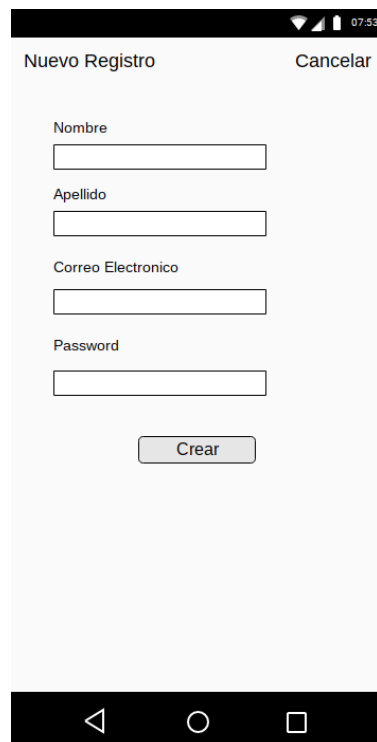
XI

Anexo C. Wireframes de la aplicación móvil.



This wireframe represents the login screen of a mobile application. At the top, there is a status bar with icons for signal, battery, and the time 07:53. Below this is a large, light gray rectangular area. In the center of this area is a placeholder image showing a stylized mountain and a sun. Below the image are two text input fields. The first is labeled 'correo electronico' and the second is labeled 'contraseña'. Below these fields are two buttons: 'Ingresar' (Login) and 'Registrarse' (Register). At the bottom of the screen is a black navigation bar with three white icons: a back arrow, a circle, and a square.

Figura 0.19. Pantalla de login.



This wireframe represents the new user registration screen. At the top, there is a status bar with icons for signal, battery, and the time 07:53. Below this is a light gray rectangular area. At the top of this area, there are two text labels: 'Nuevo Registro' on the left and 'Cancelar' on the right. Below these labels are four text input fields. The first is labeled 'Nombre', the second is labeled 'Apellido', the third is labeled 'Correo Electronico', and the fourth is labeled 'Password'. Below these fields is a single button labeled 'Crear' (Create). At the bottom of the screen is a black navigation bar with three white icons: a back arrow, a circle, and a square.

