UNIVERSIDAD TÉCNICA DE AMBATO

FACULTAD DE INGENIERÍA EN SISTEMAS, ELECTRÓNICA E INDUSTRIAL

CARRERA DE INGENIERÍA EN SISTEMAS COMPUTACIONALES E INFORMÁTICOS

Tema:

APLICACIÓN PARA LA GESTIÓN DE ÓRDENES EN RESTAURANTES DE LA CIUDAD DE AMBATO UTILIZANDO TECNOLOGÍA MÓVIL

Trabajo de Graduación Modalidad: Proyecto de Investigación, presentado previo la obtención del título de Ingeniero en Sistemas Computacionales e Informáticos

LÍNEA DE INVESTIGACIÓN:

Orientación a objetos

AUTOR: Juan José Mesias Valencia
TUTOR: Ing. PhD. Félix Oscar Fernández Peña, MSc.

Ambato - Ecuador
Agosto, 2020

APROBACIÓN DEL TUTOR

En mi calidad de Tutor del Trabajo de Titulación con el tema: "APLICACIÓN PARA LA GESTIÓN DE ÓRDENES EN RESTAURANTES DE LA CIUDAD DE AMBATO UTILIZANDO TECNOLOGÍA MÓVIL", desarrollado bajo la modalidad Proyecto de Investigación por el señor Juan José Mesias Valencia, estudiante de la Carrera de Ingeniería en Sistemas Computacionales e Informáticos, de la Facultad de Ingeniería en Sistemas, Electrónica e Industrial de la Universidad Técnica de Ambato, me permito indicar que el estudiante ha sido tutorado durante todo el desarrollo del trabajo hasta su conclusión, de acuerdo a lo dispuesto en el Artículo 15 del Reglamento para obtener el Título de Tercer Nivel, de Grado de la Universidad Técnica de Ambato, y el numeral 7.4 del respectivo instructivo.

Ambato, agosto de 2020

FELIX OSCAR FERNANDEZ PENA

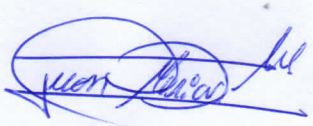
Digitally signed by FELIX OSCAR FERNANDEZ PENA
DN: cn=FELIX OSCAR FERNANDEZ PENA c=EC l=QUITO o=BANCO CENTRAL DEL ECUADOR ou=ENTIDAD DE CERTIFICACION DE INFORMACION-ECIBCE
Reason: I am the author of this document
Location:
Date: 2020-08-07 11:34-05:00

Ing. PhD. Félix Oscar Fernández Peña, MSc.
EL TUTOR

AUTORÍA

El presente trabajo de investigación titulado: "APLICACIÓN PARA LA GESTIÓN DE ÓRDENES EN RESTAURANTES DE LA CIUDAD DE AMBATO UTILIZANDO TECNOLOGÍA MÓVIL" es absolutamente original, auténtico y personal. En tal virtud, el contenido, a efectos legales y académicos que se desprenden del mismo, son de exclusiva responsabilidad del autor.

Ambato, agosto de 2020



Juan José Mesias Valencia
CC: 1804569364

APROBACIÓN DEL TRIBUNAL DE GRADO

En calidad de par calificador del Informe Final del Trabajo de Titulación presentado por el señor Juan José Mesias Valencia, estudiante de la Carrera de Ingeniería en Sistemas Computacionales e Informáticos, de la Facultad de Ingeniería en Sistemas, Electrónica e Industrial, bajo la Modalidad Proyecto de Investigación, titulado "APLICACIÓN PARA LA GESTIÓN DE ÓRDENES EN RESTAURANTES DE LA CIUDAD DE AMBATO UTILIZANDO TECNOLOGÍA MÓVIL", nos permitimos informar que el trabajo ha sido revisado y calificado de acuerdo al Artículo 17 del Reglamento para obtener el Título de Tercer Nivel, de Grado de la Universidad Técnica de Ambato, y al numeral 7.6 del respectivo instructivo. Para cuya constancia suscribimos, conjuntamente con la señora Presidenta del Tribunal.

Ambato, agosto 2020

Firmado electrónicamente por:
ELSA PILAR URRUTIA

Ing. Elsa Pilar Urrutia Urrutia
PRESIDENTA DEL TRIBUNAL

Firmado electrónicamente por:
DAVID OMAR GUEVARA AULESTIA

Ing. David Guevara
PROFESOR CALIFICADOR

Firmado electrónicamente por:
FRANKLIN OSWALDO MAYORGA MAYORGA

Ing. Franklin Mayorga
PROFESOR CALIFICADOR

DERECHOS DE AUTOR

Autorizo a la Universidad Técnica de Ambato, para que haga uso de este Trabajo de Titulación como un documento disponible para la lectura, consulta y procesos de investigación.
Cedo los derechos de mi Trabajo de Titulación en favor de la Universidad Técnica de Ambato, con fines de difusión pública. Además, autorizo su reproducción total o parcial dentro de las regulaciones de la institución.

Ambato, agosto de 2020

Juan José Mesias Valencia
CC: 1804569364

Dedicatoria

Se la dedico a mi madre Angelita y a mi padre Julio por tener el valor y la fortaleza de haber forjado a sus hijos con el ejemplo del esfuerzo y sacrificio.

Juan José Mesias Valencia

Agradecimiento

Agradezco a la Universidad Técnica de Ambato por cumplir con su misión de formar profesionales con pensamiento critico y competentes para la sociedad.

A mis compañeros.

Gracias totales.

Juan José Mesias Valencia

Índice

Índice de figuras

Índice de tablas

Resumen Ejecutivo

Los restaurantes en la actualidad se ven en la necesidad de incorporar nuevas tecnologías a su modelo de negocio de esta manera optan por paginas web, redes sociales y canales de comunicación como WhatsApp entre otros para ofertar y dar a conocer sus productos y servicios.

Gracias a las nuevas tecnologías los restaurantes tienen la posibilidad de optar por una solución a la medida que permita la gestión interna del proceso o modelo de negocio que manejan para prestar el servicio, como se puede evidenciar varios de las franquicias de comida rápida ofertan sus servicios de manera digital.

A nivel nacional la incorporación de nuevas tecnologías en el sector gastronómico se ha incrementado gradualmente ofreciendo el servicio a domicilio como tal, un factor no tomando en cuenta con anterioridad dentro del sector es la gestión de las órdenes de una manera digital cuando el cliente se encuentra dentro del restaurante, así el cliente reduce sus tiempos en realizar la orden y de esperar mientras la misma es entregada.

El presente proyecto fue enfocado en la gestión de órdenes cuando el cliente está sentado en una mesa dentro del restaurante, el mismo fue desarrollado con Reactjs que permite un diseño adaptativo por componentes y así garantizar la disponibilidad del proyecto en diversos dispositivos.

La metodología utilizada para el desarrollo del proyecto fue la de prototipo evolutivo, el cual se basa en presentar un prototipo inicial, valorarlo con el usuario y de esta manera llegar a una versión final.

API-REST, React, ReactJS, REDUX, Laravel, WebSockets.

Abstract

Restaurants are currently in need of incorporating new technologies into their business model, so they opt for web pages, social networks and communication channels such as WhatsApp, among others, to offer and publicize their products and services.

Thanks to new technologies, restaurants have the possibility of opting for a tailored solution that allows the internal management of the process or business model they manage to provide the service, as evidenced by several of the fast food franchises that offer their services. digitally.

At the national level, the incorporation of new technologies in the gastronomic sector has gradually increased offering home delivery as such, a factor not previously taken into account within the sector is the management of orders in a digital way when the client is inside the restaurant, thus the client reduces his time in placing the order and waiting while it is delivered.

This project was focused on order management when the client is sitting at a table inside the restaurant, it was developed with Reactjs that allows an adaptive design by components and thus guarantee the availability of the project in various devices.

The methodology used for the development of the project was that of an evolutionary prototype, which is based on presenting an initial prototype, evaluating it with the user and thus reaching a final version.

API-REST, React, ReactJS, REDUX, Laravel, WebSockets.

INTRODUCCIÓN

El presente proyecto de investigación denominado "APLICACIÓN PARA LA GESTIÓN DE ÓRDENES EN RESTAURANTES DE LA CIUDAD DE AMBATO UTILIZANDO TECNOLOGÍA MÓVIL" se encuentra dividido en los siguientes capitulos:

CAPÍTULO I. "MARCO TEÓRICO", se plantea un problema a investigar se establece la justificación y los objetivos que guiaran el desarrollo del proyecto.

CAPÍTULO II. "METODOLOGÍA", se reúne todas las técnicas y herramientas necesarias para el desarrollo del proyecto, además se define las etapas de que cubrirán el desarrollo del mismo.

CAPÍTULO III. "RESULTADOS Y DISCUSIÓN", se describe de manera precisa el desarrollo del proyecto sus características y funcionamiento haciendo énfasis en las partes fundamentales del desarrollo.

CAPÍTULO IV. "CONCLUSIONES Y RECOMENDACIONES", en esta sección se señalan las conclusiones y recomendaciones que se han encontrado a lo largo del desarrollo del proyecto.

CAPÍTULO I

Marco Teórico

1.1 Tema de investigación

Aplicación para la gestión de órdenes en restaurantes de la ciudad de Ambato utilizando tecnología móvil

1.2 Antecedentes investigativos

1.2.1 Contextualización del problema

En Ecuador, los emprendimientos dedicados al sector gastronómico se han incrementado de manera considerable en los últimos años con el fin de convertir al Ecuador en una potencia gastronómica [1] generando un alto nivel competitivo dentro de esta rama, donde cada local comercial, a más de buscar destacar por su oferta culinaria, se enfoca en brindar una atención de calidad a sus clientes. Sin embargo, entre los principales inconvenientes con los que se enfrentan quienes se dedican a esta actividad, es la gestión ineficiente de órdenes, que, al ser tomadas de manera tradicional, usando papel y lápiz, llegan a presentar situaciones como: confusión de órdenes y retraso en los pedidos, provocando que el grado de satisfacción del cliente decaiga de manera considerable, generando una reputación negativa para el local comercial. Adicionalmente, el menú de una gran cantidad de estos establecimientos es definido de manera empírica y a criterio de los propietarios, dejando de lado los gustos y preferencias de los clientes, lo que a largo plazo acarrea un declive en ventas.

Las grandes franquicias de comida han hecho uso de las tecnologías de la información para tener una gestión eficiente de su proceso de ventas, teniendo un impacto positivo, el cual es visto de manera favorable por las personas que frecuentan estos lugares, puesto que reconocen la eficiencia al momento de realizar su compra [2]. No obstante, una cantidad reducida de restaurantes pertenecientes a la categoría de pequeñas empresas incorporan herramientas tecnológicas para su gestión, reconociendo que implementar estas soluciones tecnológicas constituyen una gran ventaja competitiva, ya que les permite brindar servicios personalizados, llegar directamente al cliente y diferenciarse de la competencia [3], pero que, al pensar que el costo de implementación de estas soluciones es elevado, no se las llega a considerar como una alternativa factible. En la provincia de Tungurahua, las gestiones para mejorar el servicio de atención al cliente por parte de la Asociación de Restaurantes de Tungurahua (ASOREST) se está enfocando bajo el principio de "Mantener buenas prácticas de atención y calidad en el servicio" [4], reconociendo que las herramientas tecnológicas como los sitios web, redes sociales

y marketing digital han provocado que los restaurantes tengan la oportunidad de promocionarse y darse a conocer a un mayor número de personas en base a una estrategia comunicativa [5], lo que los ha llevado a poner en consideración el uso de soluciones apoyadas en las tecnologías de la información en aspectos referentes a la gestión y administración para sus locales comerciales para evaluar su impacto y eficiencia frente al sistema de gestión cotidiano. Los restaurantes ubicados en la ciudad de Ambato se apoyan principalmente en el modelo tradicional de atención al cliente, lo que los ha llevado a presentar con mayor frecuencia problemas para llevar a cabo su correcta gestión en el proceso de ventas, especialmente en las horas de mayor afluencia de clientes. A pesar de esta situación, los establecimientos no apuestan por alternativas tecnológicas para mejorar su situación, aludiendo a la creencia de que el consto de implementación de éstas es relativamente alto.
De acuerdo con investigaciones relacionadas al problema mencionado se puede mencionar los resultados obtenidos por Marco Antonio Muños Torrea que realizo en el proyecto denominado "Desarrollo de una aplicación móvil para la realización de reservas y toma de órdenes en el restaurante LongHorn" [2], en la Universidad de Inca Garcilaso de la Vega Perú, que hace énfasis en la importancia que tiene la aplicación de las nuevas tecnologías en el desarrollo de la gastronomía peruana y que además los restaurantes están en la posibilidad de brindar un excelente servicio utilizando herramientas inteligentes. Una solución interesante que también hay que mencionar es la que se llevó a cabo por parte de Galo Fernando Peñaherrera Morán y Brian Antonio Torres Reynoso que desarrollaron el proyecto denominado "Automatización de la gestión de órdenes de pedidos para restaurantes con servicio a la mesa y a domicilio" [6], en la ciudad de Guayaquil en la Universidad de Guayaquil. En esta investigación se menciona el notable crecimiento gastronómico en el sector de Guayaquil debido a la gran demanda que ejercen los turistas en la zona y propone la solución de una aplicación móvil para la gestión de pedidos a domicilio y a la mesa, el proyecto mencionado está conformado por dos aplicaciones en diferentes plataformas ya que no explota las tecnologías móviles actuales para desarrollar una sola aplicación para la gestión de la misma y la toma de órdenes en el restaurante. Por otro lado, esta está el proyecto de Paul Andrés Ochoa Quinteros que realizo el proyecto denominado "Aplicación interactiva para gestión de órdenes y pedidos en restaurantes" [7], realizada en la Universidad de Azuay, en el cual hace énfasis en la tecnología que tenemos a disposición para realizar una aplicación interactiva que contenga elementos multimedia para dispositivos Android, que permita a los clientes realizar las órdenes y los pedidos de manera ágil y sencilla, dicha aplicación solamente se enfoca en los usuarios Android lo que con el tiempo conllevaría un problema de escalabilidad. Cabe mencionar una de las investigaciones realizadas en la ciudad de Ambato por Diana Cristina Altamirano Andrade con el tema "Aplicación móvil con realidad aumentada como estrategia de marketing 2.0 para el menú del restaurante chimichurri moros & menestras en la ciudad de Ambato" [3], que se encuentra en el repositorios de la Universidad Técnica de Ambato, es importante destacar esta investigación ya que es un claro ejemplo de la aplicación de nuevas tecnologías en los restaurantes de la ciudad, mostrando la capacidad de enfocarse en la perspectiva del cliente al realizar sus pedidos brindándole un extra de interacción con el menú digital.

1.2.2 Fundamentación teórica

Web 2.0

La Web 2.0 describe un conjunto de tecnologías de Internet de próxima generación. Estos protocolos y herramientas facilitan la creación de aplicaciones en línea que se comportan dinámicamente, al igual que el software tradicional basado en PC. También son muy sociales, alentando a los usuarios a manipular e interactuar con el contenido de nuevas maneras. La Web 2.0 empuja la potencia informática del escritorio a Internet, lo que significa menos tiempo y dinero gastado en la administración de software para PC. Como regla general, las herramientas Web 2.0 también son menos costosas que el software tradicional, y muchas incluso son gratuitas. Debido a que están basados en la Web, todo lo que necesita para comenzar es un navegador actualizado [8].
En 1984, el cofundador de Sun Microsystems, John Gage, acuñó la frase "la red es la computadora" para describir su visión del futuro de la tecnología de la información. Esta fue una declaración audaz en ese momento, porque anticipaba un futuro en el que las redes de datos serían lo suficientemente potentes como para suplantar a los mainframes y PC de escritorio como un recurso de TI principal [8].
En general, las características clave de la Web 2.0 son:

- Se puede acceder a las aplicaciones web desde cualquier lugar.
- Las aplicaciones simples resuelven problemas específicos. El valor reside en el contenido, no en el software utilizado para mostrar el contenido.
- Los datos se pueden compartir fácilmente.
- La distribución es de abajo hacia arriba, no de arriba hacia abajo.
- Los empleados y clientes pueden acceder y utilizar herramientas por su cuenta.
- Las herramientas sociales alientan a las personas a crear, colaborar, editar, clasificar, intercambiar y promover información.
- Se alientan los efectos de red; Cuantas más personas contribuyan, mejor será el contenido.

