FACULTAD DE INGENIERÍA Y CIENCIAS AGROPECUARIAS

DESARROLLO DE UN PROTOTIPO PARA LA GESTIÓN Y VENTAS DE MEDICINA A TRAVÉS DE RECETAS MÉDICAS USANDO GEOLOCALIZACIÓN

Trabajo de Titulación presentado en conformidad
a los requisitos establecidos para optar por el título de
Ingeniero Electrónico y Redes de la Información

Profesor Guía
Ms. Carlos Andrés Guaita Ayala

Autor
Santiago Alejandro Montero Santacruz

Año
2017

DECLARACIÓN DEL PROFESOR GUÍA

"Declaro haber dirigido este trabajo a través de reuniones periódicas con el estudiante, orientando sus conocimientos y competencias para un eficiente desarrollo del tema escogido y dando cumplimiento a todas las disposiciones vigentes que regulan los Trabajos de Titulación".

_____

Carlos Andrés Guaita Ayala
Master Universitario en Ingeniería Biomédica
C.I.: 1715607071

DECLARACIÓN DEL PROFESOR CORRECTOR

“Declaro haber revisado este trabajo dando cumplimiento a todas las disposiciones vigentes que regulan los Trabajos de Titulación”.

_____

Verónica Fernanda Falconí Ausay
Magister en Ciencias de la Computación y Comercio Electrónico Msc
C.I.: 0502395270

DECLARACIÓN DE AUTORÍA DEL ESTUDIANTE

“Declaro que este trabajo es original, de mi autoría, que se han citado las fuentes correspondientes y que en su ejecución se respetaron las disposiciones legales que protegen los derechos de autor vigentes”.

_____

Santiago Alejandro Montero Santacruz

C.I.: 1715521413

AGRADECIMIENTO

A mis padres Vicente y Wilma por apoyarme constantemente y ser los pilares de mi vida.

A mis hermanas Viviana y Cristina por su cariño y consejos a lo largo de toda mi etapa académica.

Santiago

RESUMEN

Actualmente, la industria farmacéutica es un sector que posee mercados de constante crecimiento y potencial expansión, sin embargo, sus medios de distribución (farmacias) no brindan una atención completa al consumidor perjudicando el nivel de venta de sus productos. Una de las dificultades que muchos clientes enfrentan, especialmente pacientes que no se encuentran en buen estado de salud, es la necesidad de adquirir sus insumos médicos desde la comodidad de sus hogares. Es por ello que, con el fin de ayudar a solventar dicho problema, el presente Trabajo de Titulación propone el desarrollo de un sistema que facilite una comunicación directa entre farmacia y consumidor a través de un portal web y una aplicación móvil, permitiendo brindar un mejor servicio a la compañía farmacéutica.

ABSTRACT

Nowadays, the pharmaceutical industry is a sector with markets in constant growth and potential expansion; however, pharmacies don't provide a whole customer service relating to the distribution points resulting in decrease of their products sales. One of the difficulties that many clients face, especially patients who are not in good health, is the need to purchase their medical supplies from the comfort of their homes. In order to solve this problem, this present project proposes the development of a system which facilitates direct communication between pharmacies and costumers through a web portal and a mobile application, providing a better customer service to the pharmaceutical company.

ÍNDICE

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

1. INTRODUCCIÓN

1.1. Presentación del Trabajo

El presente Trabajo de Titulación propone el desarrollo de una aplicación para dispositivos móviles inteligentes con sistema operativo IOS y de un gestor web que en conjunto funcionarán como un sistema de ventas de recetas médicas a partir de fotografías tomadas por los clientes.

Este sistema se adaptará a las necesidades y a los productos de las diferentes farmacias para pretender ser una forma alternativa para la venta de insumos médicos.

1.2. Alcance

El trabajo de titulación contempla la implementación en un ambiente de pruebas de un aplicativo móvil para IOS a fin que el consumidor pueda buscar la farmacia más cercana a su ubicación actual. Una vez seleccionado el punto de venta más cercano o de su mayor agrado, el cliente procederá a tomar una foto de su receta médica y se remitirá a la farmacia de su elección.

El punto de venta seleccionado receptará una notificación de un nuevo pedido, el cual podrá ser descargado y este seleccionará los productos que tiene en stock, de esta manera se emitirá una respuesta con el valor total del pedido al cliente, mediante el uso de la página web que también se desarrollará en el presente trabajo de titulación, aclarando que no se incluirá un módulo de facturación electrónica o un sistema contable para la farmacia.

Se analizará los requerimientos para el correcto funcionamiento de la página web y base de datos, según estos se seleccionará el servidor más idóneo.

La última fase corresponde a la ejecución de pruebas de funcionamiento del prototipo sin reporte de fallas en el sistema.

1.3. Organización del Trabajo

El presente trabajo propone el desarrollo de un ambiente de prueba de un aplicativo móvil para IOS a fin que el consumidor pueda buscar la farmacia más cercana a su ubicación actual.

Como primer paso se procederá analizar el lenguaje de programación más eficiente y el desarrollo de la página web para gestionar las solicitudes, una vez realizado dicho análisis mediante herramientas como diagramas de casos de uso, se procederá a diseñar diferentes pantallas de administración a fin de facilitar el trabajo del personal farmacéutico o administrador del aplicativo en cada una de las empresas.

Después se ejecutará el mismo proceso con el aplicativo móvil y sus WebService a través de diagramas de casos de uso y de secuencias, en el cual se determinará las interfaces más óptimas para su desarrollo y orden.

Adicionalmente, se estudiará los requerimientos para el correcto funcionamiento de la página web y base de datos; de acuerdo a ello, se seleccionará el servidor más idóneo, su infraestructura tecnológica más adecuada y sus medidas de seguridad óptimas.

La última fase corresponde a la ejecución de pruebas de funcionamiento del prototipo a nivel global y su estabilidad.

2. ANTECEDENTES

2.1. Teléfonos Móviles Inteligentes

Los teléfonos móviles inteligentes se han convertido en una herramienta de uso masivo a nivel mundial, gracias a su integración de servicios de

telecomunicaciones e internet en un único dispositivo y al amplio abanico de aplicaciones disponibles en el mercado.

Estos dispositivos le deben su nombre a que en un principio servían únicamente para realizar llamadas y enviar/recibir mensajes de texto plano, pero hoy en día hacen mucho más que eso como: administrar toda su agenda, buscar lugares, encontrar las mejores rutas para poder acceder a ellos y adquirir productos en pocos pasos, sin contar que se adapta a las necesidades del país en donde operan y permite una completa personalización de la interfaz al usuario.

Todos los servicios que estos teléfonos inteligentes proveen, tienen un segmento de mercado que va desde niños, jóvenes, docentes, empresarios e incluso doctores quienes pueden mejorar sus prácticas y eficiencia en sus labores con el uso adecuado de los dispositivos.

Teniendo en cuenta el informe de movilidad de Ericsson (Ericsson, 2015), en el año 2015 fueron vendidos 1400 millones de teléfonos inteligentes y solo en el último trimestre de dicho año se vendieron 353 millones como se indica en la Figura 1, en la que se puede observar los porcentajes de unidades vendidas por sistema operativo.

Según Ericsson:

> *"Los Smartphone representaron cerca del 75% de todos los teléfonos móviles vendidos en el tercer trimestre de 2015, en comparación con el 70% del mismo periodo de 2014. Hoy en día alrededor de 45% de todas las suscripciones móviles de telefonía están asociados con los teléfonos inteligentes, en comparación con el 40% de 2014*"

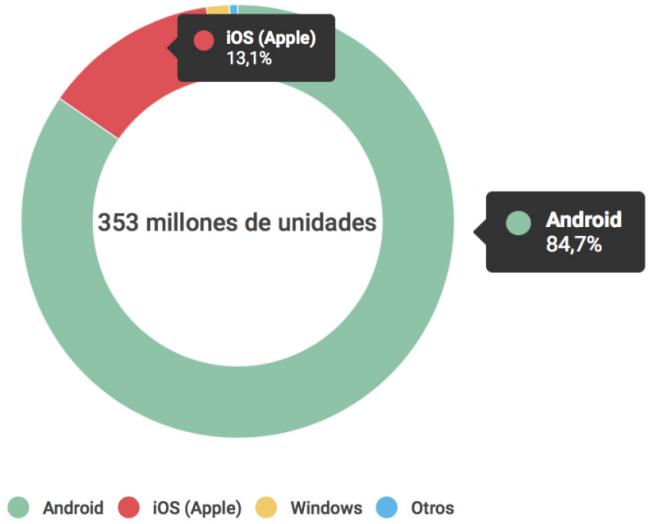


Figura 1. Ventas globales de *smartphones* según el sistema operativo (tercer trimestre de 2015)
Adaptado de (Álvarez, 2015)

Esta gran participación en el mercado se debe a diversos factores, siendo uno de los más importantes el creciente número de aplicaciones que se encuentran en cada una de sus respectivas tiendas como: Apple Store y Google Play.

El incremento del número de aplicaciones se debe a que estos dos sistemas operativos dan la posibilidad de desarrollar sus aplicativos a terceros. Google Play al publicar el aplicativo en su tienda conlleva un costo, además de poseer diferentes maneras para manejar las notificaciones. Así también, Apple cobra un valor de 99$ al año a sus desarrolladores dando acceso a publicar aplicativos y manejar notificaciones de tipo *push*. Adicionalmente, esta suscripción ofrece varios servicios extra, y garantizar a los usuarios finales que estos aplicativos fueron certificados y garantizados por la propia empresa Apple.

2.2. Aplicaciones de farmacias

En la actualidad, uno de los servicios más importantes que tiene la sociedad es la venta de productos médicos a través de farmacias, en las cuales es posible adquirir medicinas u otros insumos contribuyendo al buen estado de salud del consumidor.

2.2.1. Actualidad Internacional

A nivel mundial, existen varias páginas web sobre venta de medicamentos a clientes finales, pero pocas aplicaciones móviles logran ser efectivas. Un claro ejemplo es “CSV Pharmacy”, la cual detecta el código del medicamento mediante un scanner y el uso de una cámara permitiendo de este modo efectuar el pedido para un nuevo suministro. En la Figura 2 se observa una captura de pantalla de dicho aplicativo.

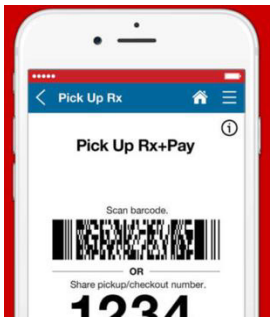


Figura 2. Copia de pantalla de CVS pharmacy
Tomado de (AppStore CVS, s.f.)

2.2.2. Actualidad Nacional

En Ecuador, actualmente las farmacias poseen un método de comercialización limitado, por cuanto los clientes tienen nulas o pocas opciones para comprar medicamentos de forma remota y en momentos prioritarios resulta complicado ubicar los locales más cercanos. En sí, los mecanismos para hacer pedidos a domicilio en el país son:

- Fybeca es el único que oferta a nivel nacional mediante vía web y app-móvi Figura 3 *y* Figura 4.

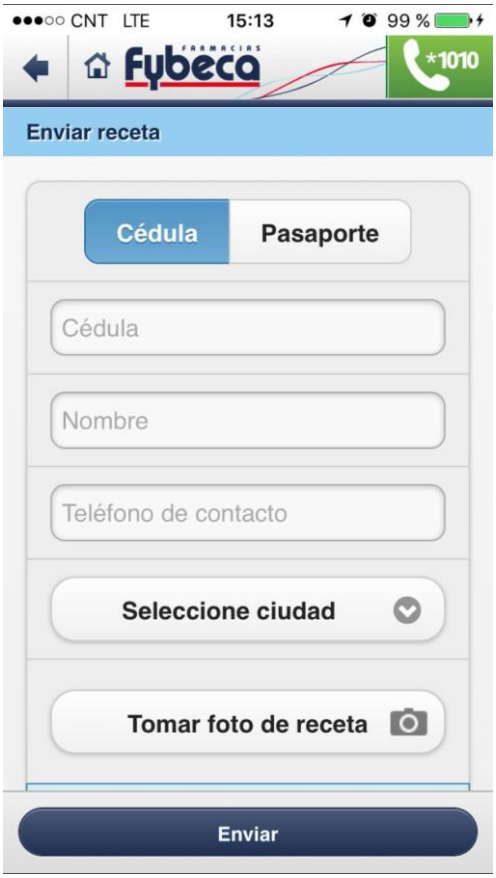


Figura 3. Página de solicitud App Fybeca
Tomado de (App Fybeca App Store, s.f.)

Figura 4. Página Principal App Fybeca
Tomado de (App Fybeca App Store, s.f.)

- Empresas como Confiamed, Globmedical y Axis mediante vía telefónica. Dichas compañías solicitan que sus clientes entreguen una copia de la receta y de la cédula de ciudadanía al momento de recibir su pedido.

3. OBJETIVOS

3.1. Objetivo General

Desarrollar un prototipo de sistema de gestión de venta de medicinas a través de la implementación de una página web y un aplicativo móvil a fin de facilitar la adquisición de insumos médicos al consumidor.

3.2. Objetivos específicos

- Analizar los requerimientos de infraestructura, el modelamiento de la base de datos y la metodología a utilizarse para el desarrollo del aplicativo web y móvil.

- Diseñar un aplicativo móvil prototipo para la ubicación de las farmacias cercanas utilizando geo localización y el envío de una fotografía de la receta médica para la gestión y despacho de los insumos por parte de las compañías.

- Implementar en un ambiente de pruebas un aplicativo móvil prototipo para el cliente final y una página web cuya función será la gestión de recetas médicas y despacho de compra al consumidor final.

- Ejecutar pruebas de funcionamiento y rendimiento de los sistemas.

4. MÉTODOS Y HERRAMIENTAS

4.1. Metodología de Diseño de Proceso Unificado, UP

Para el desarrollo de un prototipo de sistema de gestión y ventas de medicina a través de recetas médicas, se utilizará el marco de desarrollo de software conocido como Proceso Unificado (UP). Este proceso dará las directrices que se usan normalmente en desarrollo de sistemas de calidad.

Este modelo es adaptable a los requerimientos del usuario final en nuestro caso a los clientes de las farmacias y su principal característica es iterativo e incremental, dirigido por casos de uso y se centra en la arquitectura. (Jacobson, Booch, & Rumbaugh, 2000)

Al ser Iterativo e incremental esta metodología permite a través de varias iteraciones ofrecer una nueva versión del producto desarrollado ya que la arquitectura es modificada, validada y producida en cada iteración.

En esta metodología los casos son usados para capturar los requisitos funcionales y definir los contenidos de las iteraciones. La idea es que cada iteración tome un conjunto de uso y se desarrolle todo el trayecto desde el diseño hasta las pruebas.

Al ser centrado en la arquitectura el diseño del sistema se realiza mediante el uso de modelos visuales que abarcan todos los aspectos del sistema. Estos serán componentes conectados entre sí mostrando la interfaz definida utilizando el lenguaje Unificado de Modelado (UML).

4.2. Lenguaje Unificado de Modelo (UML)

Unos de los primeros pasos del Proceso Unificado es el modelado del sistema desde varios puntos de vista utilizando el Lenguaje Unificado de Modelado – UML.
UML posee una notación grafica que permite la representación gráfica de todas las fases del proyecto, desde el análisis hasta la implementación y configuración. (Universidad Politécnica de Valencia, 2002)

Los objetivos son:

- **Visualizar:** Expresa gráficamente el sistema.

- **Especificar:** Detalla las características del sistema antes de la construcción.

- **Construir:** Usando los modelos especificados se construye los sistemas diseñados.

- **Documentar:** Los elementos gráficos diseñados sirven como parte de la documentación del sistema desarrollado.

UML está compuesto por tres clases de bloques de construcción:

- **Elementos:** Abstracción de cosas reales o ficticias.
- **Relaciones:** Relacionan los elementos.
- **Diagramas:** Colecciones de elementos con las relaciones.

4.2.1. Tipos de Diagramas UML

Un diagrama se define como la representación gráfica del conjunto de elementos con sus relaciones. Estos diagramas ofrecen varios puntos de vista del mismo modelado.

UML tiene varios tipos de diagramas, entre los cuales se tiene: (Fowler & Sccott, 1999):

- **Diagrama de Estructura:** Son los diagramas de Clases, Componentes, Objetos y despliegue que corresponden a la estructura estática del sistema.

    - **Diagrama de Clases:** Es estático y describe la estructura del sistema mostrando las clases, interfaces y atributos y sus relaciones. Este diagrama se utiliza en las etapas de análisis y diseño y suele ser el más común para describir el diseño global.

    - **Diagrama de Paquetes:** Estos muestran al sistema dividido en agrupaciones lógicas y sus dependencias. Normalmente estos paquetes están pensados como directorio y los diagramas dan una descomposición de la jerarquía lógica del sistema.

- **Diagrama de Comportamiento e Interacción:** Es el comportamiento dinámico del sistema, los cuales pueden ser casos de uso, secuencias,

colaboraciones, estados y actividades, a continuación, se explicarán los dos más relevantes.

- **Diagrama de casos de uso:** Es la representación gráfica de los casos de uso que se definen como cada interacción posible del sistema a desarrollar con sus requisitos funcionales.

- **Diagrama de secuencia:** Se muestra la interacción de los diferentes objetos del aplicativo a través del tiempo y este se modela por cada caso de uso posible. Además, este diagrama muestra detalles de la implementación del escenario incluyendo las clases y objetos utilizados.

4.3. Arquitectura Cliente Servidor

El sistema de gestión de venta de medicinas a través de recetas médicas utilizará una arquitectura cliente-servidor como se muestra en la Figura *5*, ya que el aplicativo móvil consumirá servicios web situados en un servidor remoto.

Los aspectos más importantes de esta arquitectura son que el almacenamiento, proceso y control de datos pueden ser realizados en más de un computador o servidor y el intercambio de información es mediante la interconexión de una red de comunicación. (Ibarra Avalos, 2001)

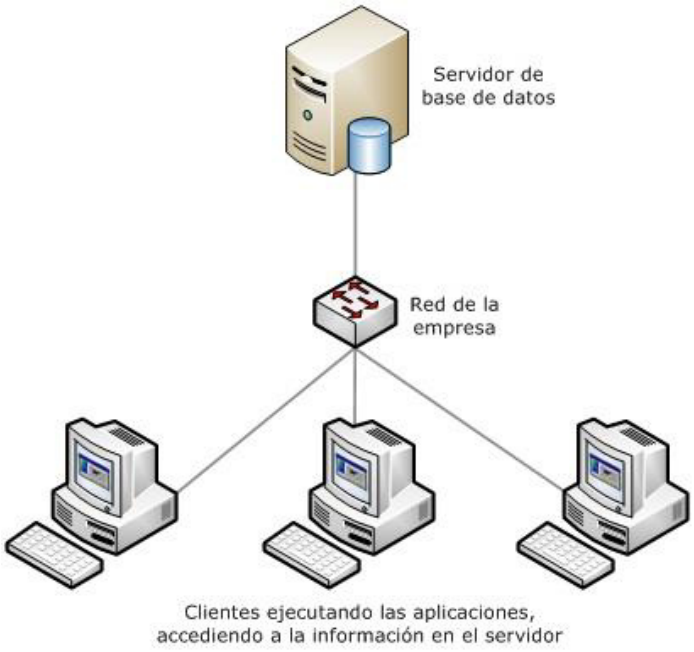


Figura 5. Arquitectura Cliente/Servidor
Tomado de (Net Humans, s.f.)

4.3.1. Elementos de la arquitectura Cliente Servidor

La arquitectura de este sistema ha ido evolucionando desde aplicaciones mono-capa que situaban tanto la aplicación como los datos en la misma máquina y ésta se encargaba de administrar la herramienta como se muestra en la Figura 6.

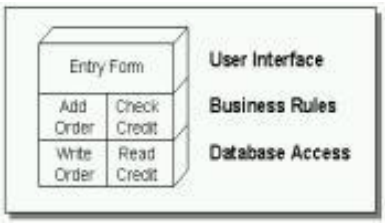


Figura 6. Arquitectura mono-capa
Tomado de (Ibarra Avalos, 2001)

El modelo en dos capas es la arquitectura clásica del cliente-servidor donde el cliente es quien implementa la interfaz, y el gestor de base de datos se encarga de las peticiones recibidas un ejemplo de esto se indica en la Figura 7.

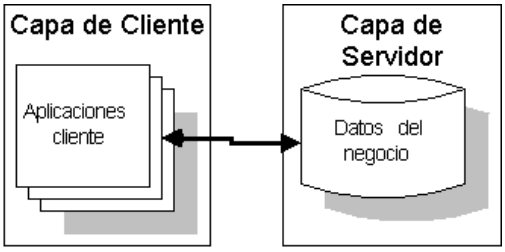


Figura 7. Arquitectura Cliente/Servidor clásica
Tomado de (Ibarra Avalos, 2001)

Por último, el modelo en tres capas se divide entre el cliente el servidor y un *middleware* que provee la interconectividad entre el cliente y el servidor a fin de intercambiar mensajes como se señala en la Figura 8.

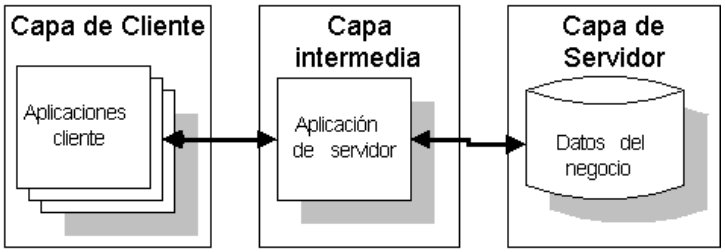


Figura 8. Arquitectura Cliente/Servidor en tres capas
Tomado de (Ibarra Avalos, 2001)

Cliente

Es el que inicia un requerimiento o petición de servicio. Esta petición inicial se genera mediante el uso de interfaces de usuario graficas (GUI) y puede escalarse a varios requerimientos de trabajo a través de la infraestructura de red, por lo tanto, la ubicación de los datos es transparente para el cliente.