Figura 0.20. Creación de nuevo usuario



Figura 0.21. Pantalla home o de inicio

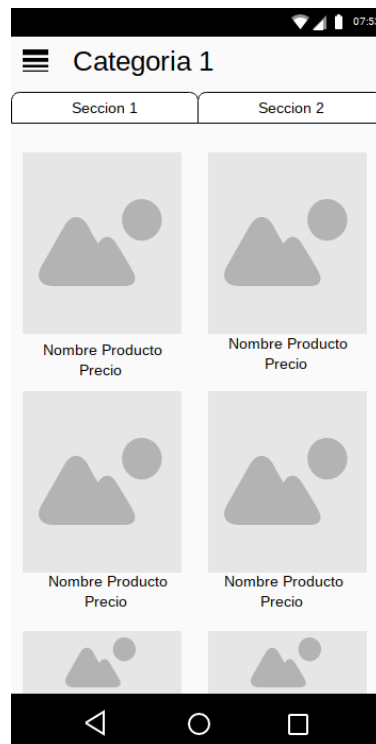


Figura 0.22. Productos por categoría

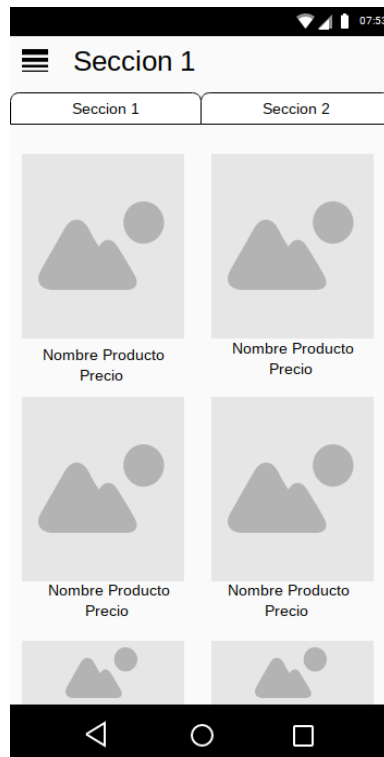


Figura 0.23. Productos por sección

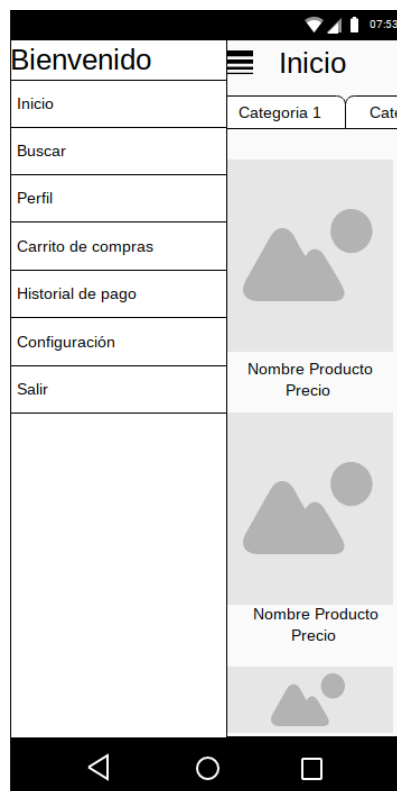


Figura 0.24. Menú lateral del usuario

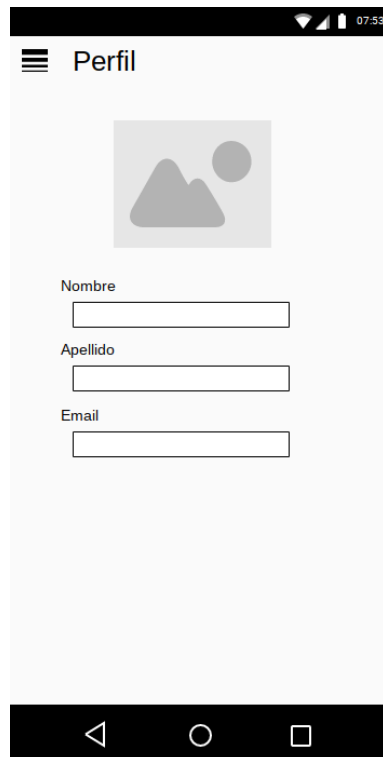


Figura 0.25. Pantalla de perfil del usuario

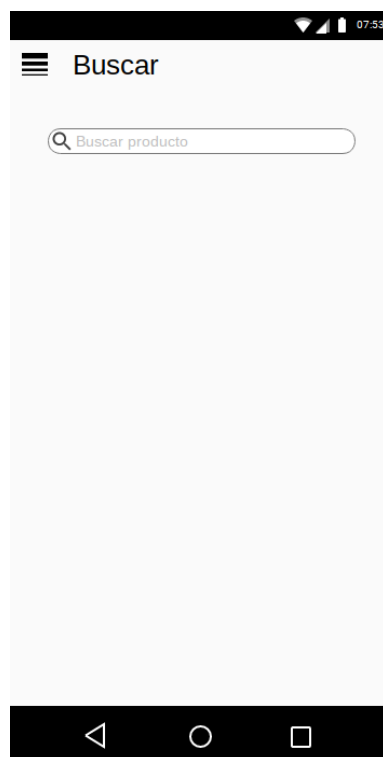


Figura 0.26. Pantalla de búsqueda



Figura 0.27. Carrito de Compra



Figura 0.28. Pantalla de Historial de pago

Configuración

Email

Cambiar Email:

Guardar Email

Datos Personales

Cambiar Nombre:

Cambiar Apellido:

Guardar Datos

Contraseña

Contraseña Anterior:

Nueva Contraseña:

Confirmar Contraseña:

Guardar Contraseña

Figura 0.29. Configuración del usuario