Aplicación web

Las aplicaciones web reciben este nombre porque se ejecutan en internet. Es decir que los datos o los archivos en los que se trabaja son procesados y almacenados dentro de la web. Estas aplicaciones, por lo general, no necesitan ser instaladas en el computador [9].
El concepto de aplicaciones web está relacionado con el almacenamiento en la nube. Toda la información se guarda de forma permanente en grandes servidores de internet y envían a los dispositivos o equipos los datos que se requieren en ese momento, quedando una copia temporal dentro del equipo del usuario [9].
En cualquier momento, lugar y desde cualquier dispositivo se puede acceder a

este servicio, sólo se necesita una conexión a internet y los datos de acceso, que por lo general son el nombre de usuario y contraseña [9].
Estos grandes servidores de internet que prestan el servicio de alojamiento están ubicados alrededor de todo el mundo, así hacen que el servicio prestado no sea tan costoso o gratuito en la mayoría de los casos y extremadamente seguro [9].
Cuando se utiliza una aplicación web se está trabajando desde un computador o dispositivo móvil, pero la mayor parte del procesamiento se hace dentro de en una red de servidores. Estos servidores pueden unir todo su poder de procesamiento con el fin de tramitar solicitudes de todo el mundo, y a su vez, utilizan servidores especializados para almacenar los datos con los que se está trabajando, así como los datos de otros usuarios. Como todo esto sucede sin problema ni demora alguna, pareciera que la aplicación se está ejecutando dentro del equipo [9].
Las aplicaciones web tienen ventajas como:

- Muchas aplicaciones web son gratuitas.
- Se puede acceder a la información en cualquier lugar y momento.
- No se depende de un computador o de algún equipo específico ya que el contenido está almacenado en la web.
- Muchas de las aplicaciones web permiten que varias personas trabajen simultáneamente en ellas.
- Los documentos y archivos no se pierden ni borran a menos que el usuario así lo desee.

Aplicación Web Progresiva

Las Progressive Web Apps entregan una experiencia igual que las aplicaciones creadas para ordenadores de escritorio y móviles que se entregan directamente a través de la web. Son aplicaciones web rápidas y confiables. Y lo más importante, son aplicaciones web que funcionan en cualquier navegador. Toda experiencia web debe ser rápida, y esto es especialmente cierto para las Progressive Web Apps. Rápida se refiere al tiempo que se tarda en obtener contenido significativo en la pantalla y brindar una experiencia interactiva en menos de 5 segundos [10].
Las Progressive Web Apps pueden ejecutarse en una pestaña del navegador, pero también son instalables. Añadir a marcadores un sitio simplemente agrega un acceso directo, pero una Progressive Web App instalada se ve y se comporta como todas las demás aplicaciones instaladas. Se inicia desde el mismo lugar que se lanzan otras aplicaciones. Puedes controlar la experiencia de lanzamiento, incluida una pantalla de inicio personalizada, iconos y más. Se ejecuta como una aplicación, en una ventana de aplicación sin una barra de direcciones u otra interfaz de usuario del navegador. Y como todas las demás aplicaciones instaladas, es una aplicación de nivel superior en el gestor de tareas. Es fundamental que una PWA instalable sea rápida y confiable. Los usuarios que instalan una PWA esperan que sus aplicaciones funcionen, sin importar en qué tipo de conexión de red estén conectados. Es una expectativa de referencia que todas las aplicaciones instaladas deben cumplir [10].

Xaamp

XAMPP es el entorno más popular de desarrollo con PHP. XAMPP es una distribución de Apache completamente gratuita y fácil de instalar que contiene MariaDB, PHP y Perl. El paquete de instalación de XAMPP ha sido diseñado para ser increíblemente fácil de instalar y usar [11].
Mucha gente conoce de primera mano que no es fácil instalar un servidor de web Apache y la tarea se complica si se le añade MariaDB, PHP y Perl. El objetivo de XAMPP es crear una distribución fácil de instalar para desarrolladores que se están iniciando en el mundo de Apache. XAMPP viene configurado por defecto con todas los opciones activadas. XAMPP es gratuito tanto para usos comerciales como no comerciales. En caso de usar XAMPP comercialmente, asegúrate de que cumples con las licencias de los productos incluidos en XAMPP. Actualmente XAMPP tiene instaladores para Windows, Linux y OS X [11].
XAMPP es una compilación de software libre (similar a una distribución de Linux). Es gratuita y puede ser copiada libremente de acuerdo a la licencia GNU GPL. Únicamente la compilación de XAMPP está publicada bajo la licencia GPL. Cada uno de los componentes incluidos tiene su propia licencia, en particular MySQL. Desde el punto de vista de XAMPP como compilación, el uso comercial es gratuito [11].

MySQL

MySQL es un sistema de administración de base de datos relacional cliente/servidor originada de Escandinavia. MySQL incluye un servidor SQL, programas clientes para acceder al servidor, herramientas administrativas, y un interfaz para escribir programas. Inicialmente, MySQL comenzó mundialmente popular debido a su velocidad y simplicidad, además de agregar otras características como replicación, subconsultas, almacenamiento procedural, vistas y disparadores. MySQL es portable y corre en sistemas operativos comerciales y todos los sistemas puestos para servidores de empresas. Lo mejor de SQL es que puede manejar enormes base de datos con miles de millones de filas. Además de ser superior es de bajo costo [12].

API-REST

Una API-REST es un conjunto de definiciones y protocolos que se utiliza para desarrollar e integrar el software de las aplicaciones. API significa interfaz de programación de aplicaciones [13].
Las API permiten que sus productos y servicios se comuniquen con otros, sin necesidad de saber cómo están implementados. Esto simplifica el desarrollo de las aplicaciones y permite ahorrar tiempo y dinero. Las API le otorgan flexibilidad; simplifican el diseño, la administración y el uso de las aplicaciones, y proporcionan oportunidades de innovación, lo cual es ideal al momento de diseñar herramientas y productos nuevos (o de gestionar los actuales). A veces, las API se consideran como contratos, con documentación que representa un acuerdo entre las partes: si una de las partes envía una solicitud remota con cierta estructura en particular, esa misma estructura determinará cómo responderá el software de

la otra parte [13].
Debido a que simplifican la forma en que los desarrolladores integran los elementos de las aplicaciones nuevas en una arquitectura actual, las API permiten la colaboración entre el equipo comercial y el de TI. Las necesidades comerciales suelen cambiar rápidamente en respuesta a los mercados digitales en constante cambio, donde la competencia puede modificar un sector entero con una aplicación nueva. Para seguir siendo competitivos, es importante admitir la implementación y el desarrollo rápidos de servicios innovadores. El desarrollo de aplicaciones nativas de la nube es una forma identificable de aumentar la velocidad de desarrollo y se basa en la conexión de una arquitectura de aplicaciones de microservicios a través de las API [13].
Las API son un medio simplificado para conectar su propia infraestructura a través del desarrollo de aplicaciones nativas de la nube, pero también le permiten compartir sus datos con clientes y otros usuarios externos. Las API públicas representan un valor comercial único porque simplifican y amplían la forma en que se conecta con sus partners y, además, pueden rentabilizar sus datos (un ejemplo conocido es la API de Google Maps) [13].

Material UI

Material-UI comenzó en 2014 para unificar React y Material Design. Hoy, Material-UI se ha convertido en una de las bibliotecas de React UI más populares del mundo, respaldada por una comunidad vibrante de más de 1 millón de desarrolladores en más de 180 países [14].

Bootstrap

Bootstrap es un marco front-end gratuito para un desarrollo web más rápido y fácil el cual incluye plantillas de diseño basadas en HTML y CSS para tipografía, formularios, botones, tablas, navegación, modales, carruseles de imágenes y muchos otros, así como complementos de JavaScript opcionales, además, brinda la capacidad de crear fácilmente diseños receptivos [15].

React JS

React crea interfaces de usuario interactivas de forma sencilla. Diseña vistas simples para cada estado en la aplicación, y React se encarga de actualizar y renderizar de manera eficiente los componentes correctos cuando los datos cambien. Las vistas declarativas hacen que el código sea más predecible, por lo tanto, fácil de depurar.
React crea componentes encapsulados que manejan su propio estado, y los convierte en interfaces de usuario complejas. Ya que la lógica de los componentes está escrita en JavaScript y no en plantillas, se puede pasar datos de forma sencilla a través de la aplicación y mantener el estado fuera del DOM [16].

Redux

Redux es una biblioteca independiente que se puede usar con cualquier capa o marco de la interfaz de usuario, incluidos React, Angular, Vue, Ember y vanilla

JS. Aunque Redux y React se usan comúnmente juntos, son independientes entre sí. Se utiliza Redux con cualquier tipo de marco de UI. Normalmente se utiliza una biblioteca de "enlace de UI" para vincular Redux con un marco de UI, en lugar de interactuar directamente con la tienda desde el código de UI. React Redux es la biblioteca de enlace oficial de Redux UI para React [17].

Código QR

El código QR (Quick Response Barcode o código de barra de respuesta rápida), se creó en Japón, en 1994. Al igual que el resto de códigos de barra, contiene o almacena información pero, en este caso, es posible almacenar más de 4200 caracteres alfanuméricos. Para lograrlo, el código QR se representa utilizando una matriz de puntos. Se caracteriza por los tres cuadros que se encuentran en las esquinas y que permiten detectar la posición del código al lector. Hoy en día este tipo de códigos se pueden ver en folletos, carteles, publicidad etc., permitiendo interactuar con el mundo a través de un smartphone, ya que una vez escaneado el código con el teléfono, gracias a su cámara y a un lector compatible que se pueda conseguir de manera rápida y gratuita, se decodifica instantáneamente su contenido.
Para que un código QR sea legible una vez impreso es recomendable que su tamaño no sea inferior a $3cm^2$; aunque un código de 1,5 cm de lado también es utilizado, estos últimos no soportan la misma cantidad de caracteres ni muchas personalizaciones [18].

JWT

JSON Web Token (JWT) es un estándar abierto (RFC 7519) que define una forma compacta y autónoma para transmitir información de forma segura entre las partes como un objeto JSON. Esta información se puede verificar y confiar porque está firmada digitalmente. Los JWT se pueden firmar usando un secreto (con el algoritmo HMAC) o un par de claves pública/privada usando RSA o ECDSA [19].
Aunque los JWT se pueden cifrar para proporcionar también secreto entre las partes, nos centraremos en los tokens firmados . Los tokens firmados pueden verificar la integridad de los reclamos que contiene, mientras que los tokens cifrados ocultan esos reclamos de otras partes. Cuando los tokens se firman utilizando pares de claves públicas/privadas, la firma también certifica que solo la parte que posee la clave privada es la que la firmó [19]. Estos son algunos escenarios en los que los tokens web JSON son útiles:

- **Autorización:** este es el escenario más común para usar JWT. Una vez que el usuario haya iniciado sesión, cada solicitud posterior incluirá el JWT, lo que le permitirá acceder a rutas, servicios y recursos que están permitidos con ese token. El inicio de sesión único es una característica que utiliza ampliamente JWT hoy en día, debido a su pequeña sobrecarga y su capacidad de usarse fácilmente en diferentes dominios [19].

- **Intercambio de información:** los tokens web JSON son una buena forma de transmitir información de manera segura entre las partes. Debido a que los JWT pueden firmarse, por ejemplo, utilizando pares de claves públicas / privadas, puede estar seguro de que los remitentes son quienes dicen ser. Además, como la firma se calcula utilizando el encabezado y la carga útil, también puede verificar que el contenido no haya sido alterado [19].

Web Sockets

WebSockets es una tecnología avanzada que hace posible abrir una sesión de comunicación interactiva entre el navegador del usuario y un servidor. Con esta API, puede enviar mensajes a un servidor y recibir respuestas controladas por eventos sin tener que consultar al servidor para una respuesta [20].

Javascript

JavaScript (JS) es un lenguaje de programación ligero e interpretado, orientado a objetos con funciones de primera clase. Aunque es más conocido como el lenguaje de scripting para páginas web, muchos entornos no relacionados con el navegador también lo usan, tales como node.js, Apache CouchDB y Adobe Acrobat. Es un lenguaje script multiparadigma, basado en prototipos, dinámico, soporta estilos estilos orientados a objetos, imperativos y declarativos [21].
El estándar de JavaScript es ECMAScript. Desde el 2012, todos los navegadores modernos soportan completamente ECMAScript 5.1. Los navegadores más antiguos soportan por lo menos ECMAScript 3. El 17 de Julio de 2015, ECMA International publicó la sexta versión de ECMAScript, la cual es oficialmente llamada ECMAScript 2015, y fue inicialmente nombrada como ECMAScript 6 o ES6. Desde entonces, los estándares ECMAScript están en ciclos de lanzamiento anuales. Esta documentación se refiere a la última versión del borrador, que actualmente es ECMAScript 2019 [21].
JavaScript no debe ser confundido con el lenguaje de programación Java. Ambos "Java" y "Javascript" son marcas registradas de Oracle en Estados Unidos y otros países. Sin embargo, los dos lenguajes de programación tienen sintaxis, semántica y usos muy diferentes [21].

Axios

Se trata de una librería Javascript capaz de ejecutarse tanto en el navegador como en NodeJS, que facilita todo tipo de operaciones como cliente HTTP. Con Axios puedes realizar solicitudes contra un servidor, completamente configurables, y recibir la respuesta de una manera sencilla de procesar. La librería está basada en promesas, por lo que al usarla se puede generar un código asíncrono bastante ordenado. Incluso es capaz de usar Async / Await para mejorar la sintaxis, aunque en ese caso se estaría obligado a transpilar el código para que fuera compatible con navegadores [22].
Axios es una alternativa que ofrece diversas ventajas:

- Ofrece una API unificada para las solicitudes Ajax [22].

- Está altamente pensado para facilitar el consumo de servicios web, API REST que devuelvan datos JSON [22].
- Es muy sencillo de usar y puede ser un complemento ideal para sitios web convencionales, donde no se esté usando jQuery y aplicaciones frontend modernas basadas en librerías como React o Polymer, que no disponen en su "core" de mecanismos para hacer de cliente HTTP [22].
- Tiene muy poco peso (13Kb minimizado y todavía menos si contamos que el servidor lo envía comprimido), en unas pocas Kb nos ahorrará mucho trabajo de codificación en las aplicaciones [22].

1.3 Objetivos

1.3.1 General

- Implementar una aplicación para la gestión de órdenes en restaurantes de la ciudad de Ambato utilizando tecnología móvil.

1.3.2 Específicos

- Proponer un modelo de negocios para la gestión y toma de órdenes en los restaurantes de la ciudad de Ambato.
- Identificar una tecnología orientada a dispositivos móviles que permita el desarrollo de la aplicación propuesta.
- Desarrollar una aplicación para la gestión de órdenes en restaurantes de la ciudad de Ambato utilizando tecnología móvil.

CAPÍTULO II

Metodología

2.1 Materiales

El material para la recolección de la información será una única encuesta con el fin de aplicarla a clientes potenciales. La encuesta aplicada fue orientada hacia el uso de las nuevas tecnologías dentro de un restaurante. Se modelo de la siguiente manera.

¿En ocasiones su restaurante ha tenido inconvenientes para gestionar las ordenes?

- Si
- No

¿En ocasiones ha percibido que no cuenta con el personal necesario para la atención al cliente?

- Si
- No

¿Ha utilizado alguna aplicación para solicitar comida a domicilio?

- Si
- No

¿Qué le parece el modelo tradicional de toma de ordenes dentro de restaurantes?

- Bueno
- Malo
- Regular

¿Está de acuerdo que los restaurantes deberían incorporar nuevas tecnologías para ofertar sus servicios?

- Si
- No
- Tal vez

2.2 Métodos

2.2.1 Modalidad básica de la investigación

- **Investigación bibliográfica:** La investigación será bibliográfica ya que se enfocará en referencias de libros, revistas, revistas digitales, papers, tesis de grado relacionadas, documentos técnicos etc, para la construcción del marco teórico que se enfocará en la construcción de una aplicación web para tener un servicio optimizado para con los clientes.

- **Investigación aplicada:** Se realizará una investigación aplicada por la razón de que se aplicará todos los conocimientos obtenidos durante el trascurso de la carrera para obtener los resultados esperados que cubran las necesidades del desarrollo del presente proyecto.

2.2.2 Población y muestra

Población: Para la presente investigación la población será definida por los restaurantes de la ciudad de Ambato.
Muestra: Debido a la gran cantidad de restaurantes en la zona se tomará un grupo pequeño de restaurantes de la ciudad como muestra.

2.2.3 Recolección de información

Al aplicar la encuesta se obtuvo los siguientes datos.