Servidor

El servidor es el recurso de cómputo que se dedica a responder las peticiones del cliente, estos servidores están conectados a los clientes a través de infraestructuras de red alámbricas o inalámbricas. Estos pueden ser capaces de atender varias peticiones simultáneamente y suelen manejar la capa de negocio

de los aplicativos. Además, se puede mencionar que un servidor a su vez puede ser cliente de un segundo servidor.

Middleware

El middleware es un módulo que actúa como mediador en la comunicación entre sistemas y se ejecuta en el cliente, así como también en el servidor permitiendo a cualquier usuario del sistema comunicarse con las diferentes fuentes que se encuentren en la red. Además, gracias a este se permite desvincular los servidores con los clientes evitando la dependencia del sistema a una única tecnología propietaria.

4.4. JavaScript Object Notation (JSON)

El sistema de gestión de venta de medicinas a través de recetas médicas al poseer una arquitectura cliente servidor necesita intercambiar mensajes sobre todo entre la comunicación del aplicativo móvil con los webs services. Para realizar esta comunicación se determinó que la mejor opción es por medio del formato JSON que se conceptualiza como un formato ligero de comunicación derivado del lenguaje de programación JavaScript. El principal beneficio de dicho formato es su facilidad al momento de la lectura y escritura por parte de humanos y máquinas.

Este formato se usa para transmisión de una estructura serializada de datos a través de la red. Su principal uso es en aplicativos Clientes/Servidor siendo una alternativa al formato XML que se venía usando anteriormente. (JSON, s.f.)

Las dos estructuras que constituyen JSON son:

- Una Colección de pares de “Nombre/Valor” que podrían ser objetos o estructuras.

- Una lista ordenada de valores que pueden ser arreglos, vectores entre otros.

El formato que usa JSON para enviar la información debe estar incluido entre dos llaves “{}” y se compone del nombre de la variable seguido de “:”(dos puntos) y el valor de esta. A continuación, en la Figura 9 se muestra un ejemplo de JSON del uso en dicho sistema.


```
{
    "result": [
        {
            "puntoVenta_id": 14,
            "puntoVenta_nombre": "SanaSana",
            "puntoVenta_callePrincipal": "Av Cristobal Colon",
            "puntoVenta_calleSecundaria": "6 de diciembre",
            "puntoVenta_numeroTelefono": "12345678",
            "puntoVenta_lat": -0.20213640653424,
            "puntoVenta_lng": -78.48586318092,
            "puntoVenta_logo": "FARMAAP2-01",
            "puntoVenta_usuario": 1
        },
        {
            "puntoVenta_id": 15,
            "puntoVenta_nombre": "Fybeca",
            "puntoVenta_callePrincipal": "eloy alfaro",
            "puntoVenta_calleSecundaria": "granados",
            "puntoVenta_numeroTelefono": "1233434545",
            "puntoVenta_lat": -0.16609249644363,
            "puntoVenta_lng": -78.46877157561,
            "puntoVenta_logo": "marcas_sanasana",
            "puntoVenta_usuario": 2
        }
    ]
}
```


Figura 9. Ejemplo de objeto JSON

4.5. JQUERY

Cuando hablamos de páginas web transaccionales como son las que se implementan en este sistema, se tiene que mencionar a JQUERY, que es un framework de JavaScript utilizado para realizar validaciones y ejecutar funciones desde el lado del cliente, sin la necesidad de la comunicación con el servidor, además este lenguaje le permite a las páginas ser dinámicas y mediamente inteligentes sin que sean muy pesadas. Existen varias bibliotecas ya creadas en este lenguaje como *Bootstrap* que permiten que estas páginas se adapten a los dispositivos sean móviles o pc.

4.6. Modelo Vista Controlador (MVC)

La metodología del Proceso Unificado indica que una vez que se posea los distintos modelos del sistema se procederá con la fase de implementación del software. Es por eso que para este sistema se decidió utilizar uno de los patrones de desarrollo más utilizado hoy en día que se conoce como Modelo-Vista-Controlador (MVC).

Este patrón de desarrollo de software se basa en separar en capas distintas los datos y la lógica de negocio con la interfaz de usuario. En este sentido, MVC delinea que se debe construir tres componentes principales que son: el Modelo que se lo puede tomar como clases u objetos, la diferentes Vistas y los Controladores, en la Figura 10 se puede observar esta arquitectura cuando interacciona un usuario.

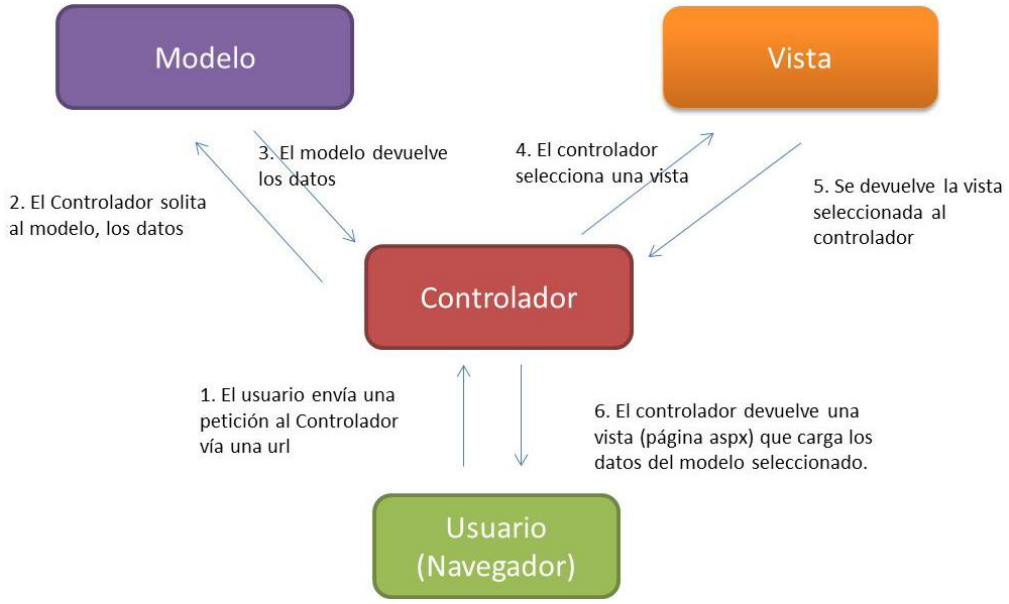


Figura 10. Modelo Vista Controlador

Esta arquitectura fue creada como medio facilitador para los desarrolladores permitiendo la separación de conceptos y reutilización de código, ya que se basa en la programación de objetos. Ofreciendo así una facilidad al mantenimiento y una escalabilidad notable, por estas razones es usado por Visual Studio en

lenguaje c#, Android Studio en *java* y en lenguaje *php* por medio de *Frameworks* como *Laravel*.

El flujo de trabajo de esta arquitectura es de la siguiente manera (Alvarez, 2014):

1. Un usuario realiza una solicitud a través de una página web que en este caso sería la Vista que envía a su vez esta petición al controlador.

2. El Controlador se encarga de procesar esta solicitud y requerir la información a la base de datos.

3. En este punto el Modelo devuelve la información solicitada al controlador y él se encarga de enviársela a la Vista.

4. Las Vistas despliegan hacia el usuario la salida de la información.

4.6.1. Componentes de la arquitectura MVC

Los componentes principales de MVC son: el Modelo, la vista y el controlador. A continuación se detalla cada uno de ellos:

- **Modelo**

  Esta es la capa donde se manipulan los datos mediante diferentes mecanismos de acceso. Estos datos se encuentran situados en una Base de Datos y el modelo se encargará de interactuar y ejecutará las funciones CRUD (Create, Read, Update, Delete).

- **Vista**

  En esta capa se ejecuta el código de la aplicación que se muestra al usuario en un aplicativo en nuestro caso este lenguaje en la página web será HTML

y los navegadores son los que se encargan de rende rizar este código y mostrar el resultado final al usuario. Esta Vista no tiene acceso directo a los datos y necesitan un intermediario hacia el modelo que en este caso es el controlador.

- **Controlador**

  Es la capa que se encarga de la comunicación entre la vista y el modelo, y puede escalarse a los requisitos necesarios que se puedan requerir de nuestra aplicación. Además, contiene el código que se utiliza para responder a las peticiones de la Vista y a la comunicación con la base de datos a través del Modelo.

4.7. Mapeo Objeto Relacional (ORM)

En la implementación del ambiente de prueba del sistema de gestión de venta de medicinas a través de recetas médicas se utilizará principalmente dos lenguajes de programación: PHP y Swift. Para PHP se usará un Framework llamado Laravel versión 5.2 con su librería de ORM llamada ENLOQUET.

Cuando se habla de programación orientada a objetos las tareas se implementan a través de la manipulación de objetos con el fin de gestionar los datos. Estos objetos suelen ser valores no escalares, sin embargo en las bases de datos únicamente se pueden manipular y almacenar valores escalares que pueden ser enteros o arreglos. Por lo tanto, para almacenar la información de un aplicativo desarrollado en un lenguaje orientado a objetos es necesario un medio de traducción entre la base de datos y el programa.

La librería ENLOQUET de ORM que incluye Laravel proporciona una implementación de ActiveRecord fácil de usar y muy agradable al usuario para el trabajo en conjunto con la base de datos. Cada tabla de la base de datos tiene un “Modelo” en el aplicativo, que se utiliza para la interacción con esa tabla.

Estos modelos pueden consultar datos de la tabla, además de poder insertar y actualizar los mismos. (Otwell, 2016)

Algunas de las ventajas del ORM es el aumento de rapidez en el desarrollo, además la reutilización de código el aumento de seguridad y el uso de un lenguaje propio para la realización de las consultas o instrucciones con la base.

Las desventajas de este tipo de mapeo puede ser el tiempo largo para las consultas por las transformaciones internas de lenguaje y posteriormente la consulta de registros en la base de datos, además que este tipo de herramienta suelen ser muy complejas para su aprendizaje, por lo tanto el tiempo de investigación puede influir al momento de elegir este tipo de mapeo.

4.8. Notificaciones por correo electrónico

El ambiente de prueba del sistema se desarrolló en plataformas gratuitas, por lo tanto las notificaciones *PUSH* propietarias no eran viables, ya que sería necesario solventar un costo de 90 $ a Apple por el servicio anual. Estas notificaciones usarán el sistema interno de los aplicativos de correos electrónicos que tengamos instalados en el teléfono móvil.

Framework como *Laravel* tiene funciones que facilitan en gran medida el envío de correos electrónicos, al hacer uso de la biblioteca *SwiftMailer*, que es utilizada por varios aplicativos Web, al poseer controladores para *SMTP, mailgun, SparkPost* entre otros. Estas funciones de *PHP* de envíos de email pueden funcionar a través de un servidor local o uno situado en la nube.

En la Figura 11 se puede observar la arquitectura cliente servidor del sistema y su transferencia de datos. El aplicativo móvil se comunica con el servidor mediante Webservice y cada vez que el servidor necesite comunicarse con el móvil este envía un email al usuario.

Figura 11. Arquitectura Movil/Servidor

4.9. SWIFT

Es un lenguaje de programación joven creado por Apple que vino a remplazar a Objectic-C, este lenguaje es intuitivo y permite el diseño de aplicaciones para IOS, MAC, APPLE TV y Apple WATCH.

Este código permite que las búsquedas sean mucho más rápidas que con su predecesor hasta 2,6 veces más veloz. Además en el 2015 paso a ser de código abierto y tiene como objetivo ser un lenguaje de programación seguro de desarrollo rápido y conciso.

SWIFT está diseñado para trabajar en conjunto con los *Frameworks COCOA y COCOA TOUCH* además que con cualquier librería creada para su predecesor *Objectic-C.*

Como dato curioso es un lenguaje fuertemente tipado, aun cuando la declaración de las variables no es siempre necesaria. Esto se debe a la habilidad del software al momento de inferir tipos.

5. RESULTADOS

5.1. Identificación de Actores

Cuando diseñamos un sistema tecnológico una de principales claves de éxito es identificar correctamente los actores principales de la solución. A continuación, se diferencia los dos actores principales del sistema desarrollado.

- **Clientes**

  Persona que tenga instalado el aplicativo en su dispositivo móvil y necesite realizar la compra de su receta médica.

- **Farmacéutico**

  Es el personal de la farmacia encargado de gestionar las solicitudes que ingresen al sistema crear las facturas y administrar la base de datos de productos de cada establecimiento.

5.2. Definición de Requerimientos

5.2.1. Requerimientos Funcionales

En la Tabla 1 se muestran todos los requerimientos funcionales del sistema de gestión de venta de medicinas a través de recetas médicas.

Tabla 1.

Requerimientos funcionales para el aplicativo móvil

Requerimientos Funcionales	Actores	Entrada	Proceso	Salida
Registro del usuario	Clientes	Nombre, email y contraseña	El cliente ingresará datos básicos como donde el sistema los registrará.	La página principal del aplicativo o un mensaje de error si los datos ingresados fueran erróneos
Autentificación del usuario	Clientes	Email y contraseña	El sistema comprobara las credenciales del usuario con el email y su contraseña.	La página principal del aplicativo o un mensaje de error si los datos ingresados fueran erróneos
Visualización de farmacias cercanas	Clientes	Código cliente	El sistema por medio de Webservice mostrará las farmacias cercanas y debe permitir seleccionar la que el cliente desee	Listado de farmacias y su ubicación
Realizar el pedido de la receta	Clientes	Foto de la receta, dirección de domicilio	El sistema permitirá que el usuario envié su pedido por medio de Webservices POST REST	Respuesta de confirmación
Visualización de estados de solicitudes	Clientes	Código usuario	El sistema permitirá al cliente visualizar las diferentes solicitudes creadas y sus estados de transacción.	Listado de solicitudes

Requerimientos Funcionales	Actores	Entrada	Proceso	Salida
Registro del Farmacéutico	Farmacéutico	Nombre, email y contraseña	El farmacéutico ingresará su nombre email y elegirá una contraseña	La página principal del Website o un mensaje de error si los datos ingresados fueran erróneos
Autentificación del usuario	Farmacéutico	Email y contraseña	El sistema comprobara las credenciales del usuario con el email y su contraseña para el ingreso a la página web	La página principal del Website o un mensaje de error si los datos ingresados fueran erróneos
Visualización datos de la farmacia	Farmacéutico	Nombre, teléfono, dirección, imagen corp., ubicación GPS	El sistema permitirá crear o modificar los datos básico y de ubicación de la farmacia	Respuesta de confirmación
Manejo de catálogo de productos	Farmacéutico	Nombre, código, stock, descripción, precio	El sistema permitirá la creación, actualización, eliminación de productos para cada farmacia	Respuesta de confirmación
Visualización de solicitudes de los clientes	Farmacéutico	Código solicitud	El sistema debe permitir la visualización de las solicitudes de esa farmacia y permitir interactuar con ellas.	Website de la solicitud con la ubicación del cliente e imagen de la receta
Procesar las solicitudes para creación de factura	Farmacéutico	Factura, código usuario, código solicitud	El sistema permitirá que el farmacéutico pueda procesar las solicitudes hechas por los clientes	Envío de correo al cliente y confirmación de la transacción

5.2.2. Requerimientos no funcionales

La Tabla 2 muestra los requerimientos no funcionales del sistema de gestión de venta de medicinas a través de recetas médicas.

Tabla 2.

Requerimientos no funcionales del sistema

Requerimientos No Funcionales	Descripción
Plataforma IOS	La aplicación móvil trabajará sobre un IPhone con sistema operativo IOS.
***WebServices* tipo JSON**	La aplicación deberá tener la capacidad de comunicarse al servidor mediante *WebServices*.
Página Web	La Página Web deberá poder funcionar correctamente en los navegadores Google Chrome, Firefox y Internet Explorer
Funcionamiento online	El funcionamiento del sistema será totalmente online
Servidor de aplicaciones Apache	El sistema funcionara en un servidor web Apache *Tomcat*.
PHP 5	La página web trabajara mediante el rende rizador de PHP 5.6.24
Motor de Base de Datos *MySQL*	El sistema deberá guardar toda la información de las solicitudes en el servidor en una base de datos *MySQL*.
Robustez y fiabilidad	El sistema se desarrollará de manera confiable y robusto a posibles errores de usuarios.
Disponibilidad	El sistema deberá tener alta disponibilidad los 365 días al año las 24 horas del día.

5.3. Diseño del sistema

A continuación, se detallarán los diagramas de estructura y comportamiento utilizados para el sistema.

5.3.1. Clases

Tomando en cuenta la metodología UP el diagrama de clases es estático y como se puede observar en la Figura 12 refleja los atributos y funciones de cada clase.

Las clases utilizadas son las siguientes:

- Users
- PuntoVenta
- Cliente
- Producto
- Solicitud
- Estado
- Factura_cabecera
- Factura_detalle

Además, se puede observar que la clase productos es la única a tener funciones de crear, editar y eliminar.

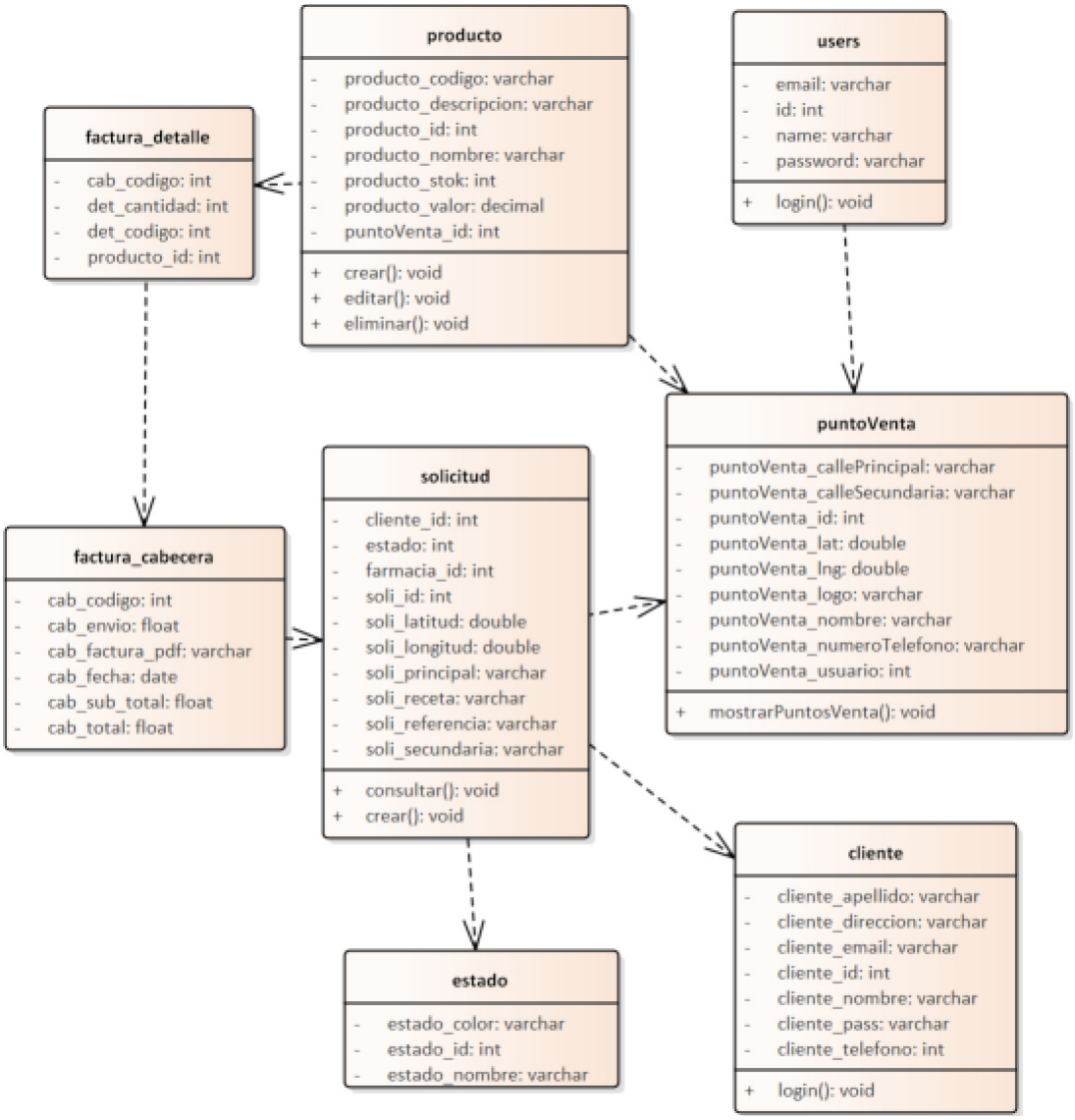


Figura 12. Diagrama de Clases

5.3.2. Componentes

El diagrama de componentes que se muestra en la Figura 13 permite visualizar el comportamiento de los servicios que proporcionan los diferentes módulos y se denota la estructura de alto nivel del proyecto.