Figura 2.1: "Pregunta 1"
Elaborado por: Juan Mesias

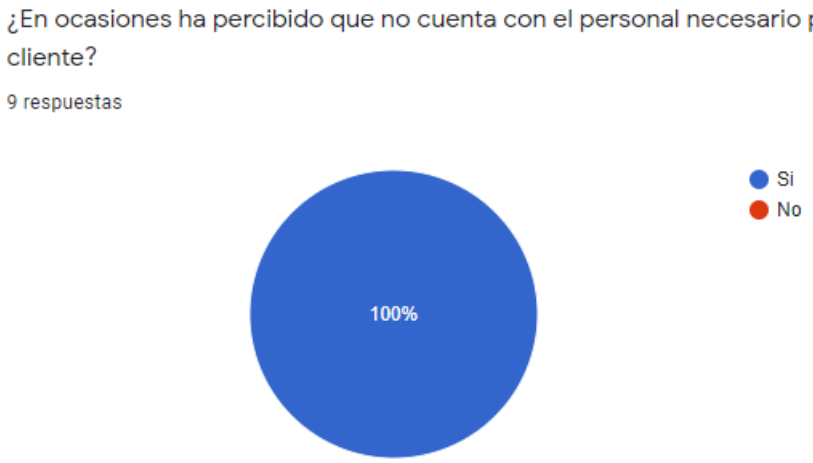


Figura 2.2: "Pregunta 2"
Elaborado por: Juan Mesías

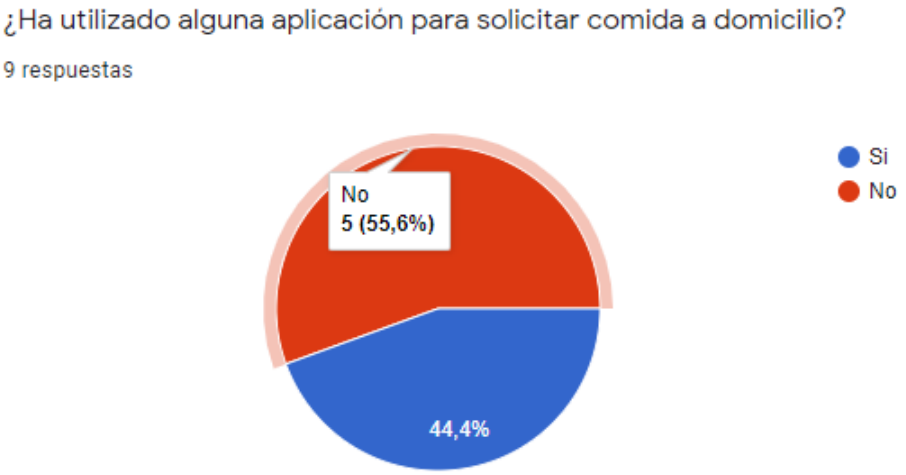


Figura 2.3: "Pregunta 3"
Elaborado por: Juan Mesías

Figura 2.4: "Pregunta 4"
Elaborado por: Juan Mesías

Figura 2.5: "Pregunta 5"
Elaborado por: Juan Mesías

2.2.4 Procesamiento y análisis de datos

La encuesta realizada demostró:

- De la totalidad de clientes potenciales encuestados la gran mayoría en ocasiones han tenido inconvenientes en la gestión de sus órdenes internamente, lo que consecuentemente causa inconformidad y mal estar en un cliente 2.1.
- En base a los datos recolectados en la segunda pregunta la totalidad de los encuestados admitieron que en ocasiones han sentido falta de personal para la atención al cliente, esto puede variar por diferentes factores (días de la semana, horas pico del día, eventos locales) 2.2.
- Uno de los factores fundamentales para la implementación de nuevas tecnologías en un área específica es la familiarización con las mismas; en la pregunta número tres se concibe que más de la mitad de encuestados está al tanto de tecnologías similares para la gestión de pedidos 2.3.
- La mayoría de encuestados sienten inconformidad con el modelo tradicional de atención al cliente en restaurantes el cual se supo manifestar que se trata de que un mesero se acerque hacia su mesa a recibir el pedido 2.4.
- En la pregunta cinco se indica que la totalidad de encuestados están de acuerdo que las nuevas tecnologías deberías ser parte de los restaurantes en la forma de ofertar sus servicios al público 2.5.

2.3 Desarrollo del Proyecto

La metodología seleccionada para el desarrollo del proyecto fue la de prototipo evolutivo, debido a que la metodología permite la interacción directa con el cliente. Además, que el desarrollo se simplifica ya que se empieza con los procesos que mejor son comprendidos por parte del desarrollador, de esta manera se presenta un prototipo inicial que posteriormente evolucionara hacia lo que se considera como un prototipo final con nuevos atributos y con la completa aprobación del cliente. La metodología fue adaptada para el desarrollo del proyecto con las siguientes etapas.

- **Definición del modelo de negocio**.
  Se justifica el funcionamiento y la eficiencia del modelo planteado enfocándose en los aspectos fundamentales del mismo.
  - Descripción del modelo de negocio.
- **Identificación de requerimientos del sistema**.
  Define el alcance del sistema con una descripción clara y precisa de lo que se quiere obtener como producto final.
  - Análisis de requerimientos del sistema.
  - Levantamiento de requerimientos funcionales.
  - Levantamiento de requerimientos no funcionales.

- **Diseño de la aplicación**.
  Propone los diseños básicos de estructuras de datos, comunicación, arquitectura de la aplicación y las actividades en los cuales se orientará la aplicación.
  - Arquitectura de la aplicación.
  - Roles de usuario.
  - Modelo fisico de la base de datos.
  - API-REST.
  - Diagramas de actividad.
  - Diagramas de secuencia.
- **Estudio y elección de herramientas para el desarrollo**.
  Se identifica las herramientas adecuadas para el desarrollo del software.
  - Herramientas para Backend.
  - Herramientas para el Frontend.
- **Desarrollo de la aplicación**.
  Se describe el desarrollo de la aplicación.
  - Desarrollo de casos de uso.
  - Diagramas de clases.
  - Desarrollo de software.
- **Pruebas de funcionalidad**.
  Se evalua el prototipo final.
  - Pruebas de funcionalidad del prototipo terminado.
  - Pruebas de estres.

CAPÍTULO III

Resultados y discusión

El desarrollo de la aplicación partió de la definición de un prototipo funcional. A través de un proceso iterativo e incremental, con la participación de usuarios potenciales, el prototipo fue evolucionando a la aplicación resultante. La concepción de la aplicación tomó como punto de partida otras aplicaciones en línea con una funcionalidad similar. A continuación se describen las etapas por las que transitó el desarrollo del software en cuestión.

3.1 Definición del modelo de negocio

El modelo de negocio es una herramienta importante que permitirá prever con claridad y exactitud que es lo qué se va a ofrecer al mercado, cómo se va a ofrecer y a qué área del mercado va dirigido. De esta manera, el modelo de negocio provee a la empresa de las bases sobre las cuales la misma crea, proporciona y capta valor [23]. La mejor manera de representar un modelo de negocio es considerando los nueve aspectos básicos en los que una empresa verán reflejado su manera de conseguir ingresos económicos y la manera de llegar al cliente. Las cuatro áreas principales cubiertas por un modelo de negocio deben ser: clientes, oferta, infraestructura y la viabilidad económica.
Los aspectos que el modelo de negocio cubre son:

- Segmento de mercado (SM).
- Propuesta de valor (PV).
- Canales (C).
- Relaciones con los clientes (RCI).
- Fuentes de ingresos (FI).
- Recursos clave (RC).
- Actividades clave (AC).
- Asociaciones clave (AsC).
- Estructura de costes (EC).

El modelo de negocio con la propuesta de valor planteado se representa en la figura 1.1.

<table>
<tr>
<td rowspan="2">Asociación Clave

Restaurantes de la ciudad de Ambato.
Clientes de restaurantes con la necesidad de un servicio rápido y de calidad.
Usuarios de smartphones con acceso a internet.</td>
<td>Actividades Clave

Optimizar la toma y gestión de ordenes dentro de restaurantes a través de un aplicativo web, brindando al cliente una mejor experiencia.
Identificación univoca de las mesas del restaurante por medio de un código QR.</td>
<td rowspan="2">Propuesta de Valor

Las dificultades en la toma de pedidos simultáneos dentro de los restaurantes, de esta manera el cliente tendrá la posibilidad que a travéz de una aplicación se le permita realizar el pedido de manera inmediata y que el mismo sea entregado a la mesa indicada, brindándole al cliente un servicio más rápido y cómodo.</td>
<td>Relaciones con Clientes

Servicio automatizado por medio de dispositivos móviles.
Cuentas personales dentro del aplicativo.</td>
<td rowspan="2">Segmentos de Mercado

Cualquier restaurante con la necesidad de mejorar la atención y disminuir el tiempo de atención a los usuarios.</td>
</tr>
<tr>
<td>Recursos Clave

Tangibles
Desarrollador.
Equipo de cómputo.

Intangibles
Internet.
Hosting.
Base de datos.
Software libre.</td>
<td>Canales

Canales digitales.</td>
</tr>
<tr>
<td colspan="2">Estructura de Costes

Desarrollador.
Internet.
Hosting.
Base de datos.
Software libre.</td>
<td colspan="3">Fuente de Ingresos

Porcentaje del valor de ventas efectuadas a través del aplicativo.</td>
</tr>
</table>

Figura 1.1: Modelo de negocio
Elaborado por: Juan Mesias

A partir de la concepción de un modelo canvas en el cual se identifica los nueve factores esenciales de un modelo de negocio, se procedió a analizar los costos fijos y los costos variables que deberán ser cubiertos al implantar el modelo propuesto, también se debe tener en cuenta la utilidad estimada por cada una de las implementaciones para de esta manera lograr un punto de equilibro en el cual quede demostrado que la implementación del modelo indicado llega a ser eficiente.

Costos fijos

Los costos fijos cubren elementos que estaran constantemente presentes durante la implantación del modelo de negocio como se demuestra en la figura 1.2.

Cantidad	Maquinaria	Valor del bien	Vida util	Depreciacion A	Deprecion M
1	equipo de computo	$700,00	3	$233,33	$19,44
	total	$700,00		$233,33	$19,44
Cantidad	**Cargo**	**Sueldos mensuales**			
1	Desarrollador	$400,00			
	total	$400,00			
Cantidad	**COSTOS de administracion**				
	publicidad	$10,00			
	suministros de oficina	$5,00			
	total	$15,00			
	Servicios basicos				
	telefonia e internet	$20,00			
	Energia Electrica	$20,00			
	total	$40,00			

COSTOS FIJOS	$474,44
COSTOS VARIABLES	$20,00
COTOS TOTALES	$494,44
COSTO UNITARIO	$0,02

Figura 1.2: Costos Fijos
Elaborado por: Juan Mesias

Costos variables

Los costos variables son aquellos que eventualmente con el paso del tiempo y la cantidad de implementaciones existentes pudiera incrementar o decrementar. El margen de utilidad se obtiene con un doce por ceinto de las ventas totales realizadas por la aplicación.1.3

Producto	Cantidad mensual	Precio	Cantidad	Precio	Costos por Aplicación
Hosting	1	$12,00	1	$12,00	$12,00
Base de Datos	1	$8,00	1	$8,00	$8,00
				Total	$20,00

COSTOS VARIABLES UNITARIOS	$20,00
UTILIDAD	$200,00
PRECIO DE VENTA	$220,00

180 12% de las ventas del restaurante mediante
200,00 la aplicación.

Figura 1.3: Costos Variables
Elaborado por: Juan Mesias

Una vez definido los costos y el margen de utilidad se procede a calcular el punto de equilibrio.1.4

DATOS		
Cvu	$20,00	
MU	$200,00	MU=PV-CVU
CF	$477,22	
PVP	$220,00	PV=MU+CVU
CVt	$ 47,72	CVT=CVU*Q

	Q=CF/PVP-Cv	Q=CF/MU	
PUNTO DE EQ	2,386	2,386	Unidades mensuales
INGRESOS	$524,94		
COSTO TOTAL	$524,94	CT=CF+CVT	
UTILIDAD	$ -	U=I-CT	

Figura 1.4: Punto de Equilibrio
Elaborado por: Juan Mesias

Con los costos calculados y la utilidad estimada se establece el punto de equilibrio en un 2,38 de unidades mensuales lo que se traduciria a que 3 aplicaciones mensuales serian necesarias para comenzar a generar una utilidad a favor a partir del modelo propuesto.

La definición formal del modelo de negocio que funcionará en los restaurantes que se acojan a la presente propuesta se representó utilizando BPM [24] tal y como se muestra en la figura 1.5. A través de esta figura se muestra la estructura funcional del proceso a automatizar para obtener un mejor control sobre el mismo.

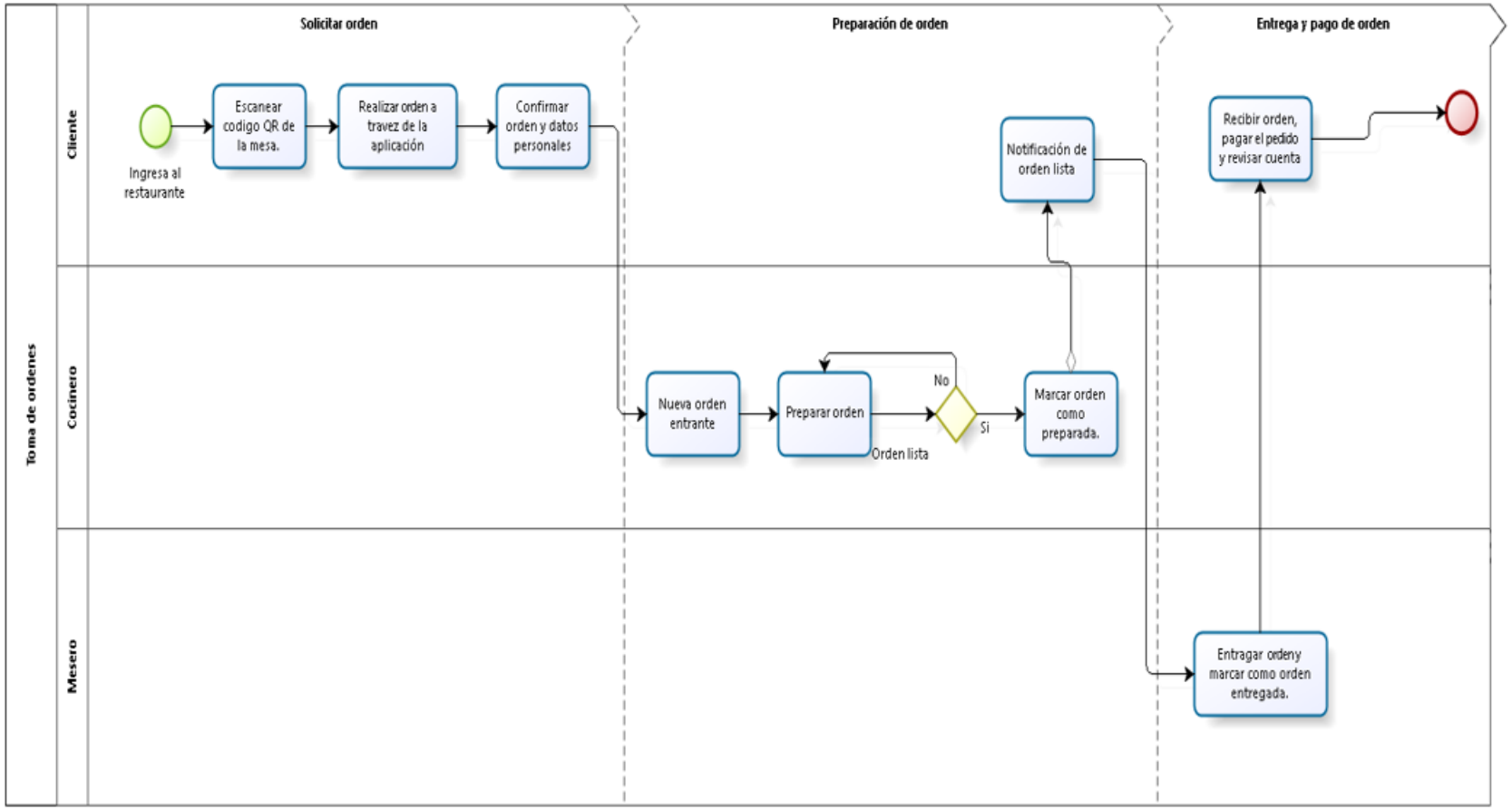


Figura 1.5: Formalización del Modelo de Negocio
Elaborado por: Juan Mesias

3.2 Identificación de requerimientos

3.2.1 Análisis de requerimientos del sistema

Los factores de mayor importancia para una eficiente atención al cliente de un restaurante son el precio, el tiempo de atención, el ambiente y la calidad de la comida [25]. Partiendo de la experiencia personal, al visitar diferentes restaurantes del área, se ha logrado identificar la dificultad para la toma y gestión de órdenes simultaneas dentro de los mismos en momentos concurridos del día.