Se puede observar también que las interfaces a utilizarse son la interfaz web y la interfaz de la aplicación móvil, estas interaccionan directamente con el *Web Browser* y con la *App IOS respectivamente.*

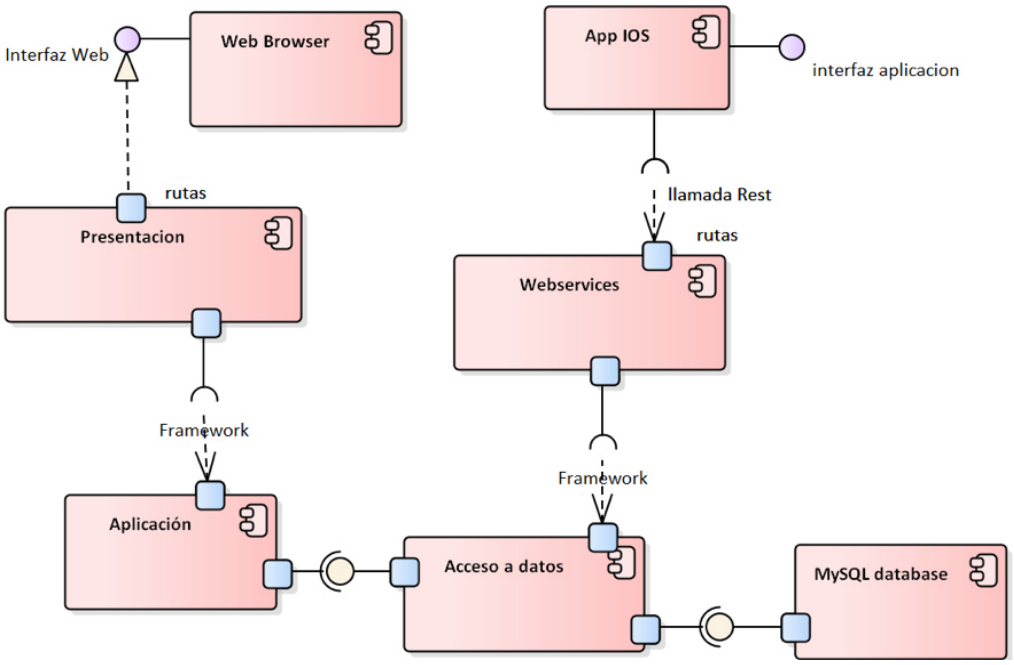


Figura 13. Diagrama de Componentes

1.1.1 Objetos

El diagrama de objetos es prácticamente el mismo que el de clases con la diferencia que este en vez de mostrar los tipos de datos muestra un ejemplo de estos datos como se puede observar en la Figura 14.

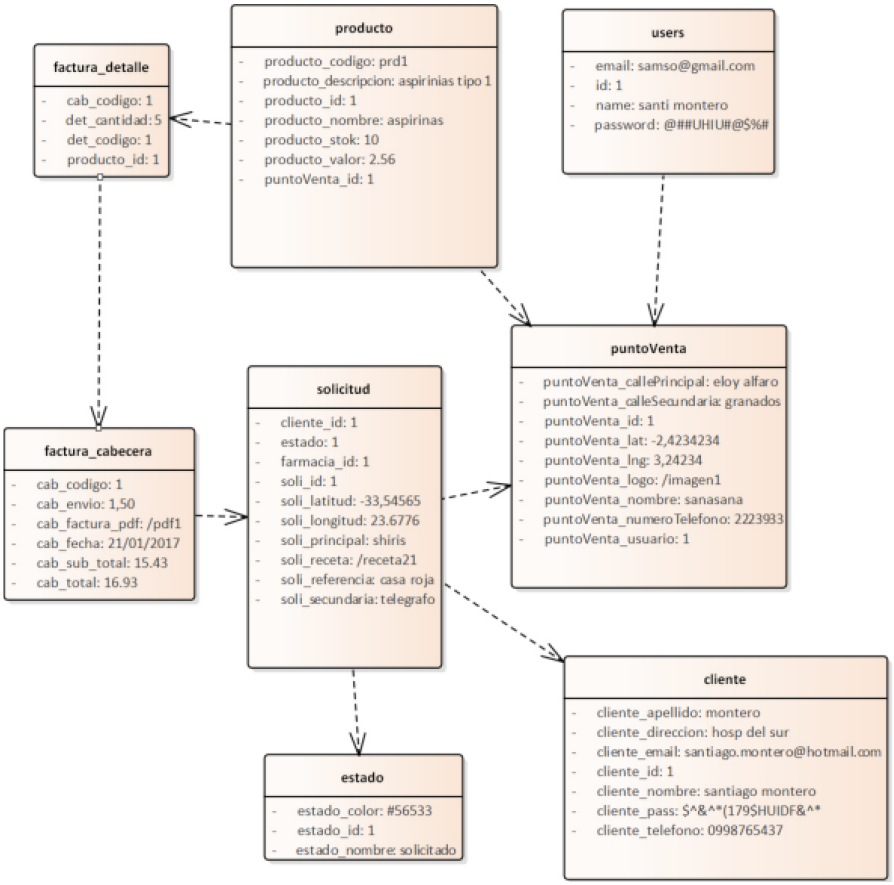


Figura 14. Diagrama de Objetos

5.3.3. Despliegue

Como se puede observar en la Figura 15 este diagrama describe el despliegue físico del sistema con la información del software y los hardware involucrados en este caso el servidor y los dispositivos que utilizaran el sistema que son un *Web Browser* por medio de una computadora y un dispositivo IOS es decir un *IPHONE*.

Se puede observar además que la comunicación entre el *WebServer* y el *Database Server* se lo realiza atreves de una interfaz de *DataBase* que es manejada en el proyecto actual por el Framework *Larave.l*

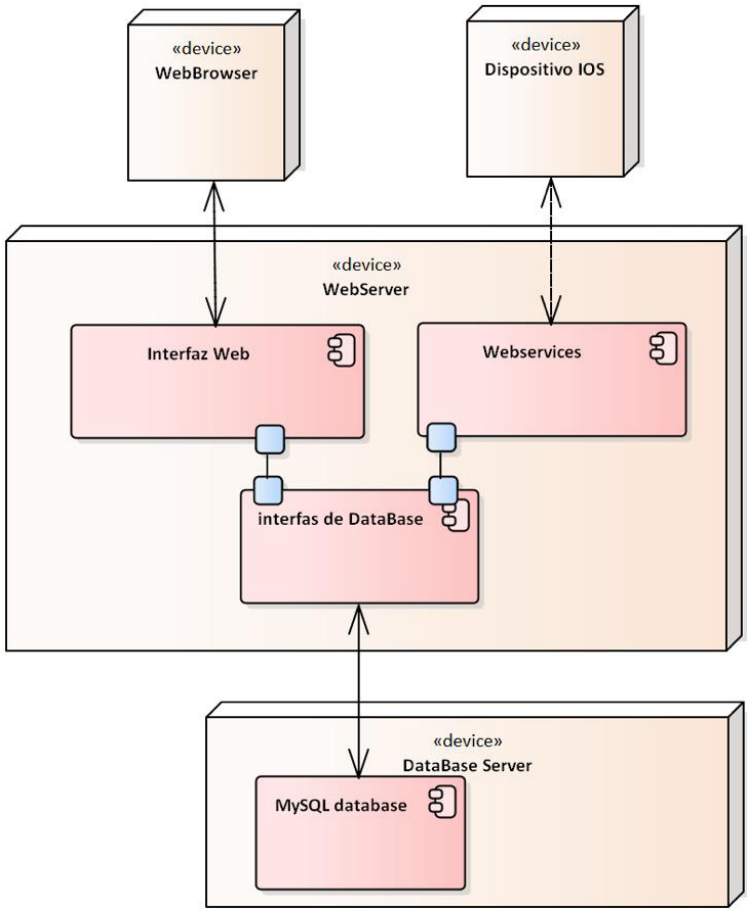


Figura 15. Diagrama de Despliegue

5.3.4. Casos de Uso

De acuerdo a la metodología UP para el desarrollo de un Software, su primer paso en el diseño del mismo seria la identificación de sus casos de uso. Los casos de uso se encargan de capturar los requerimientos funcionales del sistema y las relaciones que tienen entre ellas, además de sus actores involucrados.

En la Figura 16 se puede observar que los casos de uso y sus relaciones, que a continuación se detallan:

- Registro de usuario
- Autentificación de usuario
- Visualización de estados de solicitud
- Visualización de farmacias cercanas

Además, en la Figura 16 se observa las funciones asociadas al farmacéutico que son:

- Registro de usuario.
- Autentificación del usuario.
- Visualización de solicitudes y procesar las solicitudes para la creación de factura.
- Visualización de datos de la farmacia.
- manejo de catálogo de productos.
- procesar las solicitudes para la creación de factura.
- realizar pedido de la receta del Cliente.

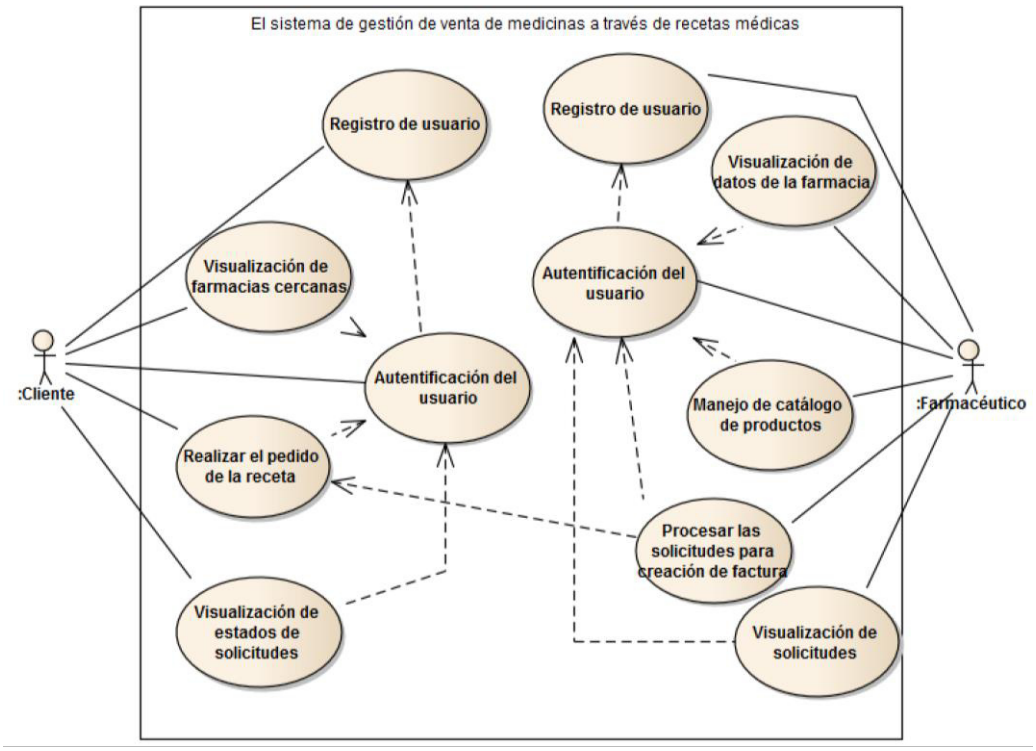


Figura 16. Casos de uso del sistema

5.3.5. Diagramas de Secuencia

Una vez identificados los casos de uso se procedió a describir la secuencia de acciones a realizarse para cumplir con los mismos. Para ello, se utilizó diagramas de secuencia, que facilitan dicha descripción de manera gráfica, además de mostrar las diferentes entidades que se involucran en cada una de estas funciones.

Los diagramas de secuencia que se muestran a continuación corresponden a los casos de uso principales dentro del aplicativo móvil.

- **Autentificación del usuario**

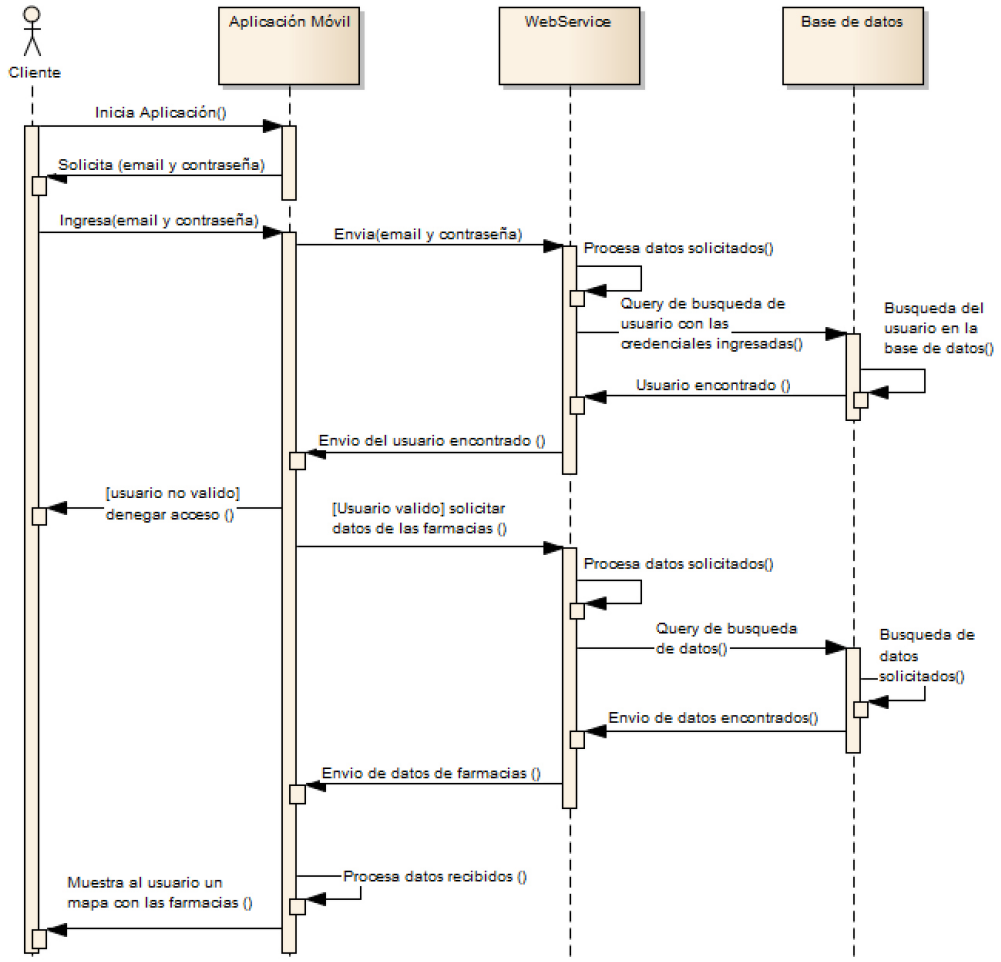


Figura 17. Diagrama de Secuencia del caso de uso Autentificación de usuario para el aplicativo móvil

En la Figura 17 se muestra el diagrama de secuencia del caso de uso “autentificación de usuario” donde se observa, el actor Cliente ingresa al aplicativo móvil con su email y *password*, el aplicativo envía esta información al servidor mediante *Webservices* donde validan que las credenciales sean correctas.

- **Realizar pedido de la receta.**

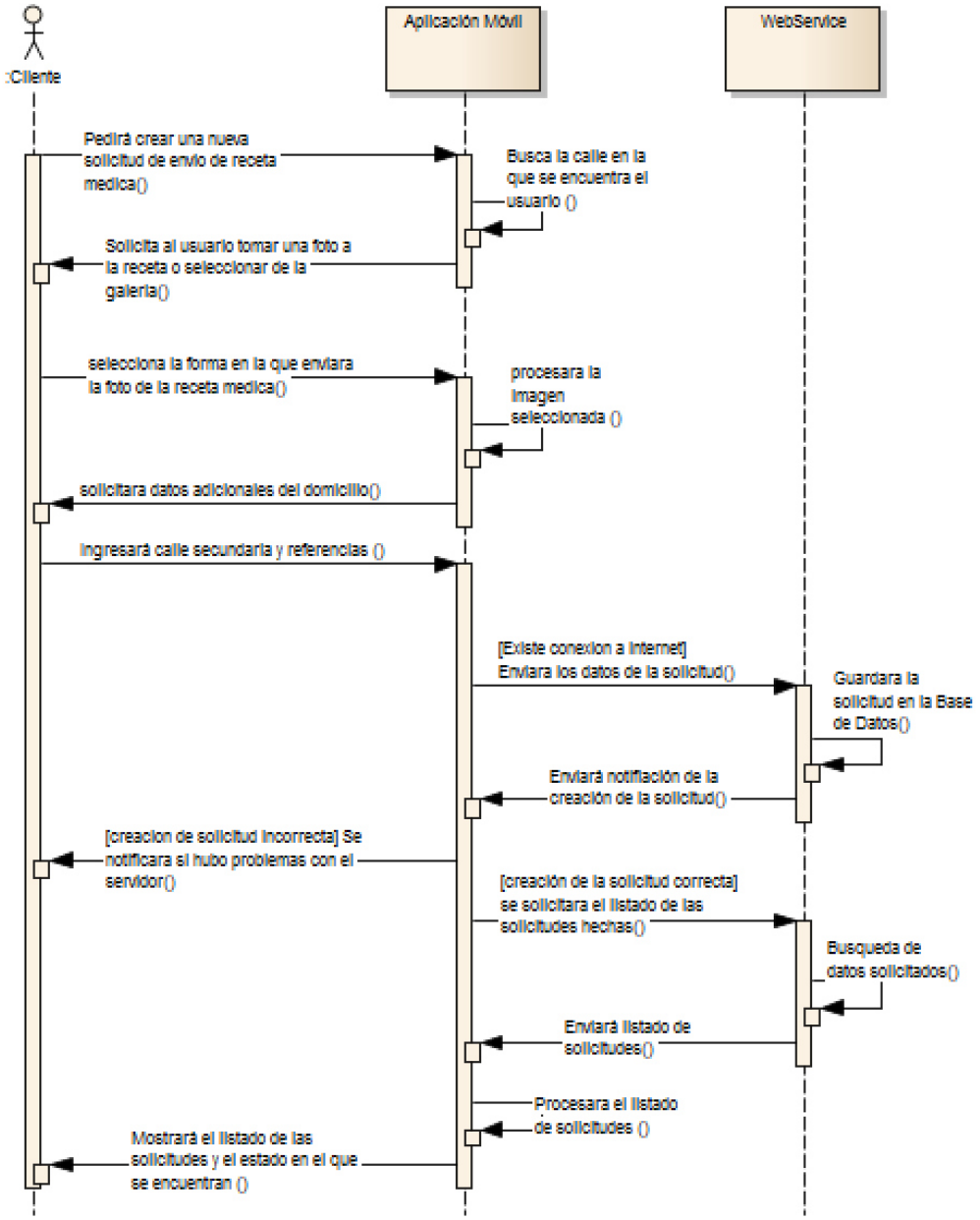


Figura 18. Diagrama de Secuencia del caso de uso de realizar pedido de la receta

En la Figura 18 se muestran el intercambio de mensajes entre los tres principales actores al momento de crear una nueva solicitud que en este diagrama serian el Cliente, aplicación móvil y los *WebServices*.

Se observa que el cliente entra a la interfaz de nueva solicitud del aplicativo móvil, este se encarga por medio de GPS detectar cual es la calle principal del cliente, además de solicitarle al usuario seleccionar una foto de la galería de imágenes o de capturar una fotografía en ese momento, también le pide información adicional del domicilio. Una vez hecho eso se procederá a enviar esta información al servidor por medio de *WebServices*.

- **Visualización de estados de solicitudes**

En la Figura 19 se observa que los actores principales continúan siendo Cliente, Aplicación móvil y *WebServices*. En la primera parte del diagrama se denota el proceso para solicitar el listado de las solicitudes hechas a las diferentes farmacias y los estados en los que se encuentran.

El proceso es totalmente en línea y el cliente podrá refrescar esta interfaz directamente desde el aplicativo con solo arrastrar el listado hacia abajo. Además, cada uno de los estados tendrá un color específico para mejor la interfaz gráfica y facilitar la interacción con el cliente al momento de la búsqueda de las últimas transacciones realizadas.

En la segunda parte del diagrama se analizó cuando el estado se encuentra pendiente de pago, la interfaz que se despliega cuando la solicitud está en este estado es la de la factura con dos opciones de aceptación o cancelación de pedido, las dos opciones proceden de la misma forma enviando por medio de Webservices un cambio de estado y una notificación por correo electrónico al farmaceuta.

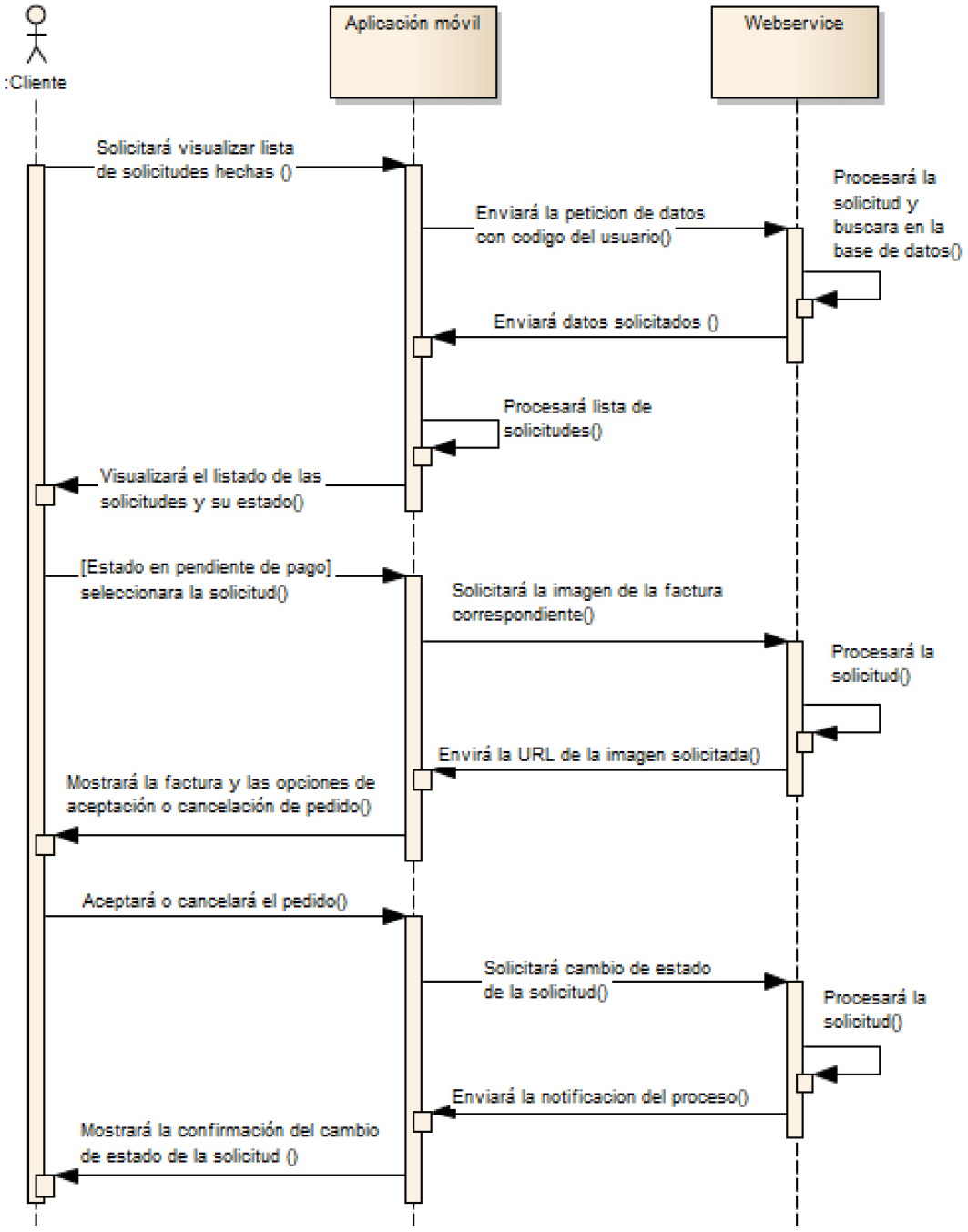


Figura 19. Diagrama de Secuencia del caso de uso de visualización de estados de solicitudes

Los diagramas de uso que se muestran a continuación son los principales para la página web y como interacciona con el servidor además de sus principales actores.