A partir de una observación minuciosa en el área, se pudo identificar que la incorporacion de nuevas tecnologías como parte de su gestión interna en lo que se refiere a la toma y manejo de pedidos, proporcionaría un claro valor agregado frente a la competencia.

El análisis de los restaurantes condujo a realizar pruebas en aplicaciones en línea de grandes restaurantes. Dichas aplicaciones están enfocadas en el servicio de entrega a domicilio, pero no cubren la perspectiva de un cliente dentro del restaurante. Se puede decir que la atención al cliente presente en el restaurante sigue teniendo lugar de manera tradicional; un empleado de la entidad se acerca y toma el pedido del cliente. En la mayoría de los casos lo hace a mano en un papel; esto incide en los prolongados tiempos de espera por parte de los clientes, lo que causa una sensación de desatención y servicio deficiente.

Como resultado, se decidió optar por un aplicativo multiplataforma que cubra distintas etapas de la gestión de toma de órdenes dentro del restaurante, enfocándose en la recepción del pedido y la gestión del menú a ofrecer. De esta manera se busca mejorar la atención al cliente y por ende reducir los tiempos de atención al mismo. Desde el punto de vista técnico, el proyecto quedó enfocado hacia el desarrollo de software libre, así como la disponibilidad del aplicativo en diversas plataformas a manera de una PWA [26]

3.2.2 Levantamiento de requerimientos funcionales

La disponibilidad del aplicativo debe tener soporte para todas las plataformas, enfocándose en un diseño responsivo y amigable. Y estos son los requisitos funcionales para la correcta implementación de la aplicación.

- Registrar usuario.
- Autenticar usuario.
- Mostrar información de productos disponibles.
- Leer código QR que identifica a la mesa donde se le presta atención.
- Recibir el pedido.
- Actualizar los datos del perfil de usuario.
- Agregar datos de facturación del usuario.

- Revisar los pedidos realizados por un usuario autenticado.
- Modificar la información del restaurante de un usuario autenticado como dueño.
- Gestionar el estado de los pedidos para llevar.
- Gestionar el estado de los pedidos para consumo en el local.
- Listar los pedidos realizados.
- Actualizar categorías de productos.
- Actualizar datos de productos.
- Actualizar registro de mesas del restaurante.
- Actualizar roles de usuarios del sistema.

3.2.3 Levantamiento de requerimientos no funcionales

Los requerimientos no funcionales para el desarrollo del aplicativo que se tuvieron en cuenta fueron:

- El aplicativo debe estar disponible para todos los dispositivos, por lo que debe tener un diseño responsivo.
- Utilizar Material-UI para garantizar la estabilidad de diferentes componentes.
- Garantizar la seguridad de los datos del usuario.
- Aplicativo amigable con el usuario.

3.3 Diseño de la aplicación

3.3.1 Arquitectura de la aplicación

En la figura 3.6 se muestra la arquitectura de la aplicación, tomando en cuenta los factores relevantes para el correcto funcionamiento de la aplicación.

El backend de la aplicación es una API-REST; una interfaz para acceder a través de HTTP a los servicios que expone determinado gestor de información; el formato más común para el trasiego de los datos es JSON [27].
El frontend está representado por una biblioteca javascript que se encarga de conectarse al backend mediante los siguientes pasos:

1. El frontend realiza peticiones HTTP a la API-REST con los métodos GET, POST, PUT y DELETE.
2. La API-REST realiza las acciones de recuperación y actualización de información en la base de datos.
3. La API-REST devuelve al frontend los datos resultantes de la operación solicitada.
4. El frontend interpreta los datos devueltos por la API-REST.

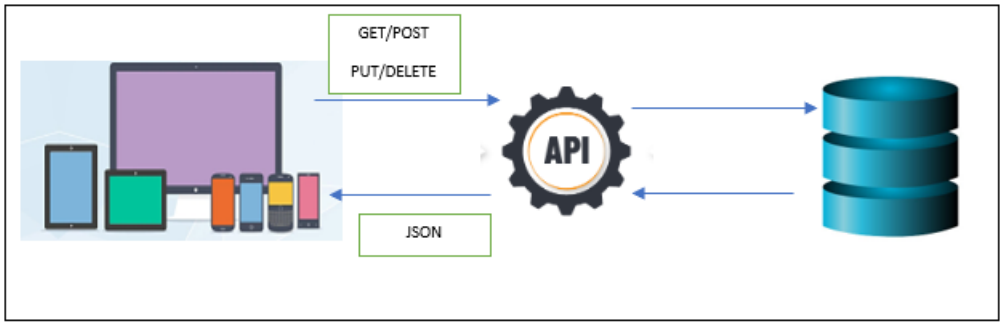


Figura 3.6: Arquitectura de la aplicación
Elaborado por: Juan Mesias

3.3.2 Roles de usuario

En la tabla siguiente se muestra los roles de usuario que maneja la aplicación.

Rol de usuario	Descripción
Administrador	Encargado de la actualización de la información del restaurante, dígase: • Información general de la entidad. • Identificadores de las mesas. • Información de categorías de productos y productos por categoría. • Estado de los pedidos.
Cliente	Gestiona la información personal y de facturación, además de los pedidos, de un cliente en particular.

Tabla 3.1: Roles en la Aplicación
Elaborado por: Juan Mesias

3.3.3 Modelo físico de la base de datos

El diseño de la base de datos propuesta se detalla en la fig 3.7.

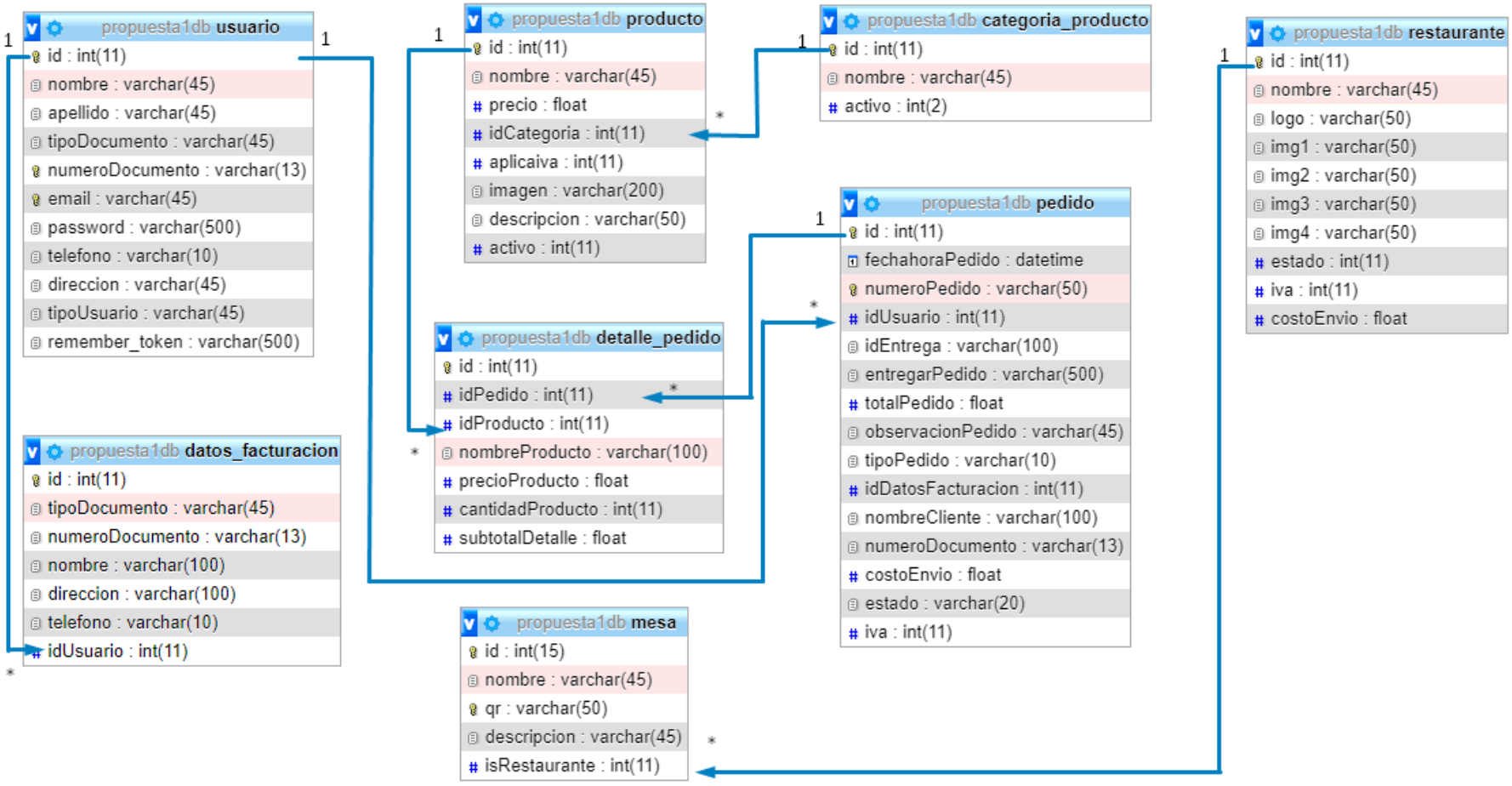


Figura 3.7: Modelo físico de la base de datos
Elaborado por: Juan Mesias

Un restaurante consta de múltiples mesas desde las que se da servicio a los clientes registrados en la aplicación. Los clientes generan pedidos de productos que están organizados por categorías. Tanto las categorías de productos como los productos en sí pueden activarse/desactivarse en función de su disponibilidad y/o decisión del personal que administra el restaurante. El precio de los productos puede variar en el tiempo, razón por la cual el modelo de datos registra el precio actual de cada producto y el precio de cada producto en cada pedido de manera independiente. Lo mismo ocurre con los datos de facturación de cada cliente, los que se gestionan independientemente, a nivel de cliente y de cada pedido de este.

En el caso de los productos, el precio vigente en el momento en que se hace un pedido es el que se utiliza para registrar dicho pedido. En el caso de los datos de facturación, los datos vigentes para el cliente activo se establecen, por defecto, como datos de facturación. No obstante, antes de proceder con la facturación el usuario puede decidir cambiar estos datos para la factura que se emitirá en cada momento.

El costo de envío registrado en la tabla pedido permite que el sistema se utilice para envíos a domicilio y la relación entre mesa y restaurante garantiza la escalabilidad de la aplicación para utilizar una misma base de datos en la gestión de múltiples restaurantes.

3.3.4 API-REST

La implementación de la API-REST se llevó a cabo utilizando el framework *Laravel*, el cual sigue el modelo MVC (Modelo Vista Controlador) e incluye herramientas propias para la autenticación y enrutamiento, lo que facilita el despliegue seguro de los servicios web implementados [28].

La comunicación con la API-REST se establece de dos maneras; la primera es la manera tradicional, caracterizada por el envío de una solicitud y el procesamiento de una respuesta a la misma. La segunda manera de comunicación es por WebSockets; al establecer una conexión, esta se mantiene abierta para la recepción de datos que se envíen desde el otro extremo del canal. El procesamiento de los datos tiene lugar en tiempo real. Estos dos tipos de comunicación, soportados por la API-REST implementada, se representan en la figura 3.8.

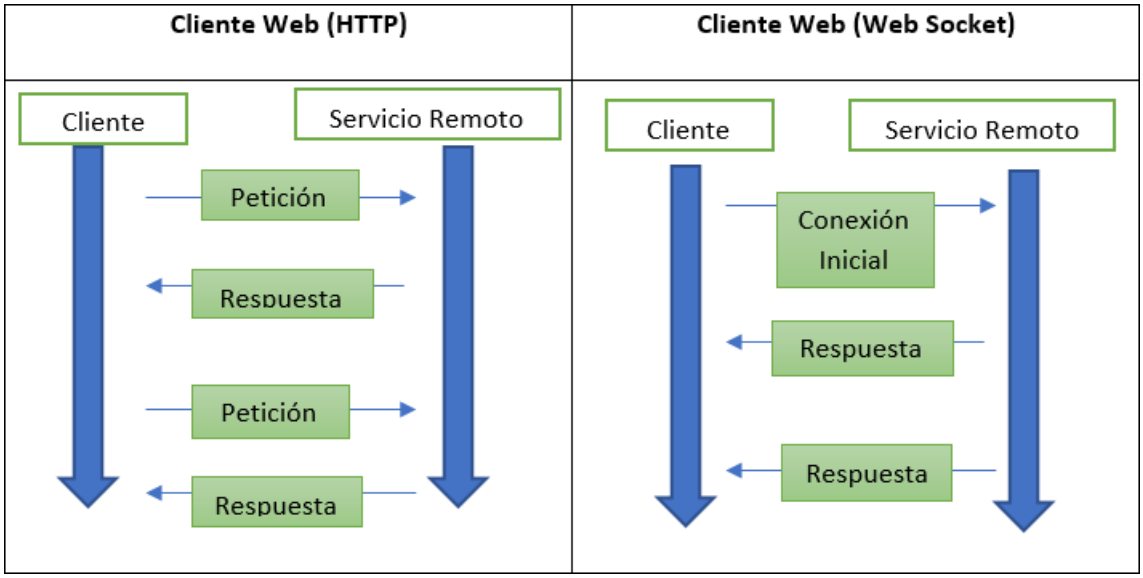


Figura 3.8: Esquemas de comunicación soportados
Elaborado por: Juan Mesias

En la tabla a continuación se documenta el diseño de los *endpoints* definidos en la API-REST implementada. Para información más detallada sobre el desarrollo de la API-REST se sugiere consultar el apéndice A.

Ruta	**Método**	**Parámetro**	**Descripción**
Usuarios			
'usuario/login'	post	sp	Permite iniciar sesión al usuario.
'usuario/logout'	post	sp	Permite al usuario cerrar sesión.
'usuario/registrar'	post	Obejto de usuario	Registra un nuevo usuario.
'usuarios'	get	sp	Lista los usuarios.
'usuarios/id'	put	Id de usuario y objeto de usuario	Actualiza un usuario.
'usuariosById'	get	Id del usuario	Obtiene la información de un usuario especifico.
'datosFacturacion/id	get	Id del usuario	Obtiene los datos de facturación de un usuario.
'datosFacturacion'	post	Objeto de dato de facturación	Inserta un nuevo dato de facturación.
Restaurante			
'restaurante'	get	sp	Obtiene la información del restaurante.
'restaurante/update/id'	put	Objeto de restaurante	Actualiza la información del restaurante.

Categorías			
'categorias'	get	sp	Lista las categorías.
'categorias'	post	Objeto de categoría	Inserta una nueva categoría.
'categorias/id'	put	Id de categoría y objeto de categoria	Actualiza una categoría.
'productos/id	delete	Id de la categoria	Elimina una categoria.
Mesas			
'mesas'	get	sp	Lista las mesas.
'mesas'	post	Objeto de mesa	Inserta una nueva mesa.
'mesas/id'	put	Id de mesa y objeto de mesa	Actualiza una mesa.
'mesas/id'	delete	Id de la mesa	Elimina una mesa.
Productos			
'productos'	get	sp	Lista los productos.
'productos/store'	post	Objeto de producto	Inserta un nuevo producto.
'productos/update/id'	put	Id del producto y objeto de producto	Actualizar un producto.
'productos/id'	delete	Id del producto	Elimina un producto.
Órdenes			
'pedidos'	post	sp	Inserta un nuevo pedido.
'pedidosPreparar'	get	sp	Obtiene los productos para preparar.
'pedidosEntregar'	get	sp	Obtiene los productos para entregar.
'pedidosByUsuario/id'	get	Id del usuario	Obtiene los pedidos de un usuario.
'pedidosGeneral'	get	sp	Obtiene todos los pedidos.
'actulizarPedido/id'	put	Id del pedido objeto de pedido	Actualiza el estado de un pedido.
Canales para WebSockets			
'pedidos'		Emitira la información al ingresar una nueva orden.	
'pedidoPreparado'		Emitira la información cuando la orden sea marcada como preparada.	
'pedidoCancelado'		Emitira la información cuando la orden sea marcada como cancelada.	