- **Visualización de datos de la farmacia**

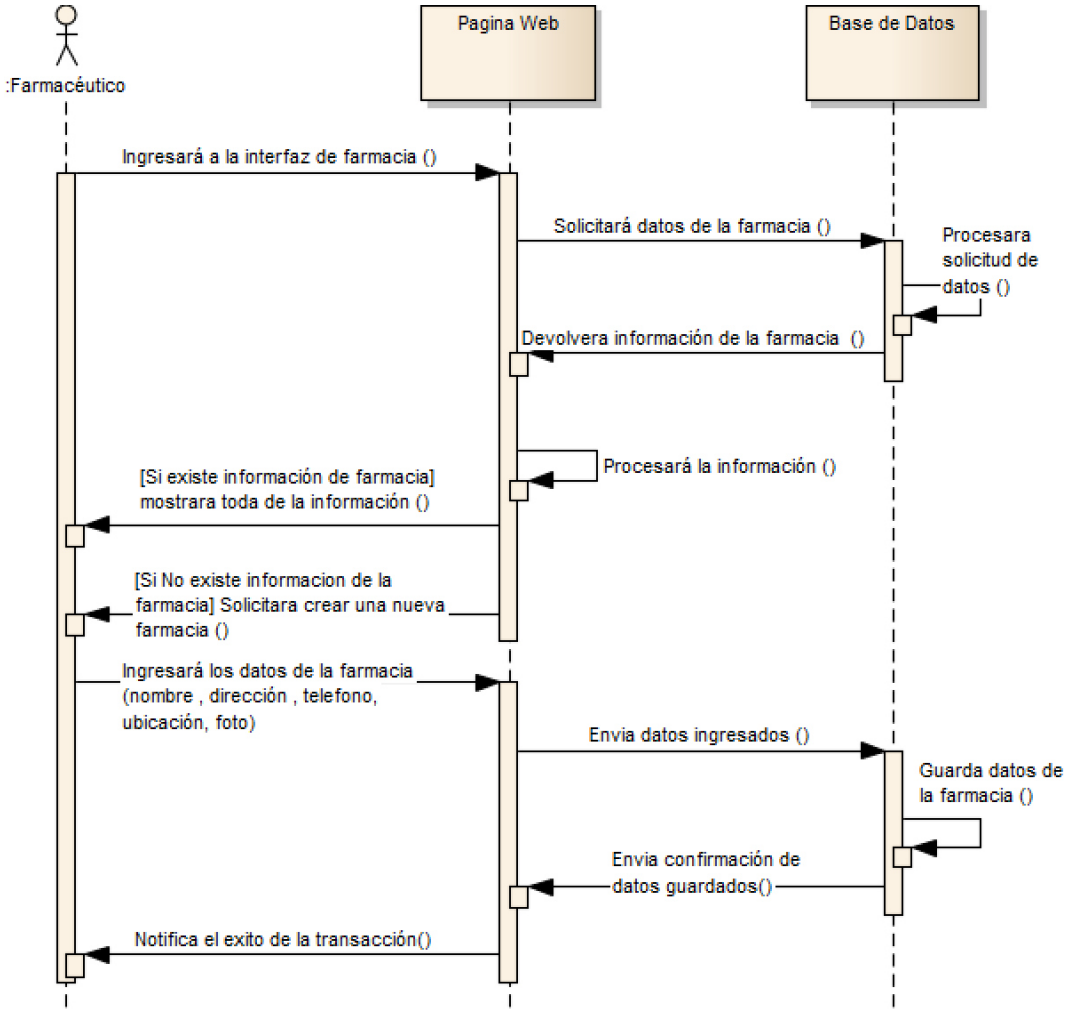


Figura 20. Diagrama de Secuencia del caso de uso de la visualización de datos de la farmacia

En la Figura 20 se describe el diagrama de secuencia del caso de uso “visualización de datos de la farmacia”. Una vez que el farmacéutico se haya registrado será redirigido a la pantalla de datos de la farmacia en el diagrama se describe como la página web solicita a la base de datos la información de esta farmacia si existe la desplegará al usuario en caso contrario le requerirá que ingrese la información básica del establecimiento como el nombre la dirección el teléfono y una imagen de la empresa además por medio de un *pluning* de *google-map* podrá ubicar donde se encuentra la farmacia en el mapa.

Una vez haya acabado de completar el formulario enviara la información a la base de datos para ser guardada y al ser un aplicativo sincrónico esperara la confirmación de éxito de la transacción.

- **Manejo de catálogo de productos**

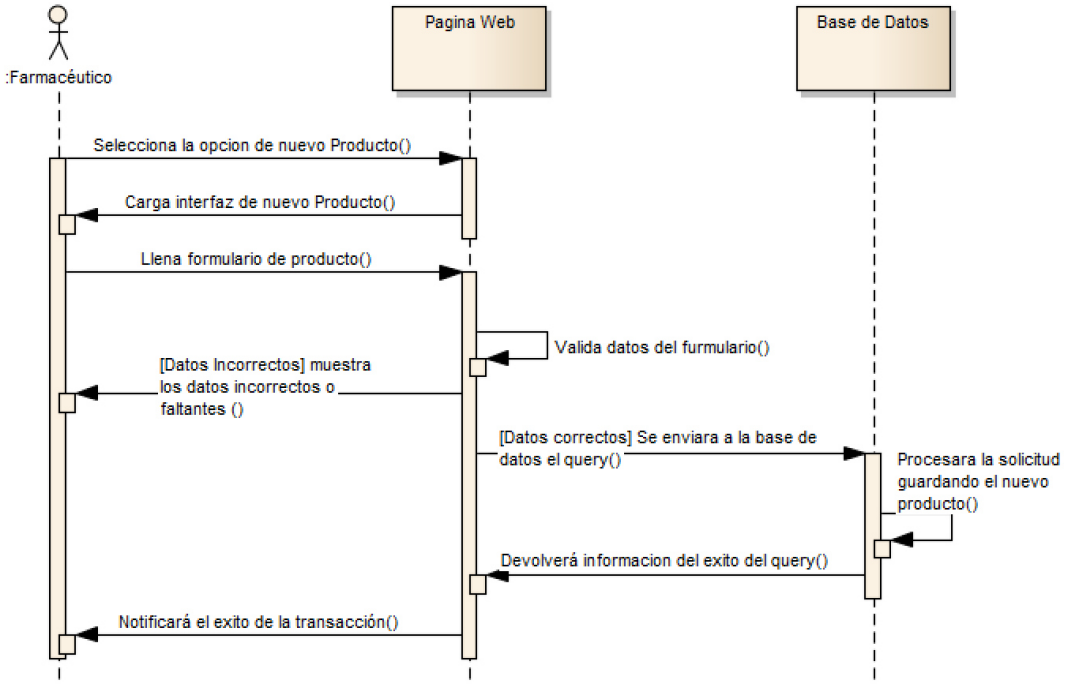


Figura 21. Diagrama de Secuencia del caso de uso del manejo de catálogo de productos

En la Figura 21 se muestra una de las opciones en el manejo de catálogo de productos, para ser precisos, al momento de crear un nuevo producto el farmaceuta interactuará con la página web un formulario, donde ingresará el nombre, el código, la cantidad, el precio y una breve descripción del producto.

Una vez completado el formulario, la página web validará que los datos ingresados sean correctos, de ser así, procederá a enviarlos para su almacenamiento en la base de datos, mostrando una notificación al usuario para informar que la transacción ha sido exitosa, caso contrario se informará si tiene que hacer modificaciones en los valores ingresados.

- **Procesar las solicitudes para creación de factura**

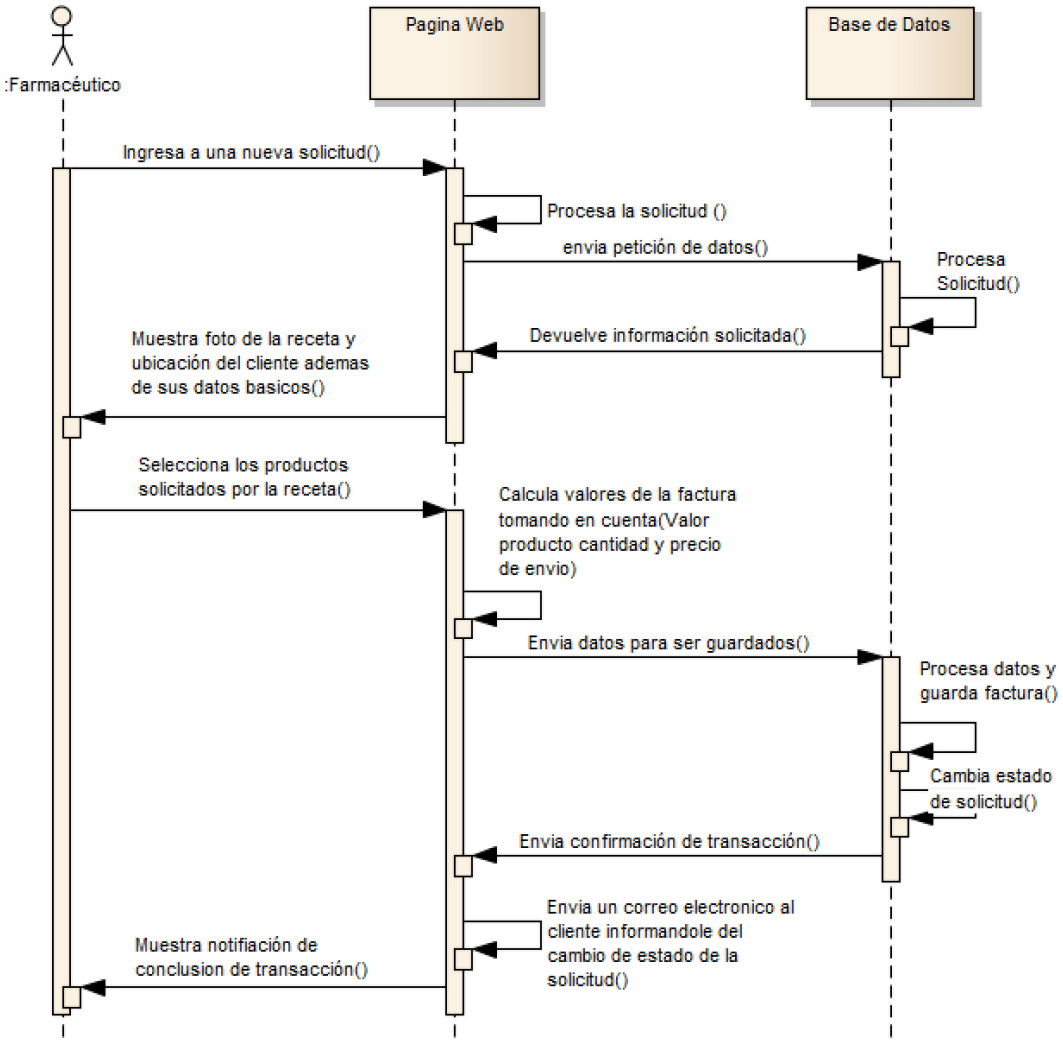


Figura 22. Diagrama de Secuencia del caso de uso al procesar las solicitudes para creación de factura

En la Figura 22 se muestra el diagrama de secuencia para “procesar las solicitudes para creación de factura” de una nueva solicitud, la página web a lo que ingrese a la interfaz de solicitud procesará la petición y solicitará a la base de datos toda la información necesaria.

La página web desplegará la imagen de la receta médica y en un mapa se mostrará la ubicación exacta del cliente para la entrega a domicilio. Una vez

hecho esto, el farmaceuta seleccionará los diferentes productos que consten en la receta y el aplicativo calculará el valor total.

Una vez terminada la factura se guardará en la base de datos y se enviará una notificación por correo al cliente que el estado de la solicitud cambio.

5.3.6. Definición de la Arquitectura

Una vez definido los diferentes casos de uso y sus diagramas de secuencia, se procedió a definir la arquitectura lógica del sistema, la cual estará en función de los casos de uso y los procesos mostrados en los diagramas de secuencias.

La arquitectura se diseñó basándose en el patrón de diseño MVC, como se explicó en el capítulo anterior.

En la Figura 23 se muestra la arquitectura lógica resultante del ambiente de prueba del sistema de gestión de venta de medicinas a través de recetas médicas, es presentado de una forma modular en el que cada módulo se relaciona con otros por medio de flujos de información.

En este patrón se puede observar las separaciones entre las Vistas o las interfaces de usuario con los Controladores y los modelos o lógica del negocio mismo que ejecutan las funciones del sistema.

Adicionalmente, se puede observar que los diferentes módulos se encuentran agrupados en conjuntos ya que la arquitectura base del proyecto es Cliente Servidor, donde el cliente es el aplicativo móvil y el servidor estará conformado por los *WebServies* y la página web. Se observa además que algunos módulos del servidor se encuentran en el conjunto del cliente ya que ambos componentes tienen un patrón MVC.

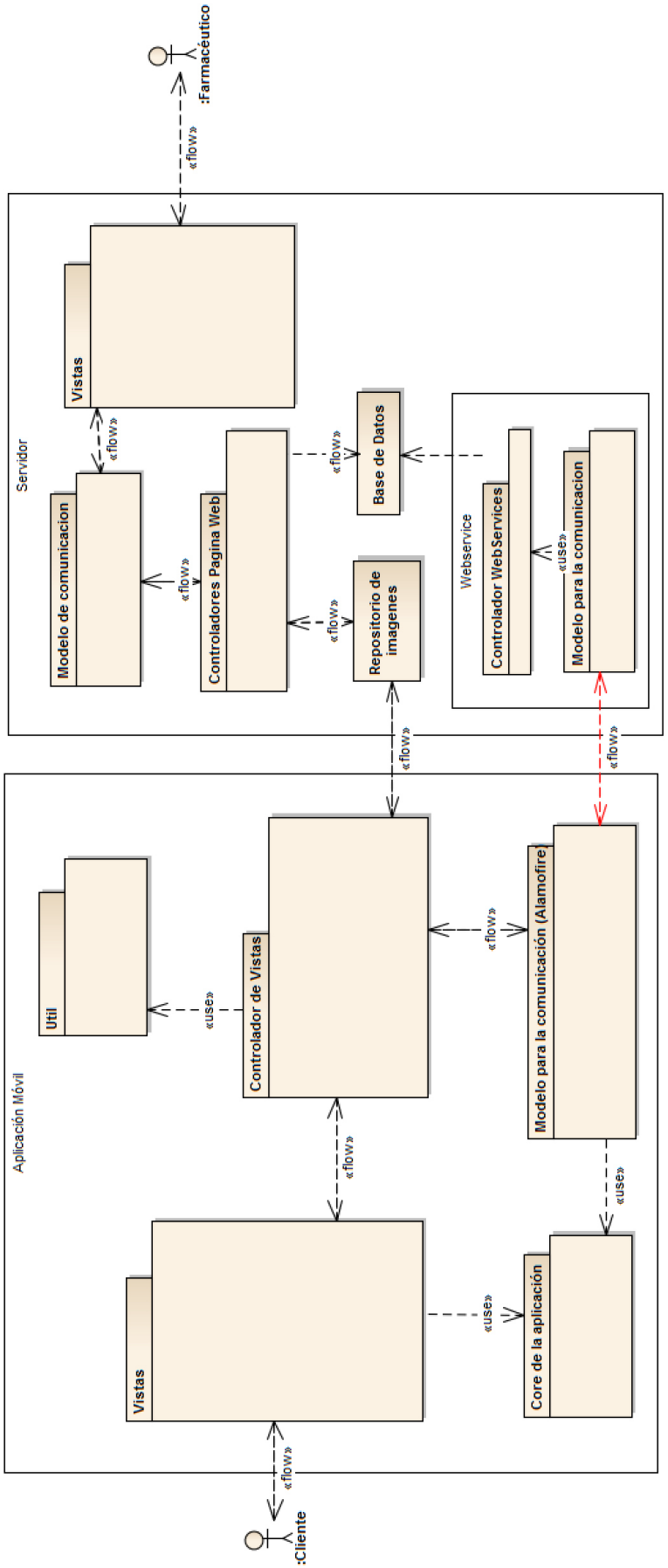


Figura 23. Arquitectura lógica del ambiente de prueba del sistema de gestión de venta de medicinas a través de recetas médicas

Para un mejor entendimiento de esta arquitectura se procederá a explicar cada uno de los módulos.

- **Vistas**

  En este módulo se encuentran en las interfaces de usuario para el aplicativo móvil y para página web. Por su parte Swift tiene un diseñador proprio para estas interfaces y lo maneja en forma de diagrama de navegación. En cambio, la página web trabaja con HTML5 y JQUERY.

- **Controladores**

  Este módulo también se encuentra presente en ambos conjuntos y es el encargado de interaccionar directamente con las vistas con respecto al aplicativo móvil, en cambio hablando del servidor tenemos dos controladores uno para las vistas de la página web y otro controlador para las funciones de los *WebServices*. En ambos casos se comunican directamente con el modelo de comunicaciones.

- **Modelo para la comunicación**

  Este módulo está presente en el aplicativo móvil como en el servidor, considerando que el modelo para la comunicación en el *WebServices* está relacionado directamente con su contraparte en el cliente para el envío y recepción de información. A su vez este mismo modelo se encuentra como intermediario en la comunicación entre vista y controlador de la página web.

- **Útil**

  Este módulo es una herramienta que incorpora funciones repetitivas y por motivos de calidad de software y evitar redundancia de código es utilizado. Es usado únicamente con los controladores del aplicativo móvil como se puede observar en la Figura 23.

- **Core de la aplicación**

  Este módulo contiene las librerías necesarias para el correcto funcionamiento del sistema y se encarga principalmente de ejecutar todos los procesos complejos e hilos de este como se observa en la Figura 23 es un módulo básico para la correcta comunicación.

- **Base de Datos**

  Este módulo será el encargado de guardar toda la información del sistema y se encuentra únicamente en el servidor los controladores de la página web y de los *WebServices* son los únicos que interaccionan directamente con él.

- **Repositorio de imágenes**

  Por último en este módulo se guardarán todas las imágenes del sistema y estará únicamente en el servidor los controladores del aplicativo móvil y de la página web interaccionarán con él para la visualización de su contenido.

5.3.7. Definición de Escenarios

Una vez descritos los casos de uso y sus diagramas de secuencia se proceden a definir los escenarios que presenta el sistema de gestión de venta de medicinas a través de recetas médicas, con los cuales tenemos dos actores humanos que son: el cliente y el farmaceuta.

Al considerar los casos de uso para el Cliente como objeto base para esta definición de escenarios, se puede observar que tiene seis interfaces principales y una secundaria, en cambio tomando el actor Farmacéutico como objeto base se puede decir que tiene cinco escenarios principales y tres secundarios.

Los escenarios mencionados anteriormente para el cliente se pueden observar en la Figura 24 en un diagrama de navegación de interfaces de usuario, donde se describe el modo en el que el Cliente deberá desplazarse por el aplicativo móvil para acceder a cada una de las interfaces.

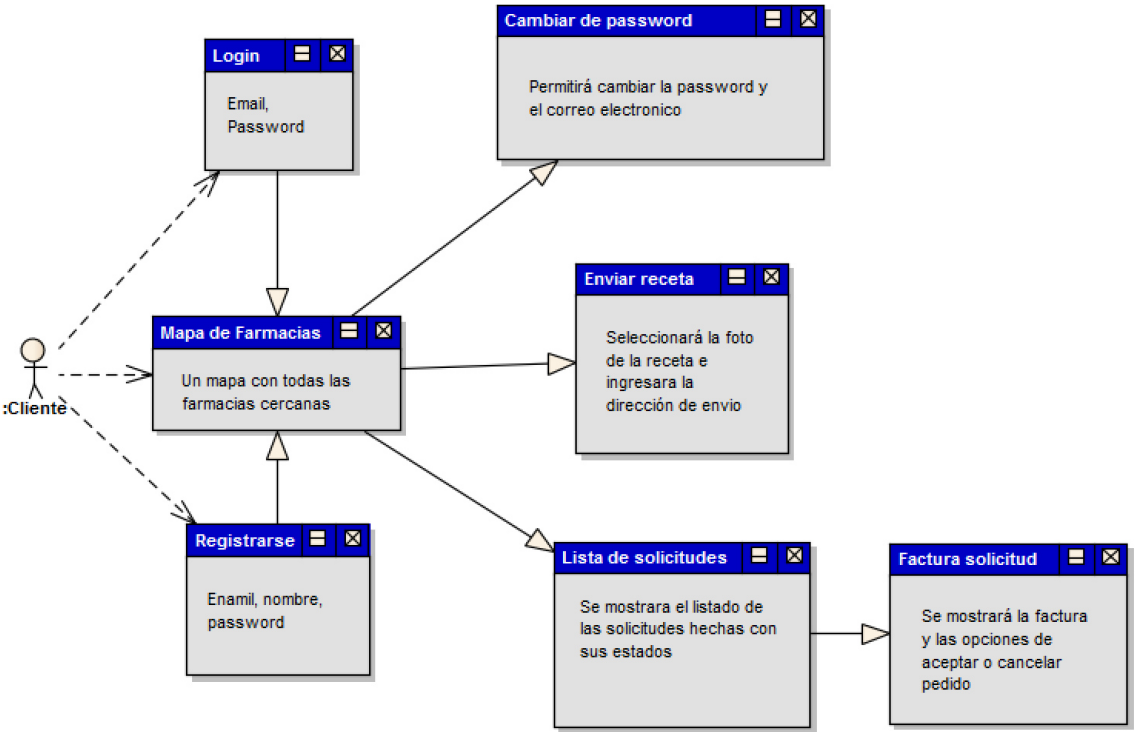


Figura 24. Diagrama de navegación de escenarios para el Cliente

En la Figura 24 se puede observar que para interactuar con las interfaces principales deberá estar registrado y autentificado. Las credenciales del usuario se guardarán en la memoria del dispositivo como variables de sesión, por lo tanto solo si se llegase a cerrar sesión el cliente deberá volver a autentificarse, caso contrario aún cuando el aplicativo se cierre no volverá a pedir credenciales al momento de abrirlo.

Descripción de escenarios para el cliente.

Los escenarios que se presentan en la Figura 24 se detallaran a continuación.

- **Registrarse**

  En este escenario el cliente deberá llenar un formulario con sus datos personales como son nombre, dirección, teléfono, email, y una contraseña. Al momento de mandar a guardar la información, el aplicativo lo enviará a la página principal donde se encuentra el mapa y sus datos de autentificación se guardaran automáticamente en el móvil.

- **Login**

  En esta interfaz de autentificación del usuario donde ingresara únicamente su email y contraseña cuando el aplicativo se lo solicite de no tener una sesión abierta en caso contrario esta interfaz viene omitida.

- **Mapa de farmacias**

  Este es el escenario principal donde en un mapa se desplegarán todas las farmacias cercanas y donde el cliente podrá seleccionar la de su mayor agrado, además la interfaz tiene los enlaces a los demás escenarios como son cambio e contraseña lista de solicitudes y cierre de sesión.

- **Cambio de password**

  Dentro de este escenario podremos actualizar nuestro correo electrónico y nuestra contraseña.

- **Lista de solicitudes**

  En esta última interfaz principal el cliente podrá ver todas sus solicitudes hechas a las farmacias, además este listado mostrará el estado en orden cronológico de la más reciente a la más antigua, con su respectivo estado con un color característico para mejor la experiencia del usuario al momento de consultar las transacciones hechas.

- **Factura Solicitud**

  Esta interfaz secundaria se podrá visualizar siempre y cuando no esté en un estado de *cancelado* o *solicitado,* y cliente podrá observar la factura hecha por el farmacéutico, además que dará las opciones de aceptación de pago en efectivo o cancelación de pedido. La opción de pedido permanecerá hasta que el estado cambie a *enviado* en ese momento el cliente no podrá más cancelar el pedido.

Los escenarios para el actor Farmaceuta se pueden observar en la Figura 25, también en un diagrama de navegación de interfaces de usuario, donde se describe el modo en el que el Farmaceuta deberá desplazarse por la página web para acceder a cada una de las ventanas.

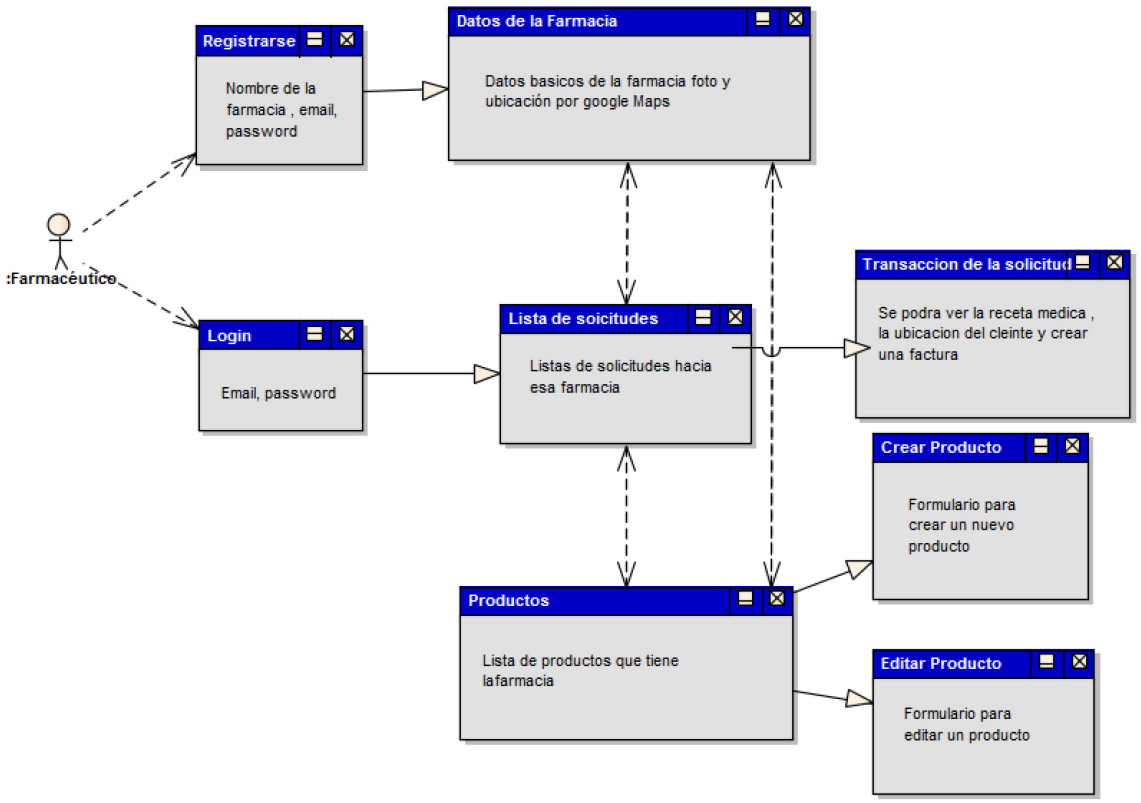


Figura 25. Diagrama de navegación de escenarios para el Farmacéutico

Al igual que para el aplicativo móvil se puede observar será necesario registrarse al sistema o proceder con la autentificación del usuario si ya se poseen credenciales para poder acceder a las diferentes ventas de la página web.

Además, hay que considerar que al ser una página web, un menú principal dejará interactuar entre las tres ventanas principales que son datos de la farmacia, listado de solitudes y productos como se muestra en la Figura 25.

Descripción de escenarios para el Farmacéutico.

- **Registrarse**

  En esta interfaz el usuario deberá ingresar su nombre correo electrónico una contraseña para poder tener acceso al sistema. Una vez concluido el registro será dirigido a la página de los datos de la farmacia.

- **Login**

  Esta será siempre la primera interfaz que los farmacéuticos se encontrarán cuando entren a la página web, si el navegador es confiable tendrán la posibilidad de guardar sus credenciales para agilizar el proceso de autentificación para futuras ocasiones, una vez ingrese en el sistema la ventana principal que se mostrara será la del listado de solicitudes pendientes.

- **Lista de solicitudes**

  En este escenario de desplegaran el listado de solicitudes hechas a esa farmacia se mostrarán en colores diferentes y tendrá opciones de interacción distintas dependiendo el estado en el que se encuentren. Estas opciones podrán ser cancelar pedido ver pedido y enviar pedido.

- **Transacciones de la solicitud**

  Esta interfaz se desplegará cuando en el listado de solicitudes se elija ver pedido, aquí se puede ver los detalles de la solicitud como la foto de la receta médica la ubicación y datos básicos para la factura del cliente, además se podrá generar una factura con los productos requeridos por la receta médica y el aplicativo se encargará de calcular el valor total de la transacción.

- **Datos de la farmacia**

  Esta será la ventana que dejará gestionar la información de la farmacia como datos de ubicación geográfica y datos básicos como nombre dirección y teléfono, además podrán subir una foto de su logo o local.

- **Productos**

  Esta será el escenario donde se mostrará el listado de todos los productos que tenga la farmacia y también podrá crear nuevos productos editarlos y borrarlos.

  - **Crear producto**

    Esta será una pequeña ventana modal donde al completar un formulario con nombre, cantidad, precio, código y una breve descripción se podrá generar un nuevo producto.

  - **Editar producto**

    Como el nombre lo dice en esta ventana modal se podrán actualizar los valores del producto como cambiar de precio o aumentar el stock.

5.4. Implementación

Una vez concluido el diseño del sistema se procedió a su desarrollo, para ello se utilizaran dos herramientas. Para el aplicativo móvil al ser para sistema IOS se usará XCODE una herramienta propietaria de Apple que usa el lenguaje Swift 3 además de ser gratuita para todos los usuarios de OSX. A su vez para la página web y los *WebSrvices* se usará IDE *NetBeans* que también es gratuito y su lenguaje de programación *Open Source* es PHP*,* además de ser totalmente compatible para publicaciones en servidores Apache Tomcat.

5.4.1. Construcción de los Módulos de la Arquitectura

5.4.1.1. Aplicación Móvil

5.4.1.1.1. Vistas aplicaciones Móviles

Las interfaces en el aplicativo móvil con lenguaje SWIFT se representan en un diagrama de navegación llamado *main.storyboard* como se muestra en la Figura 26.

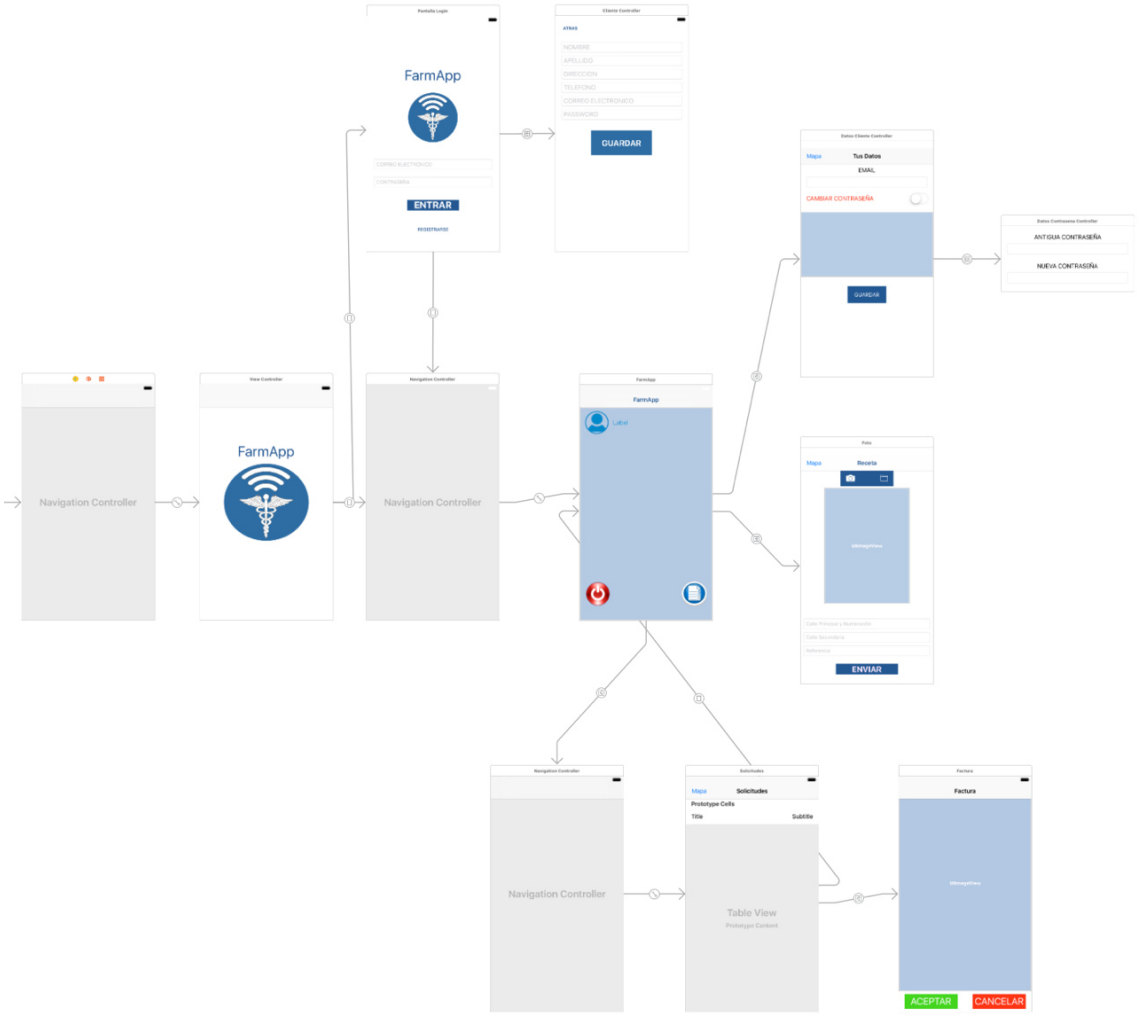


Figura 26. Diagrama de navegación del aplicativo móvil

Estas interfaces están relacionadas directamente con un único controlador y para la transacción entre una u otra interfaz se las configura por código. Además, en este tipo de interfaces son muy importante las dimensiones de cada objeto insertado ya que hay diferentes tamaños de pantalla y las interfaces tiene que ser adaptativas por eso SWIFT maneja *constrain of object* a fin que las dimensiones de cada objeto sean variables pero las distancias entre marcos y otros objetos sean constantes.

Las vistas que contiene el aplicativo móvil se enlistaran a continuación divididos en escenarios principales y secundarios:

1) Registrarse
2) Login
3) Pantalla principal (Mapa de las farmacias)
4) Cambio de contraseña
5) Crear nueva solicitud
6) Lista de solicitudes
   a) Factura de la solicitud

5.4.1.1.2. Controladores para el aplicativo móvil

Los controladores en el aplicativo móvil son programados en SWIFT 3 y su nombre al momento de crearlos se define como *UiViewController.swift* posteriormente por motivos de personalización se ponen un nombre que haga referencia a su vista.

Un controlador puede ser solamente usado por una interfaz, misma que tomará el nombre de dicho controlador. Un dato importante en el orden de archivos en un aplicativo en SWIFT es que sin importar las carpetas y sub carpetas que manejemos en XCODE, en *FINDER (explorador de archivos en Mac OS)* al momento de ver los archivos del programa no se encuentra ninguna carpeta de las creadas con la herramienta.

También se tiene que considerar que cada objeto que se cree en la vista deberá ser instanciado en el controlador para poder ser usado o leído, la mayoría de objetos en Swift además de dar la opción de ser instanciados también pueden generar funciones al momento de usarse como pueden ser los botones los Textbox o los ListView.

A continuación, se enlista los controladores usados para este aplicativo móvil creado en SWIFT:

1) PantallaLogin.swift
2) pantaPrincipal.swift
3) pantaFoto.swift
4) SolicitudesController.swift
5) FacturaControler.swift
6) ClienteController.swift
7) ViewController.swift
8) DatosClienteController.swift

5.4.1.1.3. Comunicación entre aplicativo Móvil y WebServices

Este módulo está presente en el aplicativo móvil, en los *WebServices* y en la página Web pero tienen elementos y funciones distintas. Para facilitar la comunicación entre dichos elementos se utiliza la estructura JSON.

La comunicación entre el aplicativo móvil y el servidor está gestionado por los WebServices.

Para la comunicación en Swift se agregó un *Framework* de nombre *Alamorfire* que facilita el trabajo para el programador, este que se encuentra en su versión 4.0 y es totalmente compatible con la última versión de *Swift* 3 y está totalmente diseñado para el consumo de servicios web tipo *API*.

Por su parte el *WebService* al igual que la página web se desarrolló mediante el *Framework Laravel* para agilizar el proceso de implementación del ambiente de pruebas.

La comunicación entre estos dos puntos es mediante el consumo de los *WebServices* creados, haciendo que el aplicativo móvil realice llamados *POST* y *GET* dependiendo los requerimientos y los servicios después de recibir la solicitud y hacer los procesos correspondientes devolverá su respuesta en formato *JSON*.

JSON al ser un formato estructurado creado para el intercambio de datos se presenta de forma ligera y fácil entendimiento, por tal motivo se lo utiliza dentro del modelo de comunicación tanto en el WebServices para envío de información como en el aplicativo móvil para la lectura de datos.

En la **¡Error! No se encuentra el origen de la referencia.** podemos ver la arquitectura de comunicación, desde la petición *HTTP* de móvil al servidor *PHP* que a su vez transacción con el motor de base de datos *MySQL* y devuelve una respuesta en formato *JSON*.

Figura 27. Diagrama de peticiones HTTP y respuestas JSON en un servidor PHP

Algo de tomar en cuenta es que JSON en SWIFT no puede ser convertido directamente a objeto por lo tanto siempre es necesario un barrido de información e ir instanciando uno por uno los datos dentro de un ciclo.

Al igual que para la página web con Laravel los WebServices necesitan rutas de acceso, pero a diferencia de una página web funciones como EDIT son

eliminadas como se muestra en la **¡Error! No se encuentra el origen de la referencia.**

Method	URI	Name	Action	Middleware
GET|HEAD	/		Closure	web
GET|HEAD	clientes	clientes.index	App\Http\Controllers\ClientesController@index	web
POST	clientes	clientes.store	App\Http\Controllers\ClientesController@store	web
POST	clientes/actualizar	clientes.actualizar	App\Http\Controllers\ClientesController@actualizar	web
PUT|PATCH	clientes/{clientes}	clientes.update	App\Http\Controllers\ClientesController@update	web
GET|HEAD	clientes/{clientes}	clientes.show	App\Http\Controllers\ClientesController@show	web
DELETE	clientes/{clientes}	clientes.destroy	App\Http\Controllers\ClientesController@destroy	web
POST	farmacias	farmacias.store	App\Http\Controllers\FarmaciasController@store	web
GET|HEAD	farmacias	farmacias.index	App\Http\Controllers\FarmaciasController@index	web
GET|HEAD	farmacias/prueba	farmacias.show	App\Http\Controllers\FarmaciasController@show	web
POST	solicitud	solicitud.store	App\Http\Controllers\SolicitudController@store	web
GET|HEAD	solicitud	solicitud.index	App\Http\Controllers\SolicitudController@index	web
GET|HEAD	solicitud/cancelar/{solicitud}	solicitudes.cancelar	App\Http\Controllers\SolicitudController@cancelar	web
GET|HEAD	solicitud/enviar/{solicitud}	solicitudes.enviar	App\Http\Controllers\SolicitudController@enviar	web
GET|HEAD	solicitud/{solicitud}	solicitud.show	App\Http\Controllers\SolicitudController@show	web
PUT|PATCH	solicitud/{solicitud}	solicitud.update	App\Http\Controllers\SolicitudController@update	web
DELETE	solicitud/{solicitud}	solicitud.destroy	App\Http\Controllers\SolicitudController@destroy	web

Figura 28. Rutas de los WebServices

5.4.1.1.4. Útil

Este módulo es usado solo en el aplicativo móvil y es una carpeta que contiene un conjunto de funciones repetitivas y clases y pueden ser llamadas desde cualquier parte del código para su uso.

5.4.1.2. Core

Este módulo tiene las librerías que se ejecutan en los procesos complejos del sistema. Dentro de estas librerías se encuentran procesos y sub-procesos que se encargan de la comunicación con la interfaz en el aplicativo móvil, y la creación de objetos en las diferentes pantallas. Todas estas clases son propias de SWIFT se ejecutan como HILOS y solo se puede consumir mas no modificarlas.

5.4.1.3. Servidor

5.4.1.3.1. Vistas Página Web

La visualización de la aplicación web se la realizará por medio de navegadores de internet, utilizando un lenguaje de programación llamado HTML5 para las

interfaces que permitirán al usuario utilizar el aplicativo en cualquier dispositivo sin importar el navegador que use.

Para este sistema, se usa un framework llamado LARAVEL que ofrece varias facilidades al momento de programar, por ejemplo, cuando se habla de interfaces de usuario, el framework trabaja con Blade que es un pluning proprio de Laravel que por medio de clases y funciones logra reducir el código en las vistas hasta en una tercera parte, facilita el trabajo con vistas parciales y su extensión para los archivos es blade.php.

Los archivos blade.php que corresponden a cada una de las interfaces y sub-interfaces de la página web se listan a continuación divididos en categorías:

1) Auth
   a) Login.blade.php
   b) Register.blade.php
2) Email
   a) Correo.blade.php
3) Farmacia
   a) Add.blade.php
   b) editPventa.blade.php
4) Producto
   a) Créate.blade.php
   b) Edit.blade.php
   c) Index.blade.php
5) Solicitud
   a) Edit.blade.php
   b) Factura.blade.php
   c) Index.blade.php
   6) Layout.blade.php

5.4.1.3.2. Controlador de vistas

Al ser el aplicativo móvil y la página web arquitecturas MVC ambas tendrán controladores aun cuando se programarán en lenguajes totalmente diferentes. Además, tenemos que tener en cuenta que los Webservices también tendrán sus propios controladores que serán los encargados de las validaciones y respuestas a los aplicativos móviles.

Controladores para la página web

Los controladores creados por Laravel tiene como extensión Controller.php y son enteramente creados en lenguaje PHP además se encuentran por *default* en la dirección app/http/controller.