'pedidoEntregado'	Emitira la información cuando la orden sea marcada como entregada.

Tabla 3.2: Diseño de la API-REST
Elaborado por: Juan Mesias

3.3.5 Diagramas de Actividades de la Aplicación

En primera instancia, al iniciar la aplicación, se verifica la existencia de un token de acceso almacenado en el LocalStorage de la aplicación. Si existe dicho token, se tratará de hacer un auto inicio de sesión del usuario. Caso contrario, la aplicación procederá a mostrar la información correspondiente a las categorías y productos disponibles para ser añadidos a un pedido y mostrará las opciones de inicio de sesion y registro, tal y como se documenta en la figura 3.9.

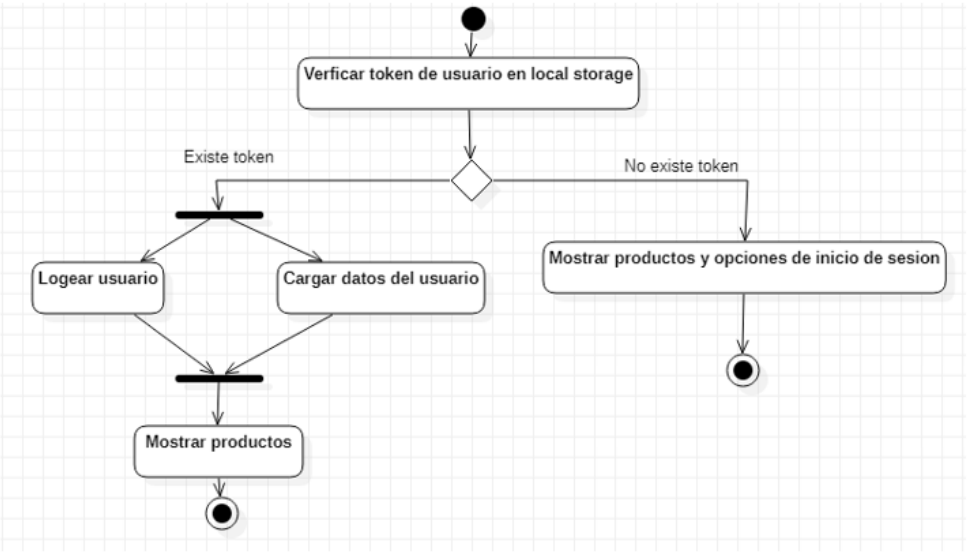


Figura 3.9: Verificación de Token de usuario
Elaborado por: Juan Mesias

El flujo determinado para el inicio de sesión o registro de un nuevo usuario, se determina de la manera convencional. El usuario ingresa los datos; estos son validados en el frontend y, posteriormente, se realiza la solicitud HTTP para validar el registro o el inicio de sesión del lado de la API-REST. Revisar figura 3.10

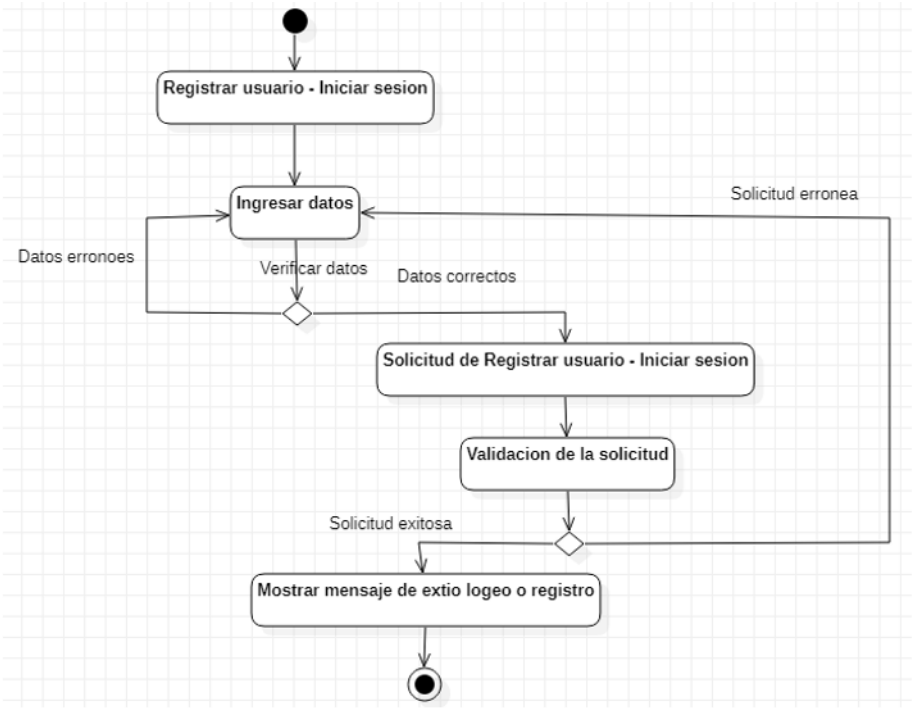


Figura 3.10: Registro de usuario - Inicio de sesión
Elaborado por: Juan Mesias

Las peticiones de datos o solicitudes de información se realizan a través de una solicitud a la API-REST. El resultado que se recupera es mostrado en la aplicación con el formato requerido. Las peticiones de información en la aplicación se realizan de dos maneras: con y sin parámetros. Las solicitudes sin parámetros se realizan al cargar la aplicación para mostrar la información general del restaurante y los productos disponibles por categorías. De esta manera, la aplicación muestra su contenido inicial. Otra manera de petición de información es sin parámetros es la conexión que se establece mediante web sockets, a partir de una conexión inicial el traspaso de datos de manera bidireccional se mantiene abierto como se muestra en la figura 3.8. Las peticiones bajo parámetros son realizadas por la aplicación para cargar la información específica del usuario como sus datos personales. El diagrama para una peticion de infromación se representa en la figura 3.11.

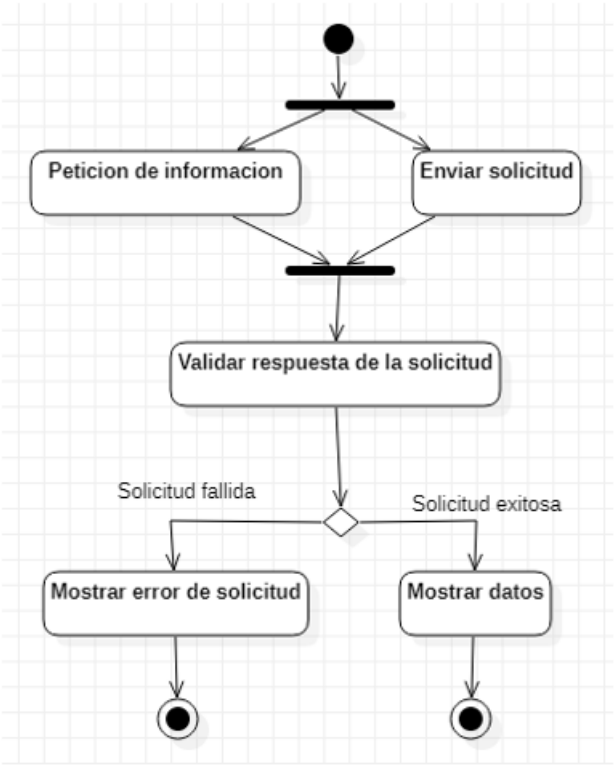


Figura 3.11: Peticiones de datos - Solicitudes de información
Elaborado por: Juan Mesias

El ingreso de datos a la aplicación se realiza con una estructura similar que actúa a través de peticiones o solicitudes a la API-REST para almacenar los datos en la base de datos. El cliente ingresa los datos requeridos para agregar una nueva entidad; estos son validados en el frontend y también son validados en la API-REST. Revisar figura 3.12.

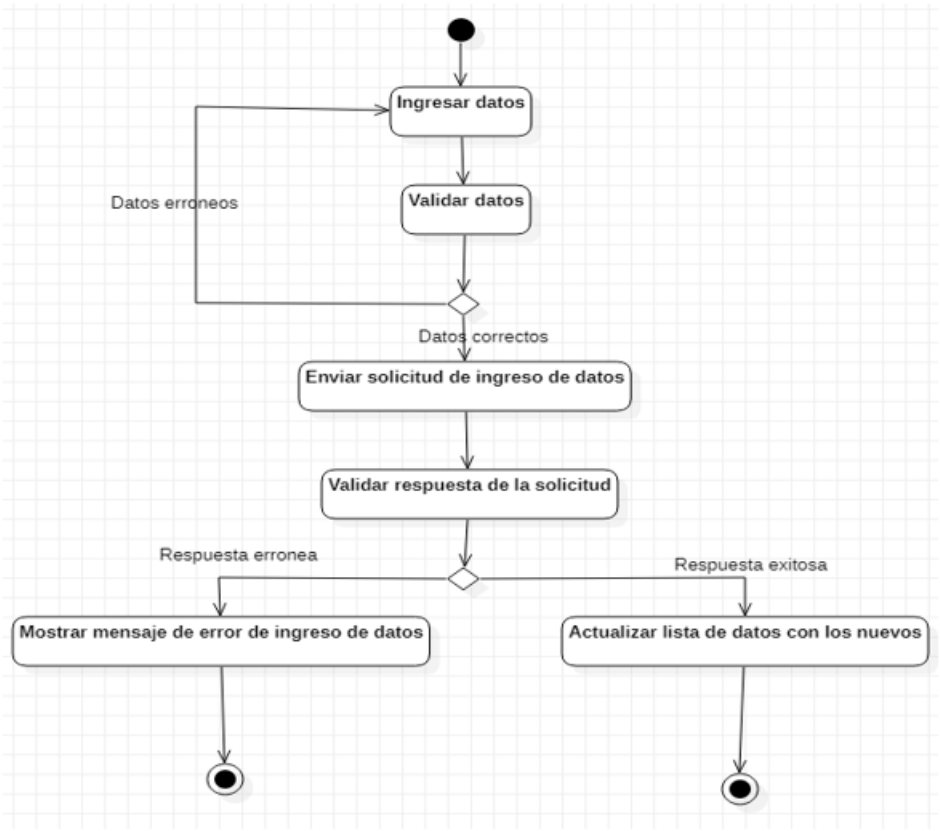


Figura 3.12: Ingreso de datos
Elaborado por: Juan Mesias

La modificación de los datos será dependiendo del perfil que presenta el usuario, ya sea administrador o usuario normal, y sigue la secuencia definida para los anteriores procesos, con la variación de que el usuario deberá elegir primero el dato a modificar. De esta manera, la modificación de información puede ser parcial o total. 3.13.

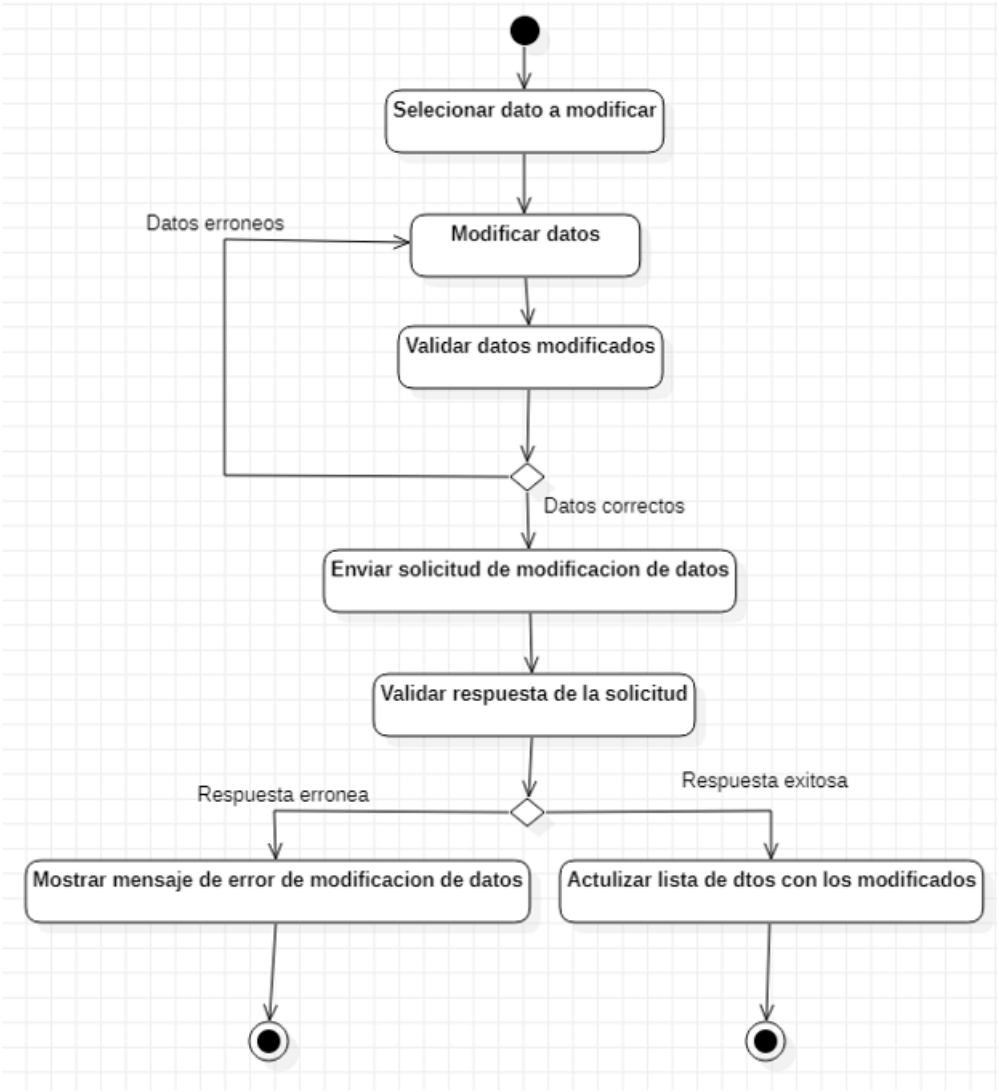


Figura 3.13: Modificación de datos
Elaborado por: Juan Mesias

Uno de los procesos más importantes de la aplicación se representa en una secuencia que permite al usuario realizar el pedido de uno o varios productos a la mesa en la que este se encuentra. El proceso empieza cuando el usuario está sentado en una de las mesas que contienen un código QR para ser escaneado, lo cual sirve como identificador para saber en qué mesa el usuario está esperando por su pedido. El usuario deberá escanear ese código QR a través de la aplicación, después de lo cual procederá a seleccionar los productos de la lista presentada en el aplicativo. Una vez que se tenga listo el pedido de productos, con su cantidad y precio total , el usuario deberá proceder a confirmar su orden, para lo cual deberá haber iniciado sesión previamente. Después de confirmar su pedido, el usuario deberá confirmar los datos con que quiere que se emita la factura correspondiente. Este proceso quedará registrado en la base de datos como un nuevo pedido. Revisar figura 3.14.