Laravel ofrece la ventaja de poder crear controladores con una sola sentencia del terminal *"php artisan make:controller EjemploController",* además es obligatorio crear las rutas para dicho controlador en el archivo routes.php, las rutas en mención hacen referencia a una función específica y éstas a la vez a un método. Estas funciones tienen métodos distintos y dependerán de la función en la que se ejecuten, por ejemplo: GET es un método que devolverá información en una lista, POST es para el envío de un formulario con datos que por motivos de seguridad no pueden ir concatenados en una URL, DELETE como su nombre lo indica se utiliza para la eliminación de datos y por último PUT que se emplea al momento de tener que actualizar algún registro. Por lo tanto, los controladores normalmente se crean con funciones pre establecidas como se puede observar en el ejemplo de la Figura 23.

```
POST      | solicitud                 | solicitud.store   | App\Http\Controllers\SolicitudesController@store
GET|HEAD  | solicitud                 | solicitud.index   | App\Http\Controllers\SolicitudesController@index
GET|HEAD  | solicitud/create          | solicitud.create  | App\Http\Controllers\SolicitudesController@create
GET|HEAD  | solicitud/{solicitud}     | solicitud.show    | App\Http\Controllers\SolicitudesController@show
DELETE    | solicitud/{solicitud}     | solicitud.destroy | App\Http\Controllers\SolicitudesController@destroy
PUT|PATCH | solicitud/{solicitud}     | solicitud.update  | App\Http\Controllers\SolicitudesController@update
GET|HEAD  | solicitud/{solicitud}/edit | solicitud.edit   | App\Http\Controllers\SolicitudesController@edit
```

Figura 29. Ejemplo de rutas de un controlador

Por último, se debe hacer referencia a las librerías necesarias para la comunicación con la base de datos y la autentificación.

A continuación, se indicarán los controladores usados para la página web:

1) Auth
   a) AuthController.php
   b) PasswordController.php
2) FarmaciaController.php
3) ProductosController.php
4) SolicitudesController.php

Controladores para los WebServices

Estos controladores son creados de igual manera que los utilizados en la página web, con la única diferencia que no interactúan con las vistas.

Además, en dichos servicios web no se utilizan los métodos DELETE y PUT, se procede a eliminarlos puesto que se crean automáticamente y en el caso de requerir la eliminación o actualización de un archivo se usará el método POST.

5.4.1.3.3. Comunicación página Web

Laravel para la comunicación de sus aplicativos, usa un archivo llamado routes.php, este será el encargado del middleware entre las interfaces y los controladores. Adicionalmente, se emplea una arquitectura Cliente- Servidor mediante el internet, por lo tanto, se requiere que el protocolo de transacciones HTTP sea el estándar en lo que se refiere a navegación web.

La comunicación entre la página web y el aplicativo móvil no se ejecuta de forma directa pero comparten la información en la base de datos, su forma de interactuar es mediante notificaciones vía email.

Las rutas utilizadas para las interfaces web se muestran en la Figura 30.

Method	URI	Name	Action	Middleware
GET|HEAD	/	home	App\Http\Controllers\SolicitudesController@index	web,auth
POST	ajaxLoadProducto		App\Http\Controllers\SolicitudesController@LoadProducto	web,auth
POST	auth/login	auth/login	App\Http\Controllers\Auth\AuthController@postLogin	web,guest
GET|HEAD	auth/login		App\Http\Controllers\Auth\AuthController@getLogin	web,guest,guest
GET|HEAD	auth/logout	auth/logout	App\Http\Controllers\Auth\AuthController@getLogout	web
POST	auth/register	auth/register	App\Http\Controllers\Auth\AuthController@postRegister	web,guest
GET|HEAD	auth/register		App\Http\Controllers\Auth\AuthController@getRegister	web,guest,guest
POST	farmacia/add	farmacia/add	App\Http\Controllers\FarmaciaController@store	web,auth
GET|HEAD	farmacia/add	farmacia/add	App\Http\Controllers\FarmaciaController@index	web,auth
GET|HEAD	home	solicitud	App\Http\Controllers\SolicitudesController@index	web,auth
POST	producto	producto.store	App\Http\Controllers\ProductosController@store	web,auth
GET|HEAD	producto	producto.index	App\Http\Controllers\ProductosController@index	web,auth
GET|HEAD	producto/create	producto.create	App\Http\Controllers\ProductosController@create	web,auth
DELETE	producto/{producto}	producto.destroy	App\Http\Controllers\ProductosController@destroy	web,auth
PUT|PATCH	producto/{producto}	producto.update	App\Http\Controllers\ProductosController@update	web,auth
GET|HEAD	producto/{producto}	producto.show	App\Http\Controllers\ProductosController@show	web,auth
GET|HEAD	producto/{producto}/edit	producto.edit	App\Http\Controllers\ProductosController@edit	web,auth
POST	solicitud	solicitud.store	App\Http\Controllers\SolicitudesController@store	web,auth
GET|HEAD	solicitud	solicitud.index	App\Http\Controllers\SolicitudesController@index	web,auth
GET|HEAD	solicitud/create	solicitud.create	App\Http\Controllers\SolicitudesController@create	web,auth
GET|HEAD	solicitud/{solicitud}	solicitud.show	App\Http\Controllers\SolicitudesController@show	web,auth
DELETE	solicitud/{solicitud}	solicitud.destroy	App\Http\Controllers\SolicitudesController@destroy	web,auth
PUT|PATCH	solicitud/{solicitud}	solicitud.update	App\Http\Controllers\SolicitudesController@update	web,auth
GET|HEAD	solicitud/{solicitud}/edit	solicitud.edit	App\Http\Controllers\SolicitudesController@edit	web,auth

Figura 30. Rutas de la Página Web

5.4.1.3.4. Base de Datos

Tanto los WebServices como la página web interaccionan directamente con la misma base de datos, en pocas palabras existirá una única base de datos en MySQL para toda la transaccionalidad del sistema.

En la Figura 31 se muestra el modelo de la base de datos del sistema y sus relaciones, las tablas creadas son las siguientes:

- Users
- PuntoVenta
- Cliente
- Producto
- Solicitud
- Estado
- Factura_cabecera
- Factura_detalle

Para todas las contraseñas del sistema se manejó encriptación MD5 ya que esta es utilizada directamente por *Laravel* haciendo nuestro sistema más seguro y de

fácil programación, además que maneja funciones propias de validación de encriptación, ya que MD5 es un encriptado cifrado imposible de des-encriptar.

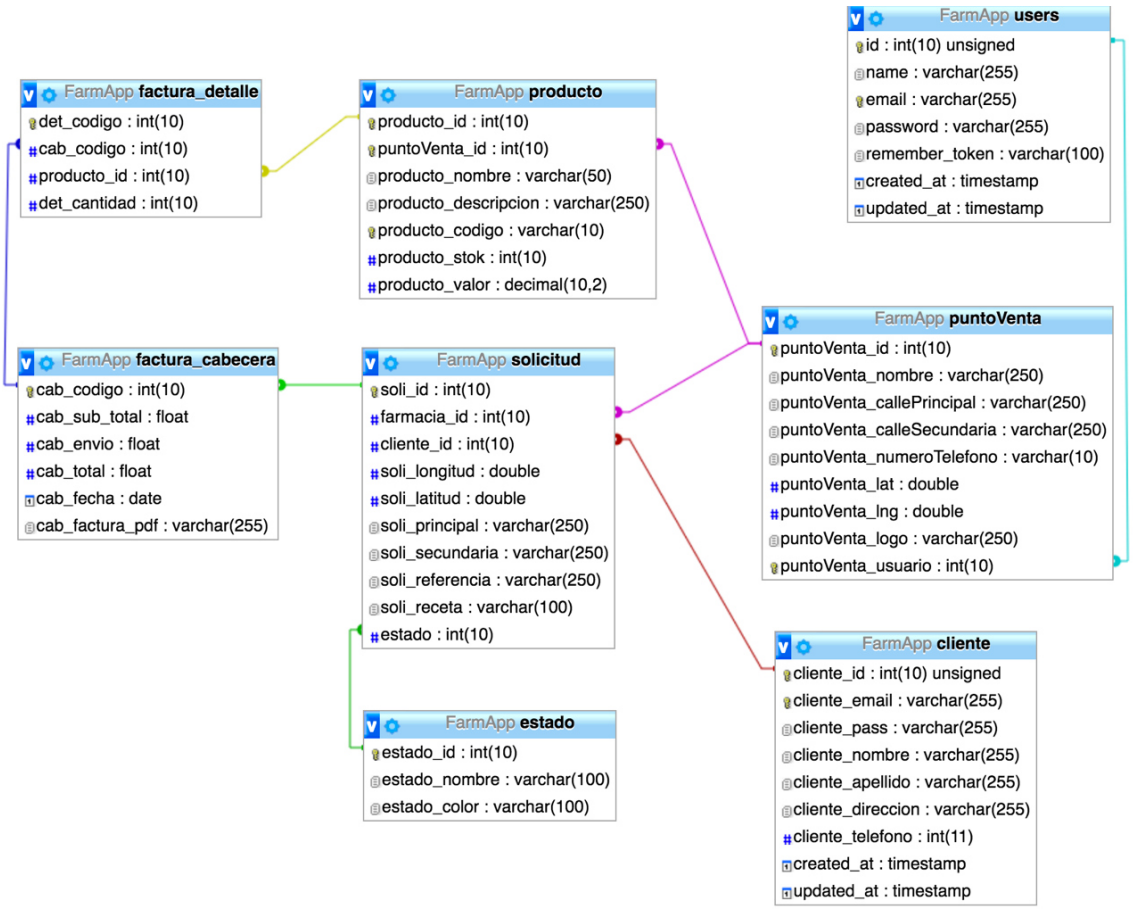


Figura 31. Modelo de la Base de Datos

5.4.1.3.5. Repositorio de imágenes

El repositorio de imágenes se encontrará únicamente en el servidor en una carpeta pública, a ella accederán la página web y el aplicativo móvil mediante el uso de los servicios web. Cada vez que un cliente deba consultar una receta médica o factura hará peticiones de locación de la imagen específica y podrán visualizarla gracias a la URL pública de dicha imagen. En cambio, para la página web al estar en el mismo servidor consumirá funciones propias de *Laravel* que facilitará todo este proceso.

5.4.2. Desarrollo de la Pagina Web

La Página web del sistema para la gestión y ventas de medicina a través de recetas médicas se ha desarrollado en *PHP 5.6.24* con el uso del *Framework Laravel 5.2* como se había mencionado anteriormente, estas interfaces fueron creados para el uso del actor Farmacéutico, para agilizar y facilitar la gestión de ventas y despacho de pedidos.

Este desarrollo se hizo en función de cada uno de los escenarios descritos en la Figura 25 y usando la arquitectura mostrada en la Figura 23 como patrón para su programación.

Por lo tanto, a continuación mostraremos cada una de las interfaces, describiendo su funcionamiento e interacción con el usuario.

5.4.2.1. Autentificación y Registro de farmacéutico

Laravel facilita el trabajo del programador ofreciendo clases y métodos para que la implementación de control de usuarios sea sencilla y ágil. En si estas interfaces ya vienen pre-programadas lo único que el programador debe hacer es llamarlas y configurarlas en el proyecto. La configuración inicial de este fichero se la puede encontrar en la ruta *config/auth.php* y los controladores que se encargan del registro y autentificación se llaman *AuthController y PasswordController* ambos se encuentran en la sub carpeta de los controladores llamada *Auth.*

En la Figura 32 se puede ver la interfaz de registro de usuario donde pedirá al usuario ingresar datos básicos. La página al momento de guardar detectara si el formulario se llenó correctamente y si no es así mostrara un mensaje en rojo enlistando los errores cometidos.

Si el formulario se completase correctamente, los datos se guardarían en la base de datos y seria direccionado a la página de datos de la farmacia. Si ya se

tuviese credenciales creadas la primera interfaz que siempre encontrará al abrir el sitio será la mostrada en la Figura 33, En esta página simplemente se colorará el email y la contraseña el sitio validara los datos y creará variables de sesión para su utilización.

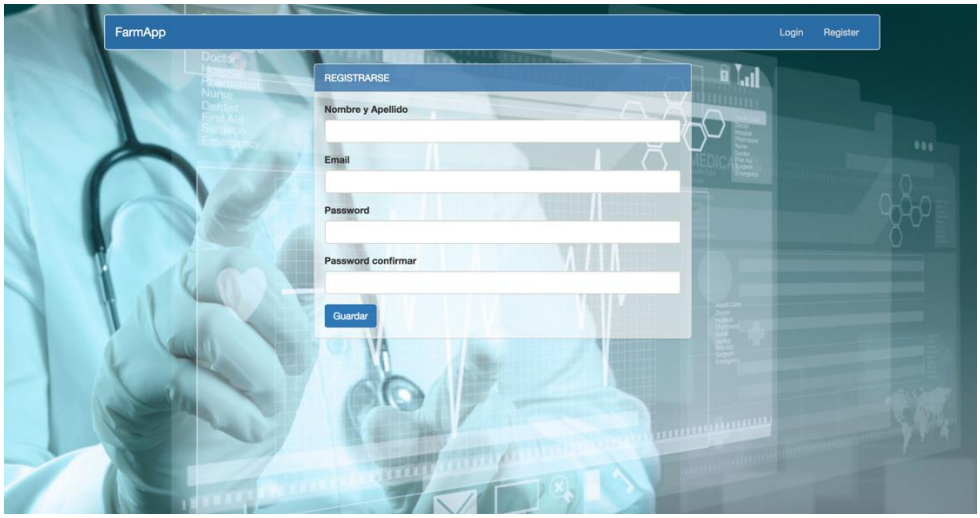


Figura 32. Interfaz de registro de usuario para la Pagina Web

Figura 33. Interfaz de Autentificación

5.4.2.2. Datos de la farmacia

Al momento de registrarse la primera interfaz direccionada es la interfaz de datos de la farmacia, esta página implementa tres partes principales como se muestra en la Figura 34, la primera partes es un formulario común para el nombre, dirección, teléfono de la farmacia, la segunda parte es para subir la foto de la farmacia o su logotipo y la última parte es para ubicar la posición de la farmacia.

Para la ubicación geo-posicional del establecimiento se utilizó una librería online de GOOGLE llamada *maps.googleapis,* esta ofrece un *PLUNING* para interfaces web que permite el geo-posicionamiento de un *MARKER*. Gracias a ello, se colocó en dos text-box la latitud y la longitud de la farmacia.

Estos datos de posicionamiento por GPS serán usados posteriormente por el aplicativo móvil por lo tanto es de gran importancia que sean configurados correctamente.

Esta página podrá ser editada cuantas veces quiera el farmacéutico solo seleccionando el botón Datos de la farmacia del menú principal.

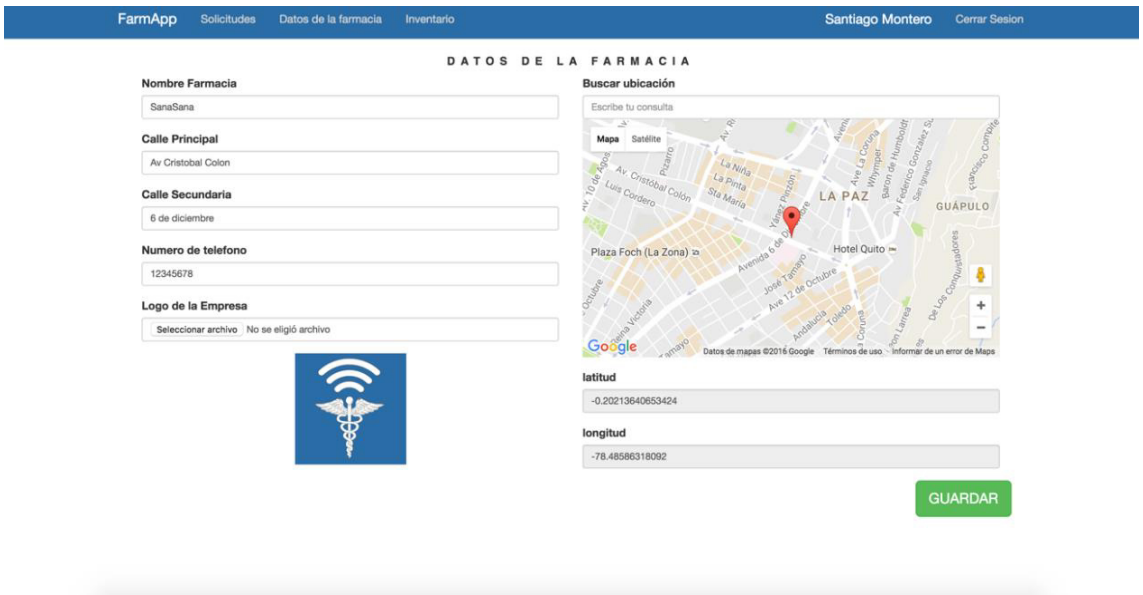


Figura 34. Interfaz de datos de la farmacia

5.4.2.3. Productos

La interfaz de gestión de productos fue desarrollada para gestionar toda la mercancía que la farmacia pueda vender por receta médica. Esta sección se divide principalmente en tres partes la primera es un listado de todos los productos con las opciones de crear, editar y borrar como se muestra en la Figura 35.

Figura 35. Interfaz del inventario de productos

Para las interfaces de crear y editar se utilizó el mismo diseño de formulario como se muestra en la Figura 36, para que se vean como pequeñas ventanas sobrestantes a la página principal se utilizó funciones por medio de JQUERY para que se puedan visualizar de forma MODAL.

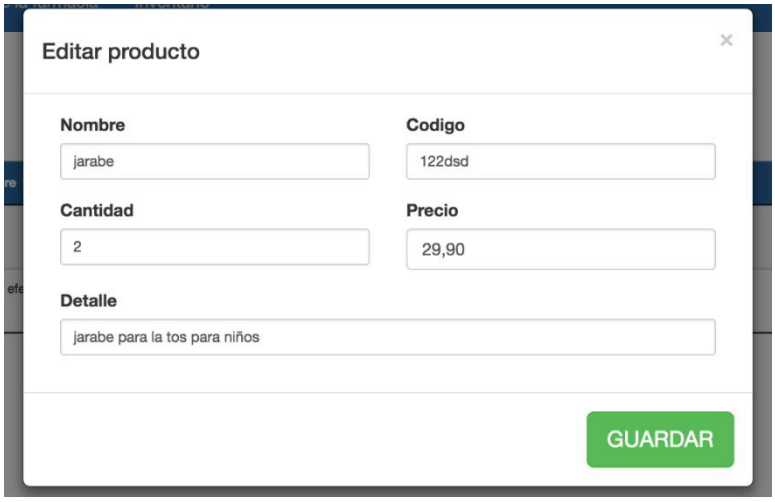


Figura 36. Ventana modal de editar productos

Estas interfaces modales de creación y edición de productos mantienen una estructura de formulario, por lo tanto si alguno de los datos es incorrecto o no es ingresado se mostrará un mensaje en color rojo para evidenciar estas falencias.

5.4.2.4. Solicitudes

La interfaz de solicitudes es la más compleja por su desarrollo en si incorpora varios *pluning* para mejor la interacción con el usuario. La primera interfaz que presenta es un listado de todas las solicitudes hechas a esa farmacia como se muestra en la Figura 37 además es posible evidenciar que cada estado es representado por un color, esto fue programado directamente en el cliente por HTML y toma el color por base de datos. Esta interfaz además incorpora por cada estado diferentes opciones que puede ser ver, enviar y rechazar solicitud.

Figura 37. Interfaz de listas de solicitudes

Para las opciones de enviar y rechazar el aplicativo interacciona por medio de peticiones Ajax al servidor y este cambia de estado a la solicitud en la base de datos además envía por medio de un correo electrónico una notificación al cliente.

Por su parte la opción ver direcciona hacia una segunda interfaz que es la que se muestra en la Figura 38, esta página contendrá los datos del cliente además de su ubicación por GPS, la foto de la receta médica y la opción de crear la factura de venta para que el cliente pueda visualizarla.

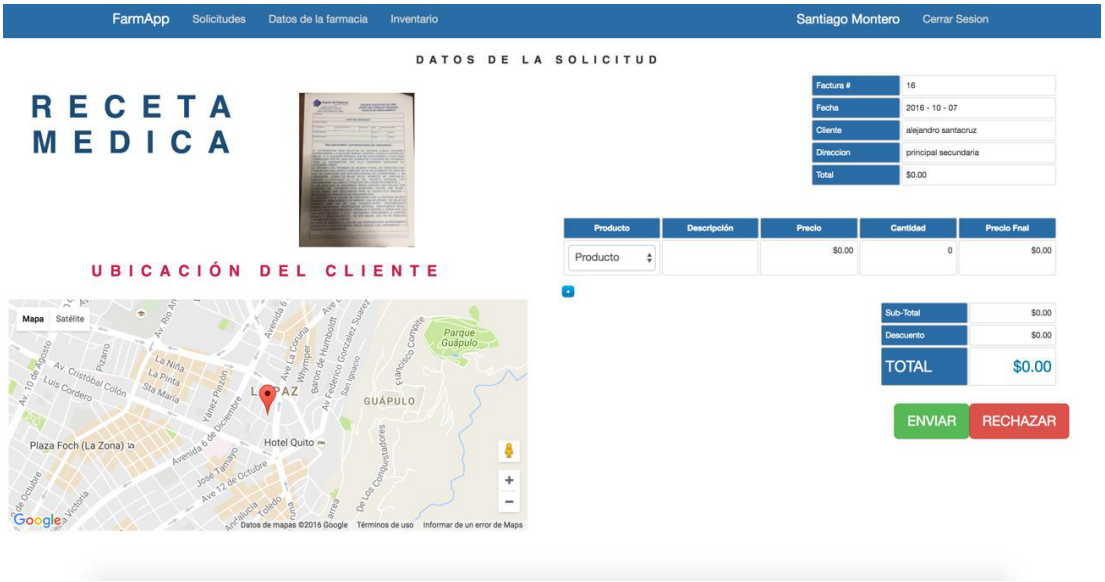


Figura 38. Interfaz de los datos de solicitud

Para la primera parte de la foto de la receta médica se incorporó una librería externa llamada *magnific-popup* que se divide una parte en *JQUERY* y otra como *CSS*. Este complemento permite crear una ventana modal de la fotografía optimizando su tamaño y calidad.

Para el mapa se utilizó el mismo componente de *GOOGLE* que se mencionó anteriormente con la única diferencia que por medio de *JQUERY* se inserta un *MARKER* estático en la localización del cliente.

La última parte de esta página es una interfaz de creación de factura dinámica en la parte de la cabecera tomará automáticamente los datos del cliente y en la segunda parte el farmacéutico ira ingresando uno por uno los productos mediante el uso de un List-box y la cantidad. Esta la interfaz calculará en automáticamente el valor total de la factura, además el farmacéutico decidirá el valor de la entrega a domicilio el cual también será sumado al total.

Una vez que el encargado de gestionar la página accione el botón enviar este cambiará de estado a la solicitud guardará la factura en la base de datos y por último enviará una notificación al cliente que el pedido está en espera de pago.

5.4.3. Desarrollo de los WebServices

Los *Servicios Web* son la clave de la comunicación entre el aplicativo móvil y el servidor estos fueron creados en PHP, todos estos servicios responden con una estructura en JSON y siempre uno de los elementos es el *CODE* que dependiendo de su número notificará si la función tuvo algún error, por ejemplo con el número 404 o terminó el proceso de forma correcta con el número 200.

Unas de las ventajas de utilizar *WebServices* es que evitan los problemas inherentes a firewalls por el mismo hecho de utilizar *HTTP,* además permite centralizar la información independientemente si las bases de datos se encuentran en el mismo servidor o no.

A continuación, se explicará el funcionamiento de los servicios web incorporados más importantes.

El servicio de autentificación es uno de los más complejos ya que este deberá utilizar funciones propias de *Laravel* para poder comparar la clave ingresada con la existente en la base de datos, que se encuentra encriptada. Este servicio devolverá mensajes de error distintos si hay problemas de conexión o si el email y la contraseña son erróneos.

El servicio que devuelve el listado de las farmacias y sus ubicaciones por GPS utiliza un método *POST* y no toma ningún dato de entrada.

Otro servicio fundamental del sistema es el que se encarga de crear nuevas solicitudes que usará el método *POST* y que se encargará de recibir una imagen como un arreglo de *STRING* luego convertirlo a un archivo.*PNG* y guardarlo en el repositorio de imágenes.

Por último, se menciona el servicio que devuelve el listado de solicitudes hechas por un cliente, este servicio POST pedirá como dato de entrada el código del

cliente y después de un proceso con la base de datos devolverá el listado de las solicitudes hechas con sus respectivos estados y facturas en caso de tenerlas.

5.4.4. Desarrollo del aplicativo móvil

La aplicación móvil se desarrolló usando Xcode 8 como herramienta y Swit 3 como lenguaje de programación, este desarrollo se basó en las interfaces descritas en la Figura 24 y utilizando la arquitectura de la Figura 23 debemos recordar que este aplicativo maneja un patrón *MVC.*

Considerando todo lo mencionado anteriormente se muestra a continuación una descripción de las diferentes interfaces que incorpora el aplicativo móvil.

5.4.4.1. Registro de Usuario

La primera vez que un cliente utilice el aplicativo deberá registrarse, la interfaz que aparecerá al cliente se muestra en la Figura 39 el usuario tendrá que llenar un formulario con datos básicos como nombre, dirección, teléfono, correo electrónico y una contraseña, este proceso consumirá un servicio *POST* de los *WebServices* donde enviara el formulario completo.

Una vez guardado se le mostrará al cliente la interfaz principal del aplicativo que corresponde al mapa con la ubicación de las diferentes farmacias.

Figura 39. Interfaz de registro aplicación móvil

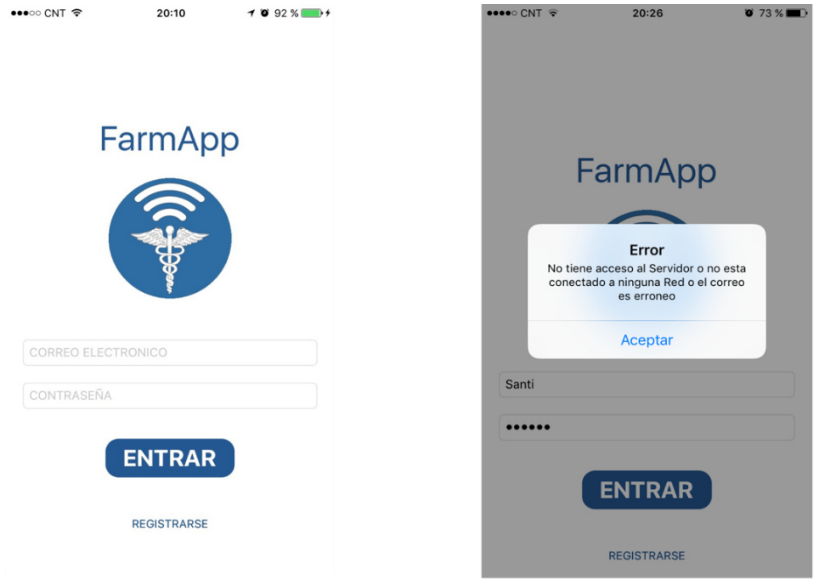
5.4.4.2. Autentificación de usuario

Este es el escenario que se presenta automáticamente cuando se abre el aplicativo y no existe una sesión abierta, podemos verlo en la Figura 40 ítem A, este como el de registrarse maneja un servicio *POST* con el servidor.

Cuando la petición no logra comunicarse con el servidor tal vez porque no tiene red o hay algún problema de infraestructura se mostrará el mensaje que se puede ver en la Figura 40 ítem B, en caso de que la contraseña sea errónea el mensaje será el siguiente “La contraseña ingresada es incorrecta”.

En el caso que las credenciales sean correctas y el servicio esté funcionando correctamente el aplicativo mostrara la interfaz principal del app y guardara como datos de *USERDEFAULT* el nombre, el código y el email del cliente.

Los *USERDEFAULT* son variables de sesión del aplicativo se guardan en la memoria del dispositivo móvil y mejoran la interacción con el cliente ya que este no deberá volver a autentificarse cada vez que abra el aplicativo.



a b

Figura 40. Escenario de autentificación

a) Pantalla de autentificación.

b) Pantalla de error de conexión al servidor

5.4.4.3. Pantalla Principal (pantalla del mapa de las farmacias)

Esta pantalla es considerada la principal ya que es la interfaz que permite acceder a todas las opciones de la aplicación, podemos verla en la Figura 41 se denotan cuatro cinco puntos principales.

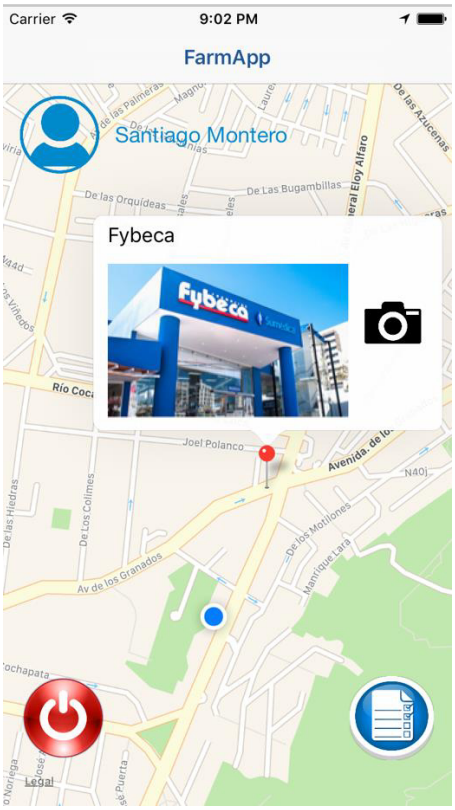


Figura 41. Interfaz del mapa de las farmacias

El primer objeto que se observa es el nombre del cliente al hacer clic en él direccionará a una interfaz que permitirá cambiar el email y contraseña.

El segundo objeto en esta interfaz está situado en el fondo y es el mapa de la ciudad para ello se usó un *pluning* proprio de *SWIFT* tomara todos los datos que maneje la aplicación de mapas de *IOS*.

Tercero *MARKERS* que indicarán la posición de las farmacias cercanas serán solicitadas por medio de un *WebServices*, al seleccionarlas mostrará foto del

establecimiento o su logo como se puede notar en la Figura 41 a lado de cada imagen existe el icono de una camera, este botón servirá para crear la solicitud.

El cuarto ítem importante en esta interfaz es un icono azul en la esquina inferior derecha de la pantalla direccionará la lista de las solicitudes hechas.

El último objeto de la pantalla es el icono de cerrar sesión en la esquina inferior izquierda de la interfaz.

Todos los botones que se presentan esta pantalla funcionan directamente en el aplicativo y no necesita comunicación con el servidor.

5.4.4.4. Pantalla de cambio de contraseña

Esta pantalla se utiliza básicamente para cambiar el email y la contraseña si es requerido como se puede ver en la **¡Error! No se encuentra el origen de la referencia.**. Como se puede notar existe un *check-box* que al seleccionarse permite cambiar la *password* pero si se deselecciona se ocultará el cambio de contraseña.

Figura 42. Interfaz de cambio de contraseña

El aplicativo validará que la antigua contraseña coincida con la situada en la base de datos por medio de *WebServices*.

5.4.4.5. Envió de receta médica

En esta interfaz implementa la opción de tomar una foto o seleccionar una del carrete del móvil, esta imagen seleccionada se podrá ver como una vista previa en la pantalla del celular como se muestra en la Figura 39.

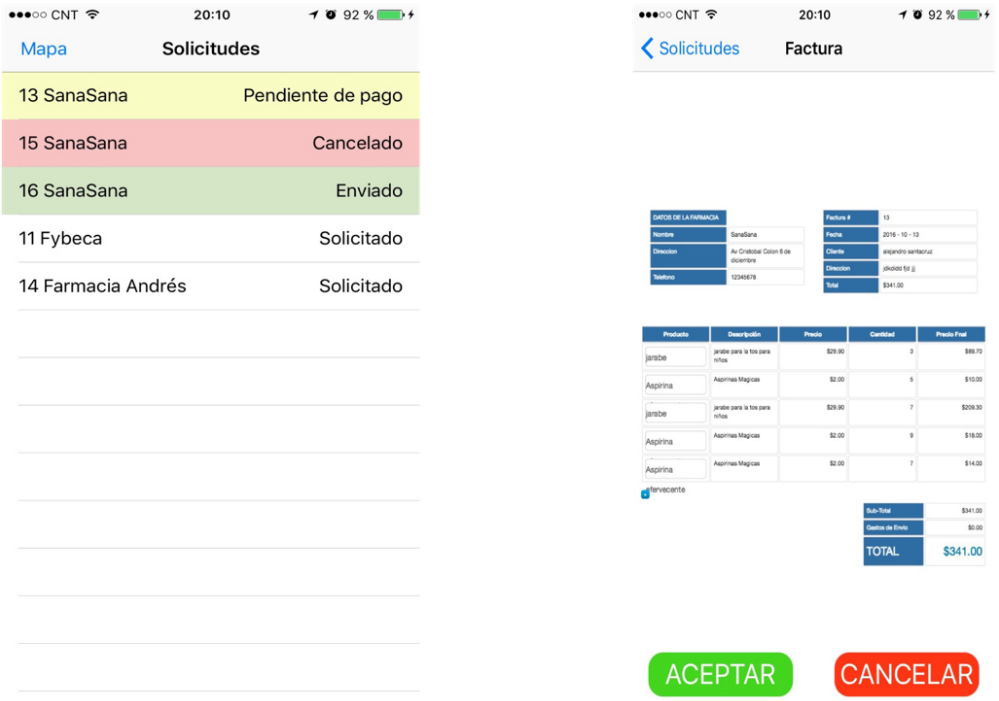
El aplicativo al ingresar en esta interfaz detectará automáticamente la calle principal en la que se encuentra y la escribirá en el primer *Text-Box*, en caso de estar situado en alguna locación sin calles cercanas el contenido de este será nulo. Además, la interfaz solicitará el ingreso de datos adicionales como calle secundaria y referencia para que la localización por parte del repartidor sea más eficiente.

Figura 43. Interfaz de envío de receta médica

Por último esta interfaz consumirá un *WebServices* con método *POST* donde enviará los datos del formulario y la imagen en formato de arreglo de datos, el servicio responderá con dos mensajes distintos si hubiese algún problema con el servicio o si la transacción finalizo correctamente. En caso de que la respuesta sea positiva el usuario será direccionado a la interfaz de lista de solicitudes.

5.4.4.6. Lista de Solicitudes

Las interfaces que incorporan esta sección son dos y se las puede ver en la Figura 40, la primera es el listado de todas las solicitudes hechas por el cliente y la segunda es la factura creada por el farmacéutico con dos opciones de interacción.



a b

Figura 44. Interfaz de solicitudes

a) Lista de solicitudes

b) Ejemplo de factura

La interfaz de listado de solicitudes incorpora un *pluning* que permite refrescar la lista con solo arrastrarla hacia abajo, ejecutando un servicio sincrónico de la información. También en la misma figura se puede notar que cada estado tiene un color único esto facilitara al cliente ubicar la solicitud deseada.

Por su parte, la interfaz de la factura mostrará la imagen de la factura y la opción de aceptar el pago del pedido que será en efectivo al momento de la entra, o rechazar el mismo. Al seleccionar cualquiera de las dos opciones por medio de

un *WebServices* se cambiará el estado de la solicitud y se enviará un email al *farmacéutico* notificando este cambio.

5.5. Infraestructura para el despliegue de las aplicaciones web y móviles

Después de identificar los actores y la arquitectura del sistema para la gestión y ventas de medicina a través de recetas médicas es necesario especificar las características de la infraestructura tecnológica del servidor que albergará la página web y los *WebServices* en el ambiente de producción.

Cuando se busca un servidor se debe prestar mucha atención a sus características, ya que esta cambiarán dependiendo de las necesidades que tenga el sistema. Las características principales serán el procesador que tiene que ser de última generación, la memoria RAM que será fundamental cuando el sistema este saturándose de solicitudes y el disco duro que deberá será de alto rendimiento y de velocidad constante aun cuando sean momentos de mayor exigencia.

Otra característica importante a tomar en cuenta al momento de seleccionar un servidor es el sistema operativo del servidor, pero ya que el sistema para la gestión y ventas de medicina a través de recetas médicas fue desarrollado en PHP y utiliza *XAMPP* que es una distribuidora de Apache facilita su instalación en los principales sistemas operativos que son Linux, OSX, y Windows.

Para un análisis más completo se analizó el ancho de banda necesario para cada transacción, y se determinaron los picos mínimos y máximos que van desde 5 kb/s hasta un tope de 1 Mb/s. Considerando que para este ambiente de prueba se calcula un máximo de veinte usuarios el servidor deberá soportar un ancho de banda máximo de 20Mb/s. Por tales razones se consideró un procesador CORE i3 de cuarta generación de 2.3GHZ, 4Gb de RAM y un disco duro de 600Gb que serán suficientes para tal ambiente.

La arquitectura de la red implementada para pruebas se la puede observar en la Figura 45 donde se denota que al ser un ambiente de test la red de trabajo será una red LAN, los dispositivos tanto móviles como laptops se conectaran por medio de una red inalámbrica al servidor.

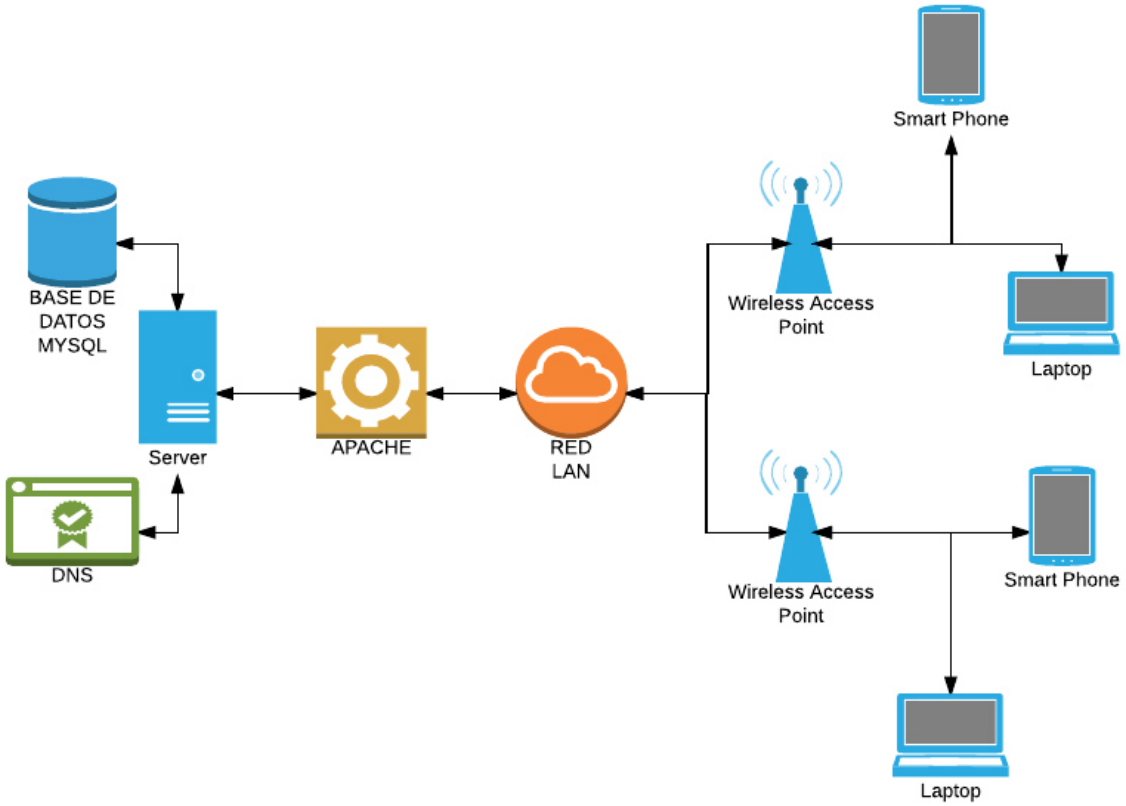


Figura 45. Arquitectura de la red del sistema para la gestión y ventas de medicina a través de recetas médicas

Por último, se deberá configurar correctamente el DNS del servidor ya que los usuarios buscaran la página web por su nombre y no por la dirección IP asignada.

5.6. Pruebas

La última etapa del proyecto de titulación fue realizar pruebas al aplicativo móvil en lo que se refiere a ancho de banda y espacio de almacenamiento. En la Figura 46 se puede observar diferentes marcadores obtenidos por medio de una

APP llamada *POSTMAN*, empezando con el literal A indica el espacio que requiere el aplicativo para ser instalado en el dispositivo que es de 32,4 MB además se puede observar el ítem *Documentos y Datos* que ocupa unos 438KB que se refiere a los datos de sesión del aplicativo que se guarda en la memoria del celular.

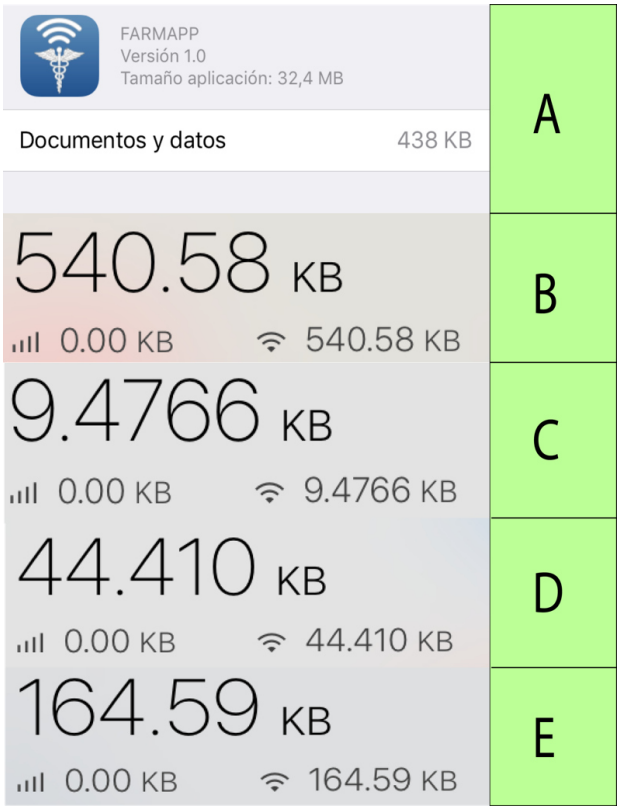


Figura 46. Ancho de banda e información de almacenamiento del aplicativo móvil

En la Figura 46 también se puede visualizar los ítems que van del literal B al E que son pruebas de consumo de ancho de banda realizadas al aplicativo al accionar diferentes funciones. El ítem B muestra la cantidad de KB necesarios para iniciar sesión al aplicativo, El literal C muestra al momento de ver la lista de solicitudes, el D es el consumo de ancho de banda al querer visualizar una Factura y por último el E muestra que son necesarios 164.59 KB para crear y enviar una nueva solicitud al farmacéutico reflejando un alto rendimiento con el mínimo consumo de datos.

Al crecimiento del aplicativo con un mayor número de farmacias el único indicador que subirá será el del Ítem B ya que al momento de inicio de sesión descargará la información de todas las farmacias en el sistema. Finalmente, teniendo en cuenta los datos obtenidos en la Figura 46 se podría asumir que con unos 50 usuarios y con un promedio de 250kb/s de ancho de banda por cada uno, una transmisión concurrente requeriría un mínimo de 12 Mb/s de velocidad por parte del servidor para brindar un servicio de alta disponibilidad y una funcionalidad adecuada.

Las pruebas referentes a funcionalidad efectuadas en el sistema y sus correcciones se detallan a continuación:

- **JQUERY página Web**

  Todas las pantallas del aplicativo Web trabajan con *JQUERY* y *CSS*, en las primeras pruebas efectuadas al sistema sobresalieron errores en el lado del cliente como se puede observar en la Figura 47. Estos errores son provocados por la falta de referencia a los distintos *JAVASCRIPT.* Por lo tanto, se solucionó incrustando las direcciones de ubicación de estas librerías en la página *LAYOUT* para evitar la redundancia de código.