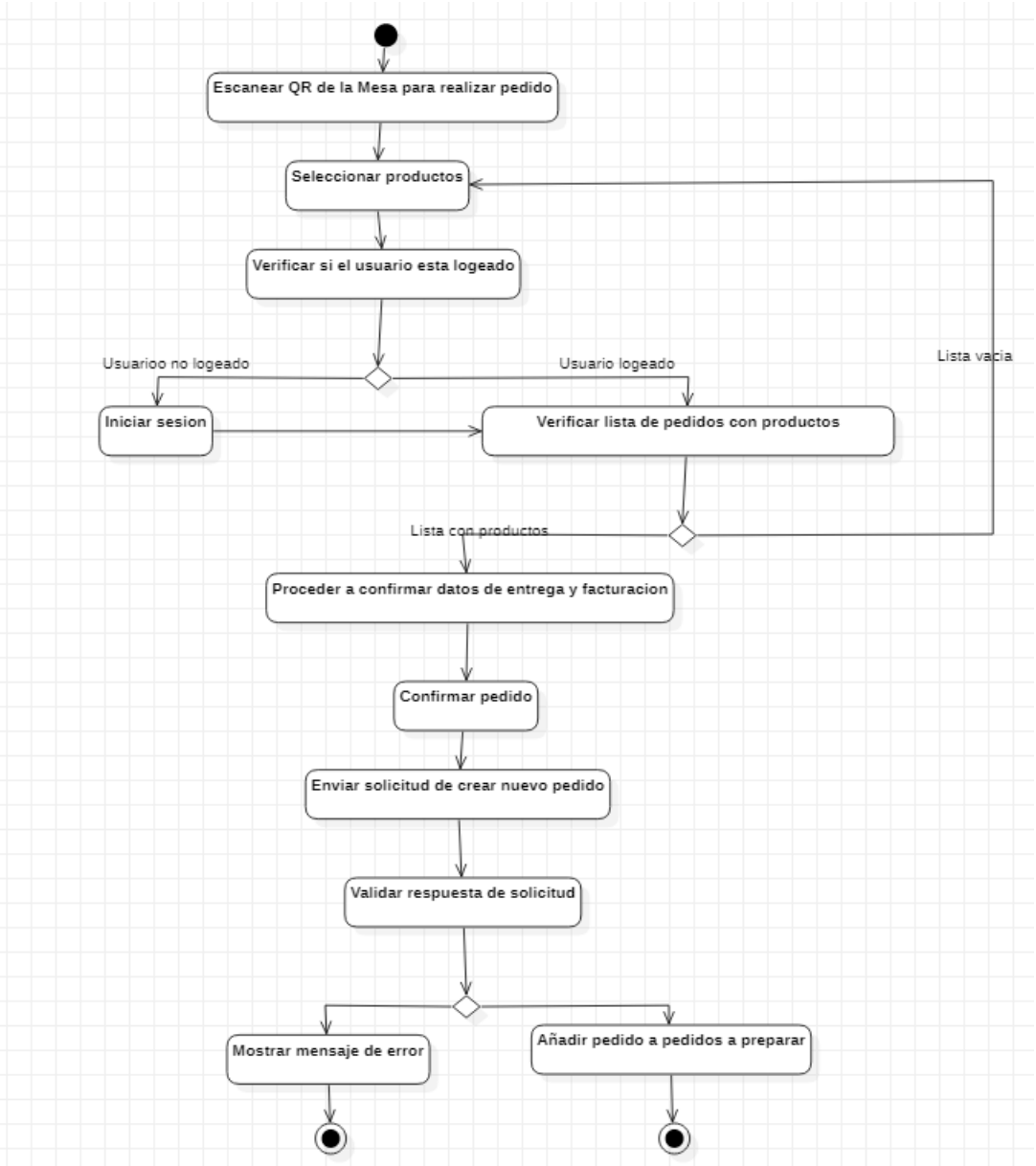


Figura 3.14: Realizar Pedido
Elaborado por: Juan Mesias

El pedido realizado por parte del cliente estará disponible en tiempo real para el personal de administración. A partir de la generación de un nuevo pedido recibido, el administrador tiene la opción de revisar el detalle del pedido, cancelar el pedido o cambiar su estado a preparado. En caso de que el pedido sea marcado como preparado se enviará una solicitud para modificar su estado, una vez correcta la solicitud se envia una notificación al usuario y el pedido inmediatamente es eliminado de la lista de pedidos a preparar y agregado a la lista pedidos para entregar todo en tiempo real. Cuando el pedido se encuentra en la lista de pedidos a entregar el administrador solo tendrá la opción de marcar el pedido como entregado la cual dará fin al seguimiento del pedido. Figura 3.15.