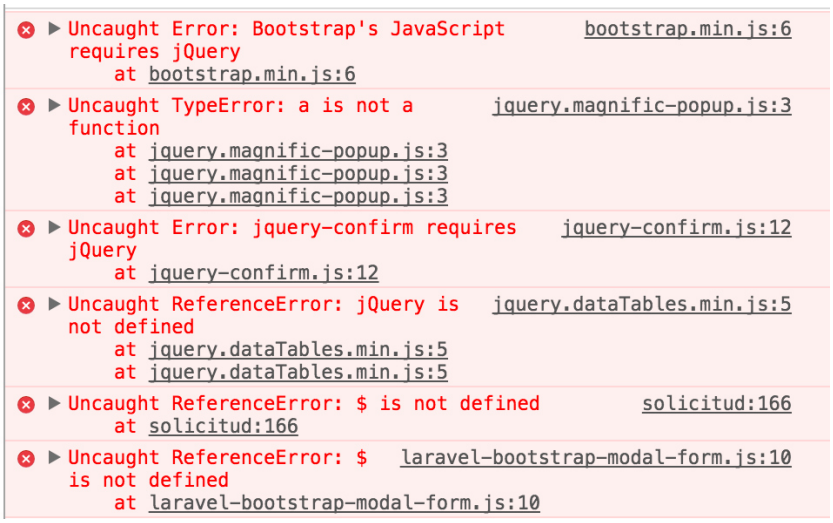


Figura 47. Errores de JQUERY

- **Envío de correo electrónico**

Una de las principales funciones que tiene el sistema es el envío de notificaciones por medio de correo electrónico. Al utilizar un *Framework* como *Laravel* lo recomendable es usar sus propias herramientas para este tipo de servicios, pero a su vez presenta inconvenientes al momento de querer ejecutar *DEBUG* en *NETBEANS* para detectar posibles fallas de programación, por tal razón se implementó el código que se puede observar en la Figura 48 y permite verificar el envío exitoso del EMAIL al cliente.

```
public function show(){

    $sent = Mail::send('email.correo', ['titulo' => 'Test','mensaje'=>'El correo de preuba fue enviado correctamente '],
            function($message)  {
              $message->to('santiago.montero92@gmail.com', 'santiago montero')
                  ->subject('Selección del mensaje de prueba');
            });
   if( ! $sent) dd(" wrong");
   dd("enviado");

}
```

Figura 48. Código de comprobación de envió de EMAIL

- **Leer *WebServices* desde *SWIFT***

La mayoría de aplicaciones móviles hoy en día trabajan con *WEBAPI REST FULL* para la comunicación con el servidor, esto presentó un reto en las pruebas de funcionalidad con los servicios generados con *Laravel*, ya que como se detalló anteriormente la herramienta utilizada no permite ejecutar *DEBUG* de forma nativa. Por tal razón, después de una investigación se implementó el código que se muestra en la Figura 49 que muestra que uno de los parámetros a enviarse es un objeto llamado "XDEBUG_SESSION_START" que obliga a *NETBEANS* a entrar al servicio en cuestión. De esta forma, fue más fácil comprobar errores en el código de los *WebServices* programados en *PHP.*

```
let parameters = [
    "cliente_email": correo.text!,
    "XDEBUG_SESSION_START" : "netbeans-xdebug"

]


var url = self.preferencias.string(forKey: "url")! + "/Webservice/public/clientes/recuperar"

Alamofire.request( url, method: .post , parameters: parameters)
    .responseJSON  {  response in
```

Figura 49. Código para hacer DEBUG a un WebService

- **Redimensionamiento de pantallas App Móvil**

Una aplicación móvil tiene que ser adaptable a cualquier tipo de pantalla en el mercado. Esto fue una complicación en las pruebas de funcionalidad realizadas al App en dispositivos con pantallas reducidas como es el IPHONE SE como se puede observar en la Figura 50 donde el objeto *UIImageView* no se ajusta totalmente al dispositivo más pequeño.

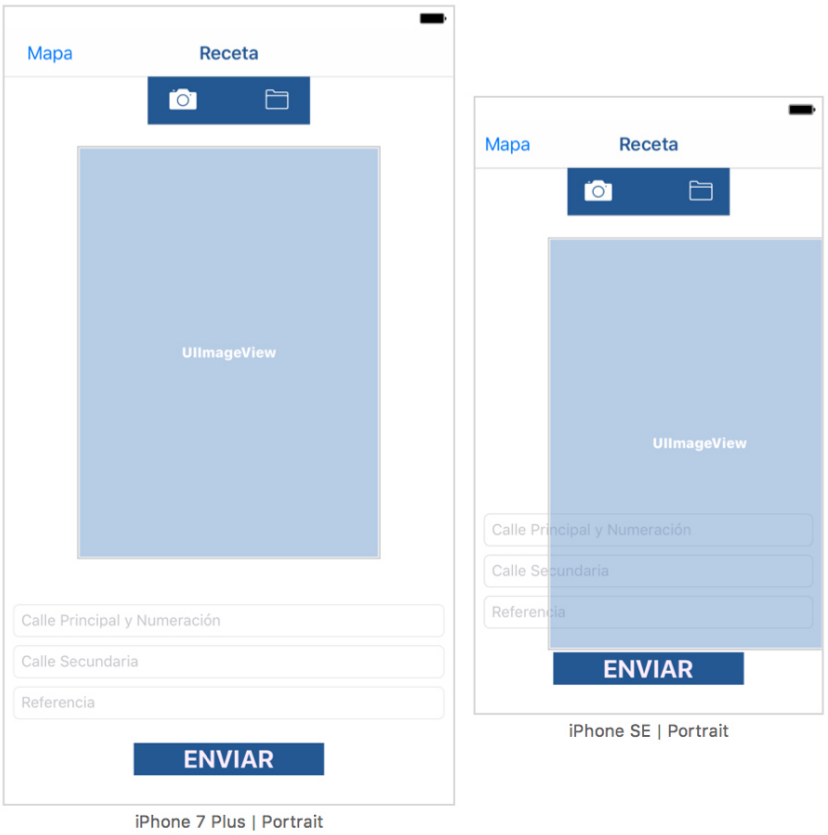


Figura 50. Pantalla del IPhone 7 Plus y del iPhone SE

Pero esto se pudo solucionar gracias a *XCode que* permite crear *constraints* que son reglas de tamaños y distancias para objetos en las interfaces. Como se puede observar en la Figura 51 se utilizaron reglas para dar prioridad a las distancias con los márgenes para que el objeto *UIImageView* se adapte de mejor manera a las diferentes dimensiones de pantalla.

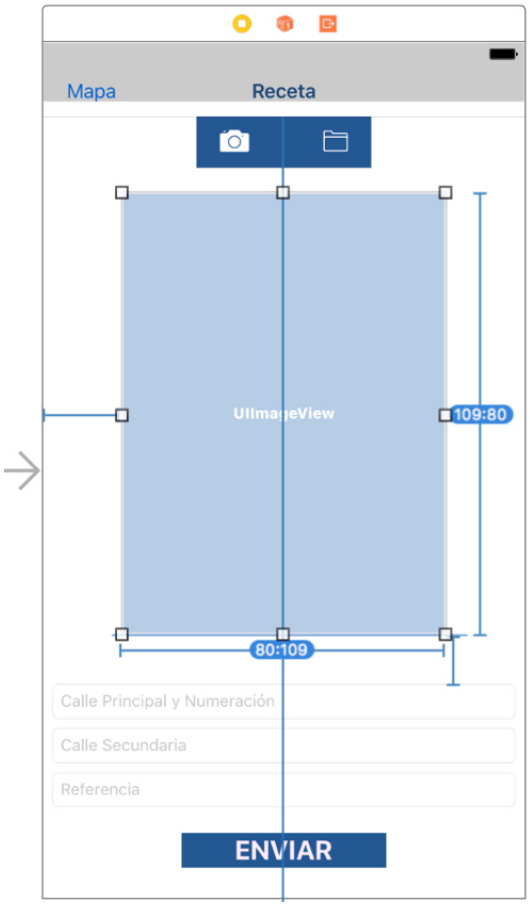


Figura 51. Constraints para interfaces Móviles

- **Respuestas *JSON***

La última prueba de funcionalidad realizada al sistema fue la eficiencia al momento de leer las respuestas *JSON* de los *WebServices* mismas que incluían fallas en las consultas con la Base de Datos provocaban errores graves al momento que *SWIFT* intentaba transformar la respuesta *JSON*

en un objeto. Por tal razón, se decidió implementar una respuesta compuesta por un código *HTTP* y el objeto resultante o un mensaje del error como se puede observar en la Figura 52.

```
// Si no existe ese fabricante devolvemos un error.
if (!$cliente)
{
    // Se devuelve un array errors con los errores encontrados y cabecera HTTP 404.
    // En code podríamos indicar un código de error personalizado de nuestra aplicación si lo deseamos.
    return response()->json(['errors'=>array(['code'=>404,'message'=>'No se encuentra un cliente con ese código.'])],404);
}

return response()->json(['status'=>'ok','data'=>$cliente],200);
```

Figura 52. Código de respuestas JSON para WebServices

6. CONCLUSIONES Y RECOMENDACIONES

6.1. Conclusiones

Las aplicaciones móviles y teléfonos inteligentes se han difundido en gran medida en los últimos años, permitiendo un crecimiento tecnológico a diferentes escalas, sin embargo el principal atractivo de los Smartphone son sus tiendas de aplicaciones, que a lo largo de los años ha ido creciendo exponencialmente, brindando la posibilidad de que cualquier persona pueda publicar aplicativos. Por tal razón en el presente proyecto de titulación se decidió implementar un software que brinde un servicio al público en general generando un sistema para la gestión y ventas de medicina a través de recetas médicas.

Al crear una nueva aplicación orientada a ser un servicio suplementario al negocio farmacéutico se consideró un análisis que determinó los requerimientos de los clientes y que los sistemas existentes actualmente son escasos y carecen de funciones de multiempresa. Por tal razón el sistema para la gestión y ventas de medicina a través de recetas médicas es totalmente viable en el mercado nacional.

La utilización del patrón MVC en el sistema facilito la programación, permitiendo separar el código en tres módulos y da la posibilidad de modificación singular de código en los mismos sin afectar los restantes. Además una arquitectura como esta facilita el escalamiento del aplicativo de una manera sencilla y rápida.

Al incorporar programación con USERDEFAULT, variables de sesión que se guardan en la memoria del dispositivo, mejora la satisfacción del usuario al utilizar esta aplicación a diario, además de aumentar la eficiencia de los recursos brindados por el teléfono inteligente al máximo y disminuir los procesos repetitivos que realiza el cliente.

El utilizar JSON como formato para el intercambio de la información, disminuye considerablemente la cantidad de ancho de banda necesario para la transacción

de operaciones, además que la interpretación de datos en este formato es sumamente sencilla para cualquier lenguaje de programación. Por tal razón es considerada como primera opción para la mayoría de programadores actualmente al momento de elegir un formato de intercambio de datos.

6.2. Recomendaciones

Al momento de querer trabajar con SWIFT en las interfaces del dispositivo móvil es importante tomar en cuenta las dimensiones de los teléfonos actuales en el mercado y configurar correctamente los Constrain para que las diferentes pantallas creadas se adapten sin ningún problema a cualquier dispositivo.

Cuando se trabaja con WebServices es importante la infraestructura del servidor ya que deberá ser de alta disponibilidad y soportar encolamiento de peticiones transaccionales con bastante frecuencia, además que por motivos de seguridad es recomendable utilizar certificado SSL y configurar el servidor para el protocolo HTTPS.

Al momento de elegir en que dispositivo empezar el desarrollar del sistema si en Android o en IOS es recomendable tomar en cuenta diferentes aspectos como son a que publico está dirigido, la complejidad de las interfaces a desarrollar y el presupuesto económico para el desarrollo del prototipo.

Hoy en día es importante al momento de desarrollar aplicaciones web hacerlas adaptables a cualquier dispositivo que se use, en otras palabras que sin importar las dimensiones de la pantalla las páginas web podrán ser usadas con facilidad en todos los dispositivos.

Incorporar medidas de seguridad en los WebServices como TOKEN y cifrado en la comunicación para garantizar mayor protección contra Sniffer. Además, se sugiere utilizar cifrado SSL y una configuración de servidor para HTTPS que unido a un correcto manejo de DNS garantizará la seguridad de la comunicación.

Para ofrecer un servicio más completo a los usuarios del sistema es necesario la implementación de funciones contable para el farmacéutico que se comunique automáticamente con el sistema para la gestión y ventas de medicina atreves de recetas médicas para ofrecer en un futuro el servicio de facturación electrónica para el cliente final.

En el afán de mejorar el rendimiento del aplicativo y ofrecer una cobertura más amplia a nivel nacional se procederá a filtrar de manera más eficaz las consultas de búsqueda de farmacias cercanas para que el cliente pueda acceder únicamente a las que ofrecen servicio a domicilio hasta su hogar.

REFERENCIAS

Alvarez, M. A. (2014). *Qué es MVC*. Recuperado el 3 de noviembre de 2016, de Desarrollo Web: http://www.desarrolloweb.com/articulos/que-es-mvc.html

Álvarez, R. (015). *Un crecimiento imparable de smartphones en México reporta 52.6 millones de dispositivos en 2014*. Obtenido de Xataka México: https://www.xataka.com.mx/celulares-y-smartphones/un-crecimiento-imparable-de-smartphones-en-mexico-reporta-52-6-millones-de-dispositivos-en-2014

Ericsson. (2015). *Ericsson Mobility Report*. Recuperado el 28 de octubre de 2016, de http://www.ericsson.com/res/docs/2015/mobility-report/ericsson-mobility-report-nov-2015.pdf

Fowler, M., & Sccott, K. (1999). *UML Gota a Gota.* Wesly: Pearson Addisson.

Ibarra Avalos, S. A. (2001). *Definición de Cliente Servidor*. Recuperado el 1 de noviembre de 2016, de Universidad de Colima / Facultad de Telemática: http://docente.ucol.mx/sadanary/public_html/bd/cs.htm#_Definici%F3n_de_Cliente

Jacobson, I., Booch, G., & Rumbaugh, J. (2000). *El Proceso Unificado de Desarrollo de Software.* Wesley: Pearson Addisson.

JSON. (s.f.). *Introduccion a JSON*. Recuperado el 3 de novimbre de 2016, de http://www.json.org/json-es.html

Net Humans. (s.f.). *Soluciones*. Recuperado el 7 de noviembre de 2016, de http://www.nethumans.com/solutions/development/Database.aspx

Otwell, T. (2016). *Eloquent: Getting Started*. Recuperado el 5 de noviembre de 2016, de Laravel: https://laravel.com/docs/5.3/eloquent

Universidad Politécnica de Valencia. (2002). *Departamento de Informática de Sistemas y Computadores*. Recuperado el 1 de noviembre de 2016, de http://www.disca.upv.es/enheror/pdf/ActaUML.PDF.

ANEXOS

Anexo 1. Manual de Usuario de FarmApp

1) Registrarse

Ingrese sus datos para poder crear un usuario y poder acceder al aplicativo de FarmApp.

Al momento de completar el registro será redirigido a la pantalla principal del aplicativo.

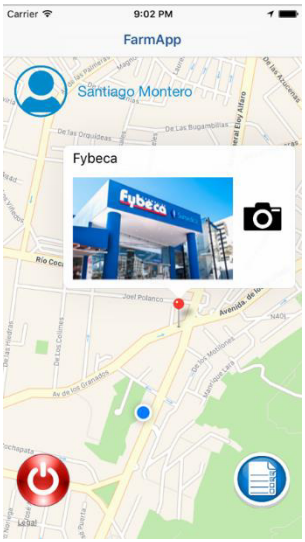
2) Autentificarse

La primera interfaz con la que se encontrará será la de autentificación donde ingresará sus datos de usuario después procederá da darle clic al botón entrar.

En el caso que los datos ingresados sean herrones o haya problemas de conexión se desplegara el mensaje informándole de la incidencia.

Si no tiene datos de usuario proceda a darle clic al botón de registro.

3) Pantalla Principal



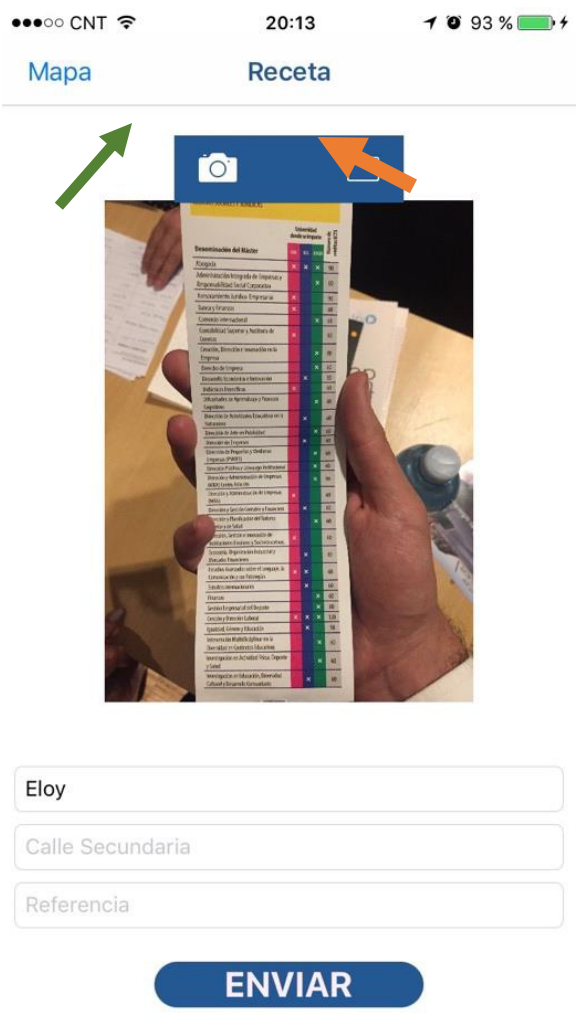
En la pantalla principal tiene la posibilidad de acceder a diferentes opciones.

A. En el botón donde se muestra su nombre podrá cambiar su contraseña y su email.
B. El botón rojo es de cierre de sesión.
C. El botón azul en la parte inferior a la derecha del aplicativo desplegara la lista de las solicitudes hechas.

D. Todos los MARKER que se observan en el mapa son farmacias que utiliza el sistema de FarmApp.

Cada vez que haga clic en una farmacia se mostrará la foto del establecimiento y le permitirá tomar una foto de la receta médica.

4) Nueva Solicitud



En la interfaz de nueva solicitud usted podrá decidir si tomar una nueva foto o seleccionar una de su carrete.

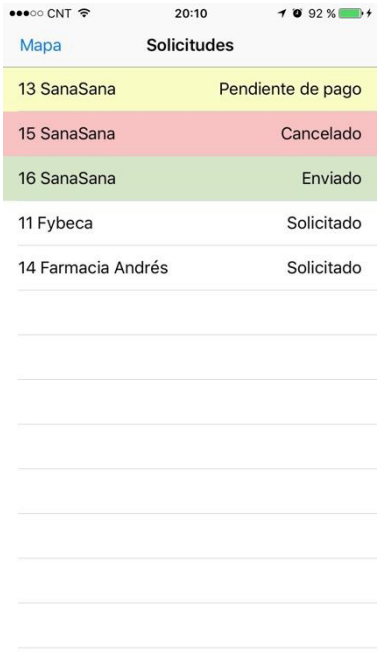
La calle principal se escribirá automáticamente gracias al GPS, pero adicionalmente le solicitará una calle secundaria y una referencia.

Una vez ingresado todos los datos requeridos se procederá a dar clic al botón enviar.

Aparecerá un mensaje de confirmación de envió y si el proceso termina de forma correcta se mostrará el mensaje correspondiente en caso contrario se visualizará cual es la incidencia.

Una vez finalizado este proceso se le re direccionará a la página de listas de solicitudes.

5) Lista de solicitudes



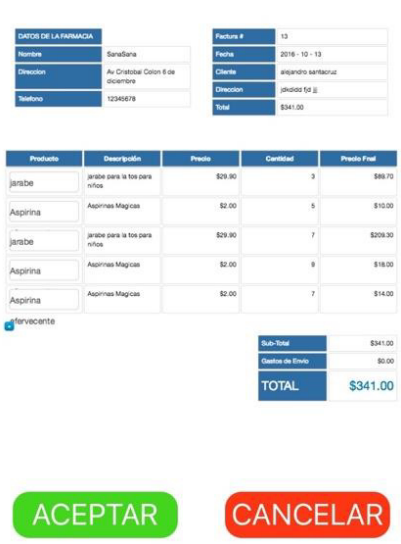
En esta interfaz usted podrá observar la lista de las solicitudes hechas,

Cada uno de los colores resalta el estado en el que se encuentra la solicitud.

Las solicitudes que se encuentren en el estado Solicitado y Cancelado serán estáticas es decir no podrá interactuar con ellos.

Al hacer clic a cualquier otro estado se le redirigirá a una página con la factura de la solicitud y opciones del pedido.

6) Factura de la Solicitud



En esta interfaz usted pondrá ver el detalle de la factura de la solicitud, con el valor individual de cada artículo costos de envió y datos de la farmacia y sus datos como cliente.

Adicionalmente se le solicitara aceptar el valor del pedido para proceder al despacho.

En caso de cancelación de pedido no podrá reversar dicha petición.

7) Notificaciones Email

Cada vez que su solicitud cambie de estado por acciones hechas por el farmacéutico usted recibirá una notificación de tipo email detallando en qué estado se encuentra su solicitud.

Cuando su solicitud cambie a despachado en el email recibido se anexará la factura del pedido.

8) Cambio de Credenciales

En esta interfaz usted tendrá la potestad de cambiar su email, además si activa el check de cambio de contraseña se actualizará también su clave.

Al momento de dar clic en guardar si todos los datos ingresados son correctos se le notificará el éxito de la transacción en caso contrario se mostrará cual es el error.