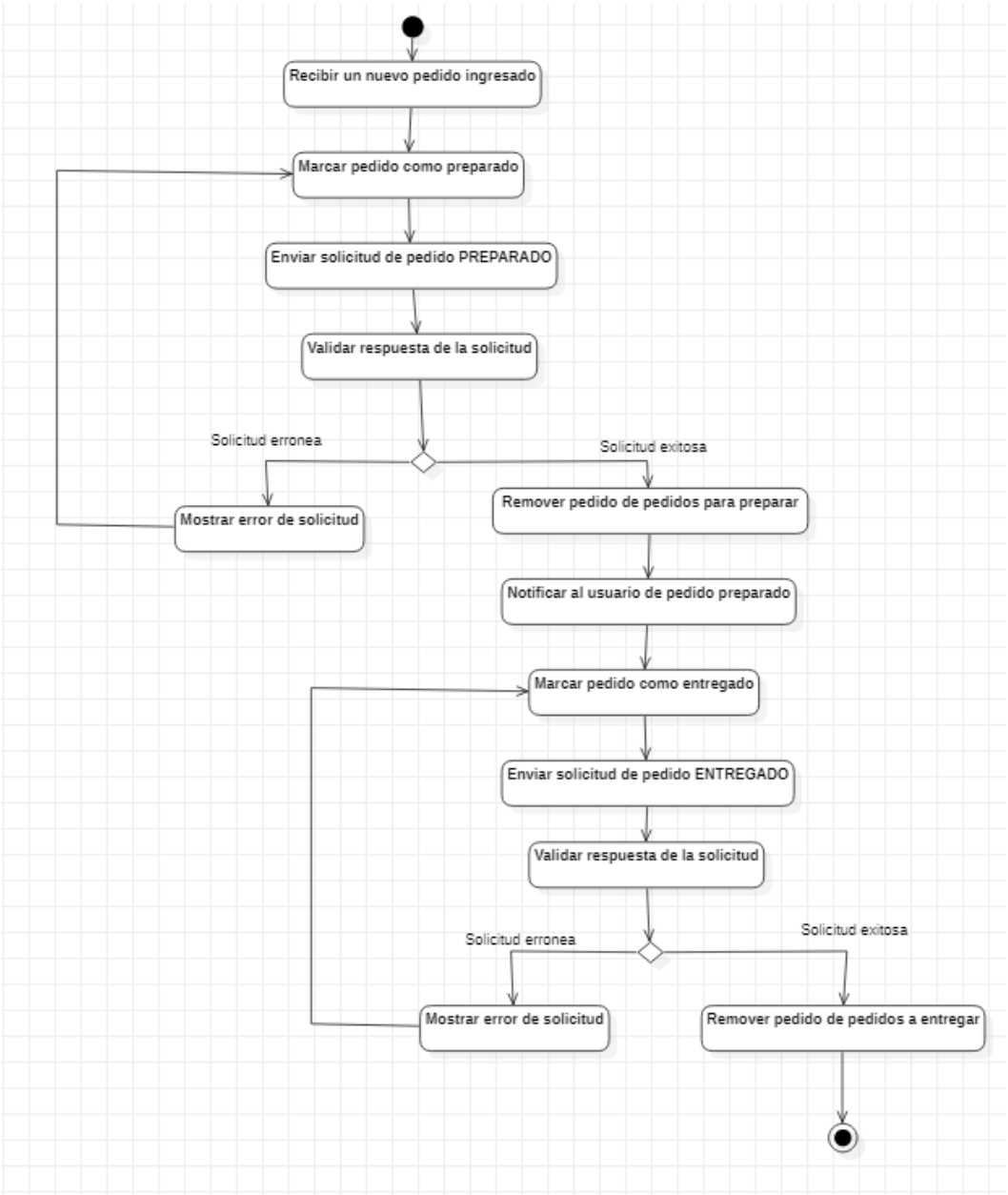


Figura 3.15: Recibo de pedido ingresado
Elaborado por: Juan Mesias

3.3.6 Diagramas de secuencia

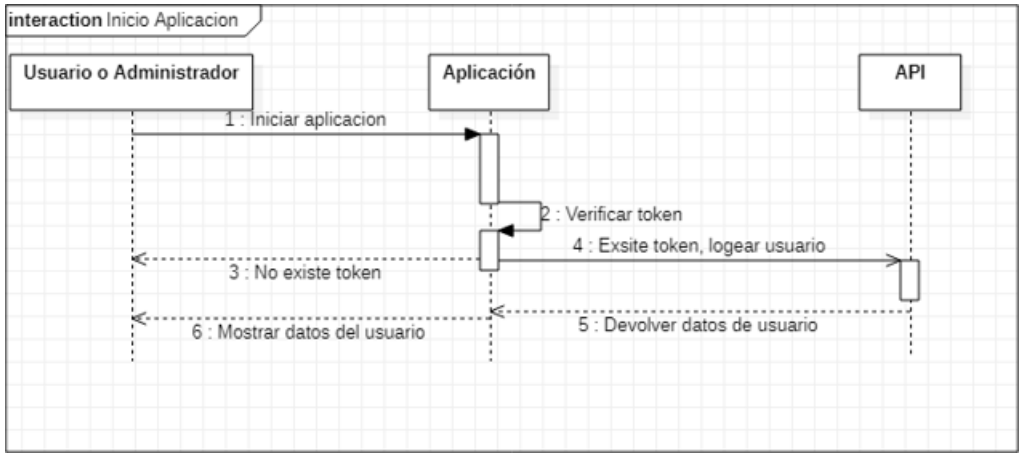


Figura 3.16: Inicio de la aplicación
Elaborado por: Juan Mesias

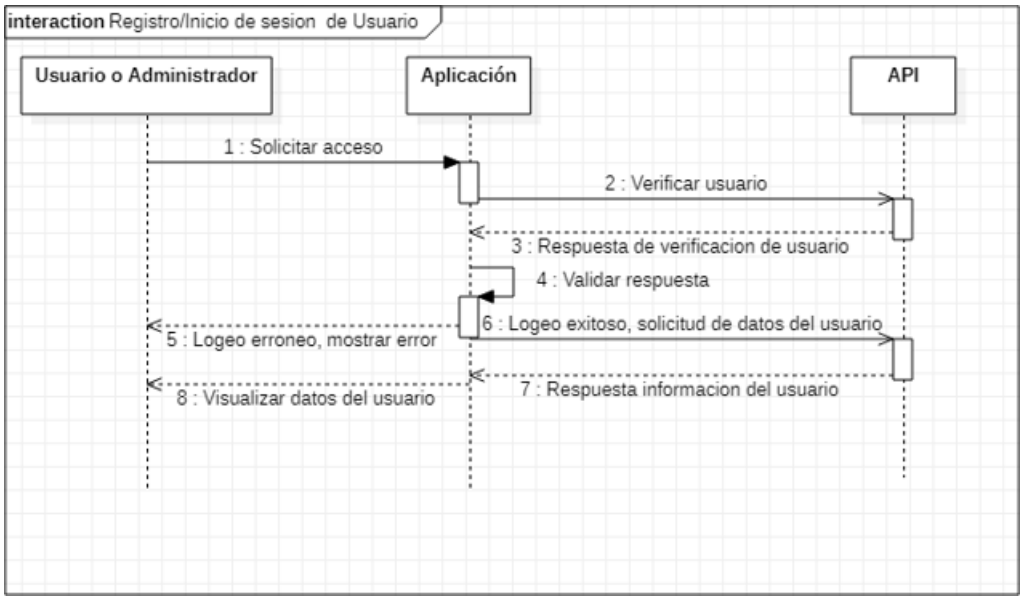


Figura 3.17: Registro/Inicio de sesión de usuario
Elaborado por: Juan Mesias

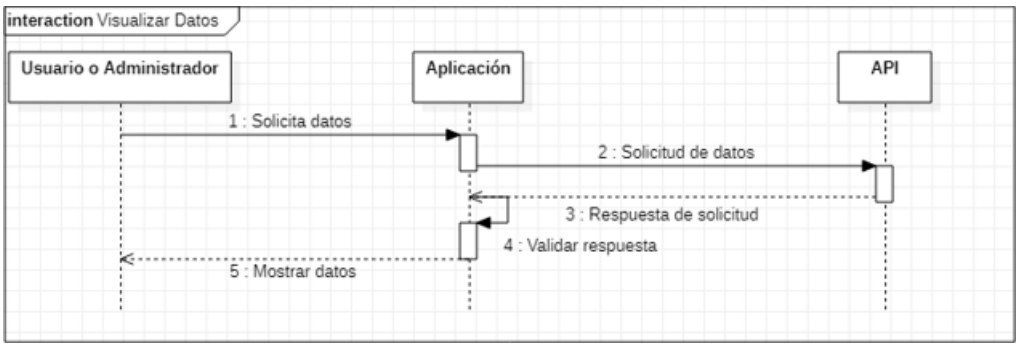


Figura 3.18: Consulta de datos
Elaborado por: Juan Mesias

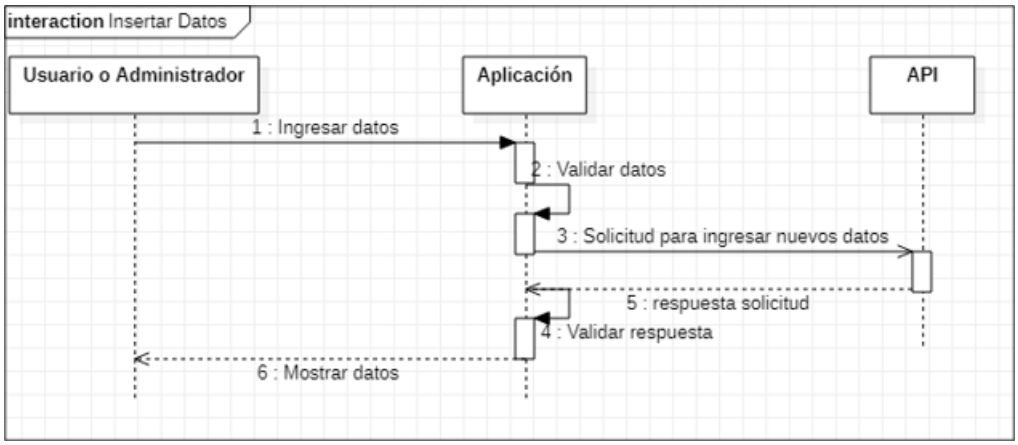


Figura 3.19: Ingreso de datos
Elaborado por: Juan Mesias

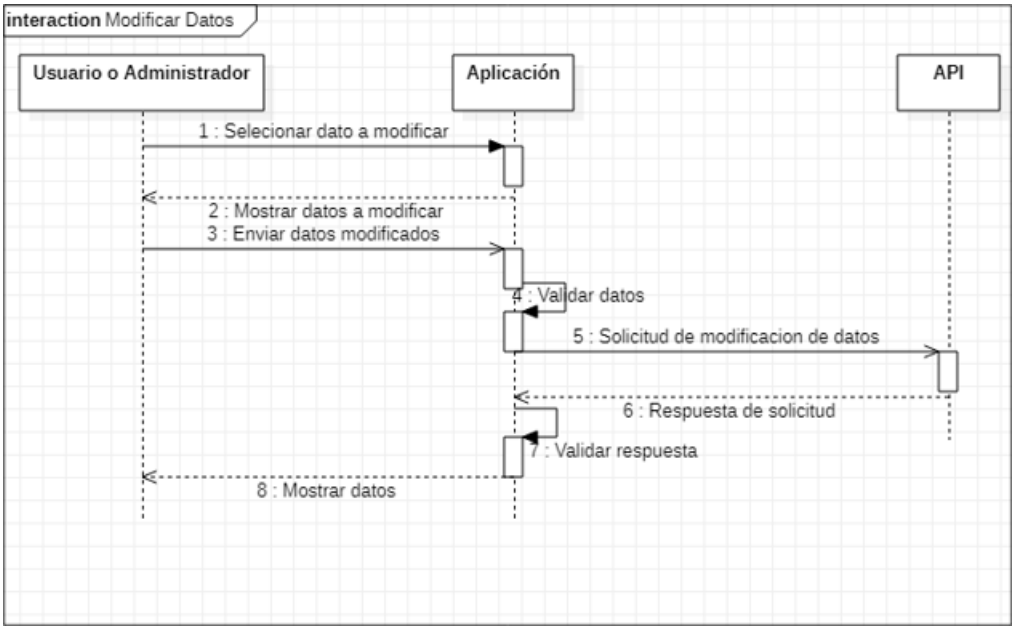


Figura 3.20: Modificación de datos
Elaborado por: Juan Mesias

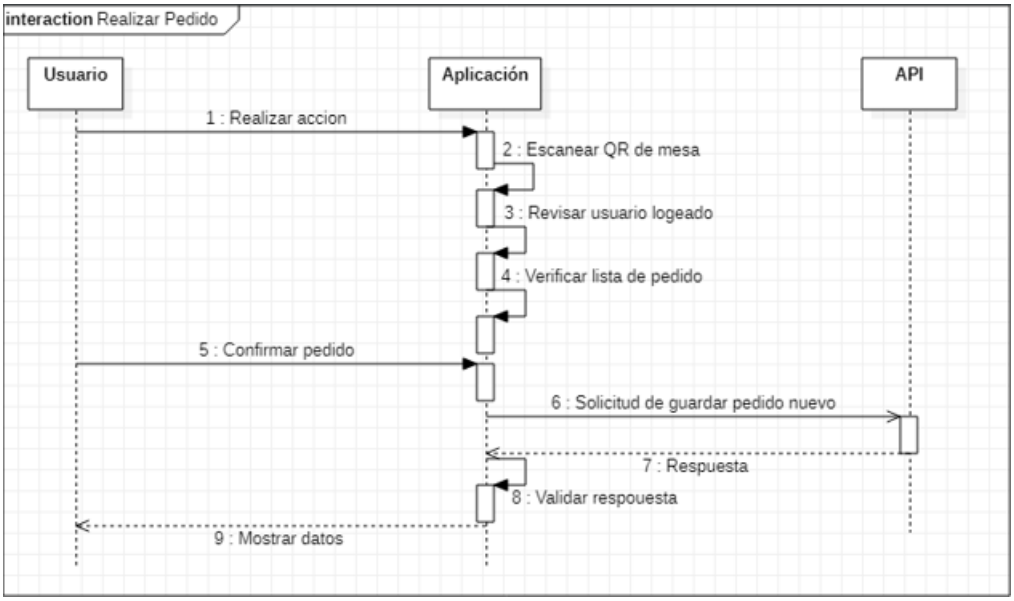


Figura 3.21: Procedimiento para hacer pedido
Elaborado por: Juan Mesias

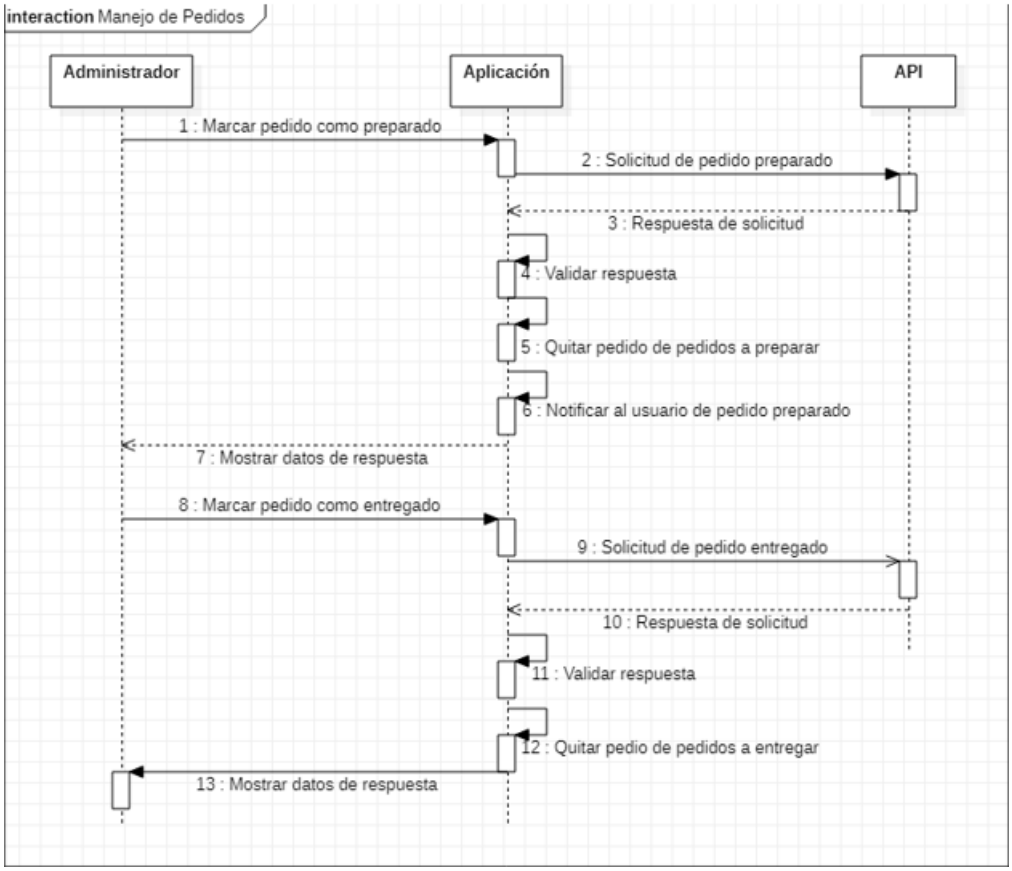


Figura 3.22: Manejo de pedidos
Elaborado por: Juan Mesias

3.3.7 Interacción con el Usuario

3.3.8 Diseño de la interfaz UI

El diseño de la interfaz de la aplicación se basa en el diseño de componentes, el diseño de cada componente debe disponer una capacidad responsiva, para adaptarse a cualquier entorno. La utilización de componentes es una característica que permite al desarrollador la reutilización de código y al mismo tiempo la reducción de tiempo de desarrollo. Para el diseño de la interfaz se puede tener en referencia varias opciones de framework que proveen ya los componentes con la disponibilidad de modificarlos y personalizarlos. La herramienta selecionada para la generacion de temas para modificar los componentes es "material-ui-themeeditor" que nos permite visualizar los temas como muestra la figura 3.23.

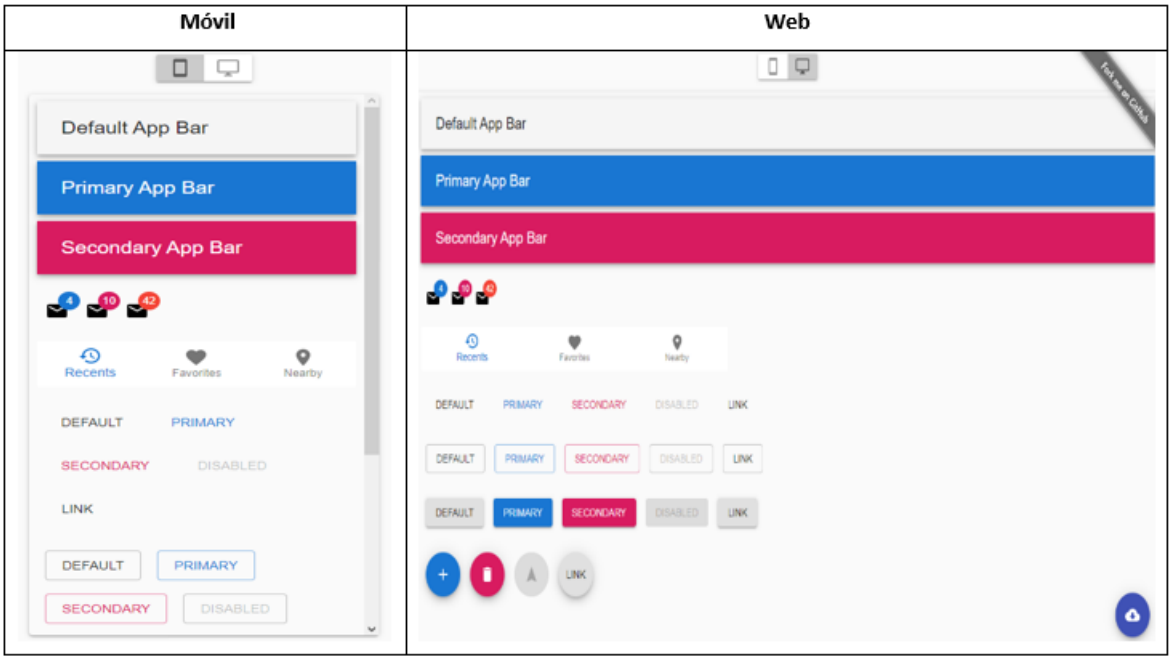


Figura 3.23: Tema
Elaborado por: Juan Mesias

3.3.9 Estilos CSS

Los estilos serán definidos por la paleta representante al tema o framework para personalizar los componentes, el tema contiene colores como los primarios y los secundarios de la aplicación. Figura 3.24

```
const theme = createMuiTheme({
  palette: {
    primary: {
      light: '#757ce8',
      main: '#3f50b5',
      dark: '#002884',
      contrastText: '#fff',
    },
    secondary: {
      light: '#ff7961',
      main: '#f44336',
      dark: '#ba000d',
      contrastText: '#000',
    },
  },
});
```

Figura 3.24: Estilos
Elaborado por: Juan Mesias

3.3.10 Usabilidad

La usabilidad de es vital importancia en la aplicación dado que permite al usuario entender y comprender la funcionalidad de la misma, logrando proveer al usuario de una herramienta comprensible para lograr las tareas específicas en el contexto previsto para su uso.

La usabilidad está estrictamente relacionada con la percepción que el usuario obtiene de la aplicación, la calidad del sistema la velocidad de sus operaciones todos estos procesos y más se ven reflejados bajo la usabilidad y la presentación de las interfaces para demostrar la eficiencia del trabajo sobre la tarea indicada.

El nivel de usabilidad en la aplicación permite al usuario guiarse intuitivamente dentro de la misma. Uno de los puntos mas importantes es el nivel de adaptavilidad de la aplicación a diferentes dispositivos y diferentes tamaños de pantallas como computadores y varios dispositivos moviles. Las herramientas utilizadas para obtener el nivel de adaptabilidad deseado fueron Material-UI, Bootstrap y la utilizacion de media queries.

3.3.11 Seguridad

La importancia de la seguridad dentro de una aplicación que maneja datos personales de usuarios es vital, dicha información es manipulable y está a disposición del usuario dependiendo de su rol el cual sería un nivel de seguridad y de acceso a los datos. Toda la información de la aplicación está concentrada el inicio de sesión o autenticación por medio de tokens de acceso implementados en la API-REST. De esta manera, únicamente los usuarios con una cuenta y con un rol específicos tendrán acceso a la información que les corresponde.

3.4 Estudio y elección de herramientas para el desarrollo del proyecto

3.4.1 Herramientas para Backend

Las herramientas backend son tecnologías que permiten conectar con el lado del servidor y trabajar con las bases de datos.

Laravel

Laravel es un marco de aplicación web con sintaxis expresiva y elegante. Donde el desarrollo debe ser una experiencia agradable y creativa para ser verdaderamente gratificante. Laravel intenta eliminar el dolor del desarrollo al facilitar las tareas comunes utilizadas en la mayoría de los proyectos web, como la autenticación, el enrutamiento, las sesiones y el almacenamiento en caché.
Laravel tiene como objetivo hacer que el proceso de desarrollo sea agradable para el desarrollador sin sacrificar la funcionalidad de la aplicación. Los desarrolladores felices hacen el mejor código. Con este fin, se ha intentado combinar lo mejor de otros marcos web, incluidos los marcos implementados en otros lenguajes, como Ruby on Rails, ASP.NET MVC y Sinatra.
Laravel es accesible, pero potente, y proporciona herramientas poderosas necesarias para aplicaciones grandes y robustas. Una excelente inversión de contenedor de control, un sistema de migración expresivo y un soporte de prueba de unidad estrechamente integrado brindan las herramientas que se necesita para crear cualquier aplicación [29].

Django

Django es un marco web Python de alto nivel que fomenta el desarrollo rápido y el diseño limpio y pragmático. Creado por desarrolladores experimentados, se ocupa de gran parte de la molestia del desarrollo web, por lo que se puede escribir una aplicación sin necesidad de reinventar la rueda. Es gratis y de código abierto.
Django fue diseñado para ayudar a los desarrolladores a llevar las aplicaciones desde el concepto hasta su finalización lo más rápido posible.
Django toma en serio la seguridad y ayuda a los desarrolladores a evitar muchos errores de seguridad comunes.
Algunos de los sitios más activos de la Web aprovechan la capacidad de Django para escalar de manera rápida y flexible [30].

ASP.NET

.NET es una plataforma de desarrollador compuesta por herramientas, lenguajes de programación y bibliotecas para construir muchos tipos diferentes de aplicaciones.
La plataforma base proporciona componentes que se aplican a todos los diferentes tipos de aplicaciones. Los marcos adicionales, como ASP.NET, extienden .NET con componentes para construir tipos específicos de aplicaciones como:
Los lenguajes de programación C#, F# y Visual Basic.
Bibliotecas base para trabajar con cadenas, fechas, archivos / IO y más.

Editores y herramientas para Windows, Linux, macOS y Docker [31].

Comparación de diferentes herramientas para Backend

De las herramientas expuestas en la sección 3.4.1 se hace la comparación para determinar cual herramienta es mejor para el Backend de la aplicación basado en los siguientes puntos.

- **Lenguaje de programación:** Es el lenguaje con el cual se debe trabajar.
- **Código abierto:** Se refiere a que si el código es abierto y puede ser modificado por cualquiera.
- **Reutilización de código:** Se refiere a la capacidad de la herramienta de reutilizar código para diferentes aplicaciones y plataformas.
- **Seguridad:** Es el nivel de seguridad que ofrece la herramienta.
- **Soporte de plataformas:** Se refiere a la capacidad de correr la aplicación en cualquier plataforma
- **Extensiones:** Se refiere a la cantidad de herramientas adicionales que se puede agregar.
- **Documentación:** Se refiere a la cantidad de información que se puede encontrar de la herramienta así como soporte técnico o blogs de ayuda de la comunidad.

	Laravel	**Django**	**ASP.NET**
Lenguaje de programación	PHP	Python	C#
Código abierto	Si	Si	No
Reutilización de código	Funciona en varias plataformas	Funciona en varias plataformas	Se debe modificar para que funcione en otras plataformas
Seguridad	Facilita las tareas comunes utilizadas en la mayoría de los proyectos web, como la autenticación	Toma en serio la seguridad y ayuda a los desarrolladores a evitar muchos errores de seguridad comunes	Se debe agregar extensiones para que la seguridad funcione correctamente
Soporte de plataformas	Si	Si	No, se debe modificar el código para que funcione en otras plataformas

Extensiones	Si	Si	Si
Documentación	Gran cantidad de documentación y ayuda de la comunidad	Gran cantidad de documentación y ayuda de la comunidad	Gran cantidad de documentación y ayuda de la comunidad

Tabla 3.3: Cuadro comparativo de herramientas para backend
Elaborado por: Juan Mesias

Del cuadro comparativo 3.4 se tomó como herramienta Laravel debido a su fácil utilización, a su nivel de seguridad e implementación. Además debido a que es gratis y de código abierto se puede modificar de ser necesario y la gran cantidad de extensiones permiten crear una aplicación completa y robusta, además existe una gran cantidad de documentación y una comunidad creciente se encarga de ayudar en caso de problemas.

3.4.2 Herramientas para el Frontend

Las herramientas frontend son tecnologías que permiten conectar con el lado del cliente es decir es lo que el usuario ve en su pantalla de celular o computador, el producto final con el cual se interactúa.

Angular

Angular es un marco de diseño de aplicaciones y una plataforma de desarrollo para crear aplicaciones eficientes y sofisticadas de una sola página. Utiliza herramientas con HTML , CSS , JavaScript y algunas de las herramientas de los últimos estándares, como clases y módulos.
La arquitectura de una aplicación angular se basa en ciertos conceptos fundamentales. Los bloques de construcción básicos son NgModules , que proporcionan un contexto de compilación para los componentes . NgModules recopila código relacionado en conjuntos funcionales; Una aplicación angular se define mediante un conjunto de NgModules. Una aplicación siempre tiene al menos un módulo raíz que permite el arranque y, por lo general, tiene muchos más módulos de características.
Los componentes definen vistas , que son conjuntos de elementos de pantalla que Angular puede elegir y modificar de acuerdo con la lógica y los datos de su programa.
Los componentes de una aplicación suelen definir muchas vistas, ordenadas jerárquica mente. Angular proporciona el Routerservicio para ayudarlo a definir rutas de navegación entre vistas. El enrutador proporciona capacidades de navegación sofisticadas en el navegador [32].

Vue

Vue es un marco progresivo para construir interfaces de usuario. A diferencia de otros marcos monolíticos, Vue está diseñado desde cero para ser gradualmente adoptable. La biblioteca principal se centra solo en la capa de vista y es fácil de recoger e integrar con otras bibliotecas o proyectos existentes. Por otro lado, Vue también es perfectamente capaz de impulsar aplicaciones sofisticadas de una sola página cuando se usa en combinación con herramientas modernas y bibliotecas de soporte [33].

React

React crea interfaces de usuario interactivas de forma sencilla. Diseña vistas simples para cada estado en la aplicación, y React se encarga de actualizar y renderizar de manera eficiente los componentes correctos cuando los datos cambien. Las vistas declarativas hacen que el código sea más predecible, por lo tanto, fácil de depurar.
React crea componentes encapsulados que manejan su propio estado, y los convierte en interfaces de usuario complejas. Ya que la lógica de los componentes está escrita en JavaScript y no en plantillas, se puede pasar datos de forma sencilla a través de la aplicación y mantener el estado fuera del DOM [16].

Comparación de diferentes herramientas para Frontend

De las herramientas expuestas en la sección 3.4.2 se hace la comparación para determinar cual herramienta es mejor para el Frontend de la aplicación basado en los siguientes puntos.

- **Lenguaje de programación:** Es el lenguaje con el cual se debe trabajar.
- **Código abierto:** Se refiere a que si el código es abierto y puede ser modificado por cualquiera.
- **Reutilización de código:** Se refiere a la capacidad de la herramienta de reutilizar código para diferentes aplicaciones y plataformas.
- **Soporte de plataformas:** Se refiere a la capacidad de correr la aplicación en cualquier plataforma
- **Extensiones:** Se refiere a la cantidad de herramientas adicionales que se puede agregar.
- **Curva de aprendizaje:** Se refiere al nivel de dificultad que se presenta al usar esta herramienta.
- **Documentación:** Se refiere a la cantidad de información que se puede encontrar de la herramienta así como soporte técnico o blogs de ayuda de la comunidad.

	Angular	**Vue**	**React**
Lenguaje de programación	Typescript	Javascript	Javascript
Código abierto	Si	Si	Si
Reutilización de código	Funciona en varias plataformas	Funciona en varias plataformas	Funciona en varias plataformas
Soporte de plataformas	Si	Si	Si
Extensiones	Si	Si	Si
Curva de aprendizaje	Se necesita conocimientos previos de javascript	Se necesita conocimientos previos de typescript	La herramienta es muy intuitiva y no se necesita muchos conocimientos previos de javascript
Documentación	Gran cantidad de documentación y ayuda de la comunidad	Gran cantidad de documentación y ayuda de la comunidad	Gran cantidad de documentación y ayuda de la comunidad

Tabla 3.4: Cuadro comparativo de herramientas para frontend
Elaborado por: Juan Mesias

Del cuadro comparativo 3.4.2 se ha escogido como herramienta para el desarrollo frontend a React debido a que 1) su curva de aprendizaje es menor, 2) existe una gran cantidad de extensiones o complementos que ayudan a mejorar la aplicación, haciéndola más robusta y completa, y 3) existe una gran cantidad de documentación para el desarrollador y ayuda de parte de la comunidad de desarrolladores.

3.5 Desarrollo de la aplicación

3.5.1 Desarrollo de casos de uso

En las figuras 5.25 y 5.26 se detallan los casos de uso identificados para el correcto funcionamiento de la aplicación, para tener los procesos bien definidos.

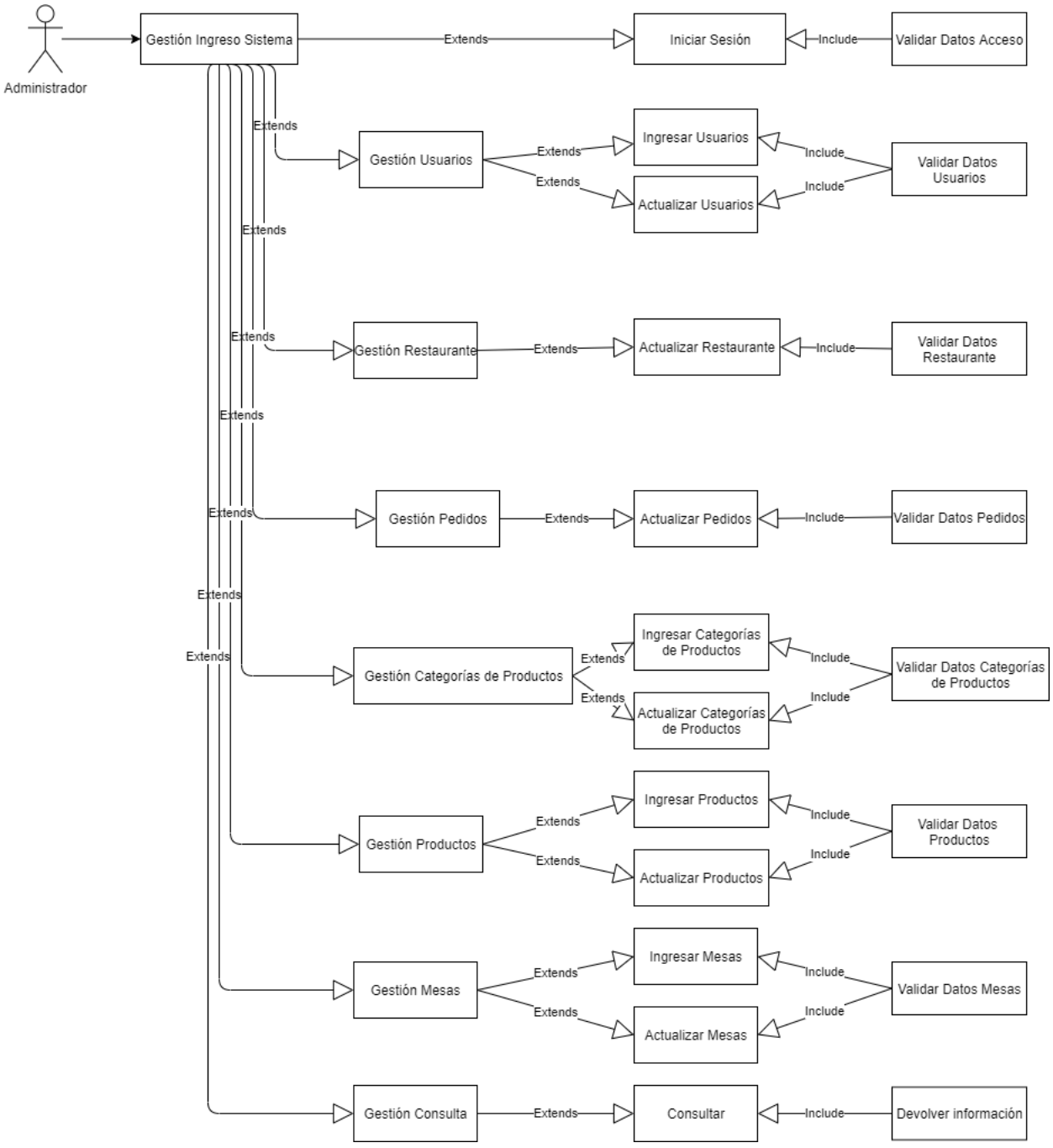


Figura 5.25: Gestión de Administrador
Elaborado por: Juan Mesias

Caso de Uso:	Iniciar sesión
Resumen:	Permite ingresar al usuario al sistema bajo un perfil determinado.
Actor:	Usuario – Administrador
Precondiciones:	El usuario debe estar registrado en el sistema.
Flujo normal:	• El actor solicita acceso a la aplicación. • La aplicación verifica datos del usuario. • La aplicación concede acceso a información personal.
Flujo alternativo:	• El usuario no está registrado. • La aplicación muestra opciones de inicio de sesión y registro. • La sesión esta previamente guardada.
Postcondiciones:	El usuario este logeado y con acceso a características especiales de la aplicación.

Tabla 3.5: Caso de uso No. 01
Elaborado por: Juan Mesias

Caso de Uso:	Gestión de Usuarios
Resumen:	Permite al usuario administrar todos los usuarios registrados.
Actor:	Usuario – Administrador
Precondiciones:	El usuario debe iniciar sesión.
Flujo normal:	• El actor crea o modifica un nuevo usuario con el perfil determinado. • La aplicación crea la solicitud hacia la API-REST para registrar o modificar el usuario. • El sistema muestra un mensaje de éxito o error según el caso. • El sistema muestra la información modificada e ingresada.
Flujo alternativo:	• El usuario no ha iniciado sesión. • Datos ingresados incorrectos.
Postcondiciones:	Usuario creado o modificado listo para ser usado.

Tabla 3.6: Caso de uso No. 02
Elaborado por: Juan Mesias

Caso de Uso:	Gestión de Restaurante
Resumen:	Permite al usuario administrar la información del restaurante.
Actor:	Usuario – Administrador
Precondiciones:	El usuario debe iniciar sesión.
Flujo normal:	• El actor modifica la información presentada del restaurante. • La aplicación envía la solicitud a la API-REST de actualización de datos del restaurante. • La aplicación muestra un mensaje de éxito o error según el caso.
Flujo alternativo:	• Datos ingresados incorrectos. • Se muestra mensaje de error.
Postcondiciones:	La información del restaurante estará actualizada.

Tabla 3.7: Caso de uso No. 03
Elaborado por: Juan Mesias

Caso de Uso:	Gestión de Pedidos
Resumen:	Permite al usuario administrar el estado del pedido realizado por un cliente.
Actor:	Usuario – Administrador
Precondiciones:	El usuario debe iniciar sesión.
Flujo normal:	• El actor modifica el estado del pedido realizado por un usuario. • La aplicación envía la solicitud a la API-REST de actualización de estado de pedido. • La aplicación muestra un mensaje de éxito o error según el caso. • El pedido se remueve de la lista en la que se encuentra según su estado.
Flujo alternativo:	• Estado del pedido marcado como cancelado.
Postcondiciones:	Estado del pedido actualizado.

Tabla 3.8: Caso de uso No. 04
Elaborado por: Juan Mesias

Caso de Uso:	Gestión de Categorías de Productos
Resumen:	Permite al usuario administrar el listado de categorías que maneja la aplicación.
Actor:	Usuario – Administrador
Precondiciones:	El usuario debe iniciar sesión.
Flujo normal:	• El actor modifica o ingresa una nueva categoría de producto. • La aplicación envía la solicitud a la API-REST de actualización o de registro. • La aplicación muestra un mensaje de éxito o error según el caso. • El listado de categorías de productos se actualiza.
Flujo alternativo:	• Datos ingresados incorrectos. • Se muestra mensaje de error.
Postcondiciones:	Categorías registradas y listas para ser consultadas.

Tabla 3.9: Caso de uso No. 05
Elaborado por: Juan Mesias

Caso de Uso:	Gestión de Productos
Resumen:	Permite al usuario administrar el listado de productos que maneja la aplicación.
Actor:	Usuario – Administrador
Precondiciones:	El usuario debe iniciar sesión.
Flujo normal:	• El actor modifica o ingresa un nuevo producto. • La aplicación envía la solicitud a la API-REST de actualización o de registro. • La aplicación muestra un mensaje de éxito o error según el caso. • El listado de productos se actualiza.
Flujo alternativo:	• Datos ingresados incorrectos. • Se muestra mensaje de error.
Postcondiciones:	Productos registrados y listos para ser consultados.

Tabla 3.10: Caso de uso No. 06
Elaborado por: Juan Mesias

Caso de Uso:	Gestión de Mesas
Resumen:	Permite al usuario administrar el listado de mesas que maneja la aplicación.
Actor:	Usuario – Administrador
Precondiciones:	El usuario debe iniciar sesión.
Flujo normal:	• El actor modifica o ingresa una nueva mesa. • La aplicación envía la solicitud a la API-REST de actualización o de registro. • La aplicación muestra un mensaje de éxito o error según el caso. • El listado de mesas se actualiza.
Flujo alternativo:	• Datos ingresados incorrectos. • Se muestra mensaje de error.
Postcondiciones:	Mesas registradas y listas para ser consultada e imprimir los códigos QR de cada una.

Tabla 3.11: Caso de uso No. 07
Elaborado por: Juan Mesias

Caso de Uso:	Gestión de Consultas
Resumen:	Permite al usuario consultar información específica de cualquier tipo de dato en la aplicación.
Actor:	Usuario – Administrador
Precondiciones:	El usuario debe iniciar sesión.
Flujo normal:	• El actor realiza la consulta • La aplicación envía la solicitud de consulta a la API-REST. • La aplicación muestra un mensaje de éxito o error según el caso. • Se muestra la información.
Flujo alternativo:	
Postcondiciones:	La información requerida se muestra en pantalla.

Tabla 3.12: Caso de uso No. 08
Elaborado por: Juan Mesias

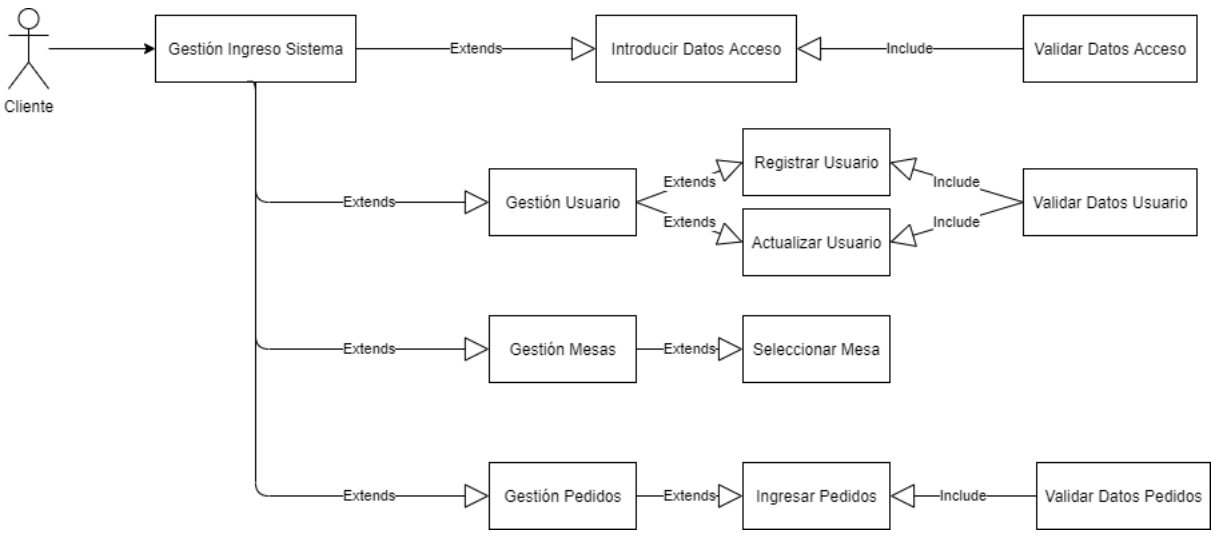


Figura 5.26: Gestión del Cliente
Elaborado por: Juan Mesias

Caso de Uso:	Gestión de Usuario
Resumen:	Permite al usuario cliente administrar su información personal en su perfil.
Actor:	Usuario – Cliente
Precondiciones:	El usuario debe iniciar sesión.
Flujo normal:	• El actor modifica su información personal. • La aplicación crea la solicitud hacia la API-REST para modificar el usuario. • El sistema muestra un mensaje de éxito o error según el caso. • El sistema muestra la información modificada e ingresada.
Flujo alternativo:	• El usuario no ha iniciado sesión. • Datos ingresados incorrectos.
Postcondiciones:	Usuario modificado listo para ser usado.

Tabla 3.13: Caso de uso No. 09
Elaborado por: Juan Mesias

Caso de Uso:	Gestión de Mesas
Resumen:	Permite al usuario seleccionar una mesa a la cual será servido su pedido.
Actor:	Usuario – Cliente
Precondiciones:	El usuario debe iniciar sesión.
Flujo normal:	• El actor escanea un código de mesa. • La aplicación consulta la mesa. • El sistema muestra la mesa seleccionada.
Flujo alternativo:	• El usuario elije realizar su pedido por ubicación.
Postcondiciones:	Puede proceder a confirmar su pedido o a realizar el mismo.

Tabla 3.14: Caso de uso No. 10
Elaborado por: Juan Mesias

Caso de Uso:	Gestión de Pedidos
Resumen:	Permite al usuario registrar un nuevo pedido.
Actor:	Usuario – Cliente
Precondiciones:	El usuario debe iniciar sesión.
Flujo normal:	• El actor llena su lista de pedidos. • El actor confirma su pedido. • La aplicación envía la solicitud hacia la API-REST de registro de nuevo pedido. • Se muestra mensaje de éxito o error según el caso.
Flujo alternativo:	• El usuario no termina de realizar su pedido.
Postcondiciones:	Puede revisar el estado y detalles de su pedido.

Tabla 3.15: Caso de uso No. 11
Elaborado por: Juan Mesias

Caso de Uso:	Gestión de Datos de Facturación
Resumen:	Permite al usuario registrar nuevos datos de facturación.
Actor:	Usuario – Cliente
Precondiciones:	El usuario debe iniciar sesión.
Flujo normal:	• El usuario llena sus nuevos datos de facturación. • La aplicación envía una solicitud hacia la API-REST para el registro de la nueva aplicación. • Se muestra mensaje de éxito o error según el caso.
Flujo alternativo:	• El usuario puede agregar datos de facturación al confirmar su pedido.
Postcondiciones:	Puede revisar y consultar sus datos de facturación.

Tabla 3.16: Caso de uso No. 12
Elaborado por: Juan Mesias

3.5.2 Diagrama de clases

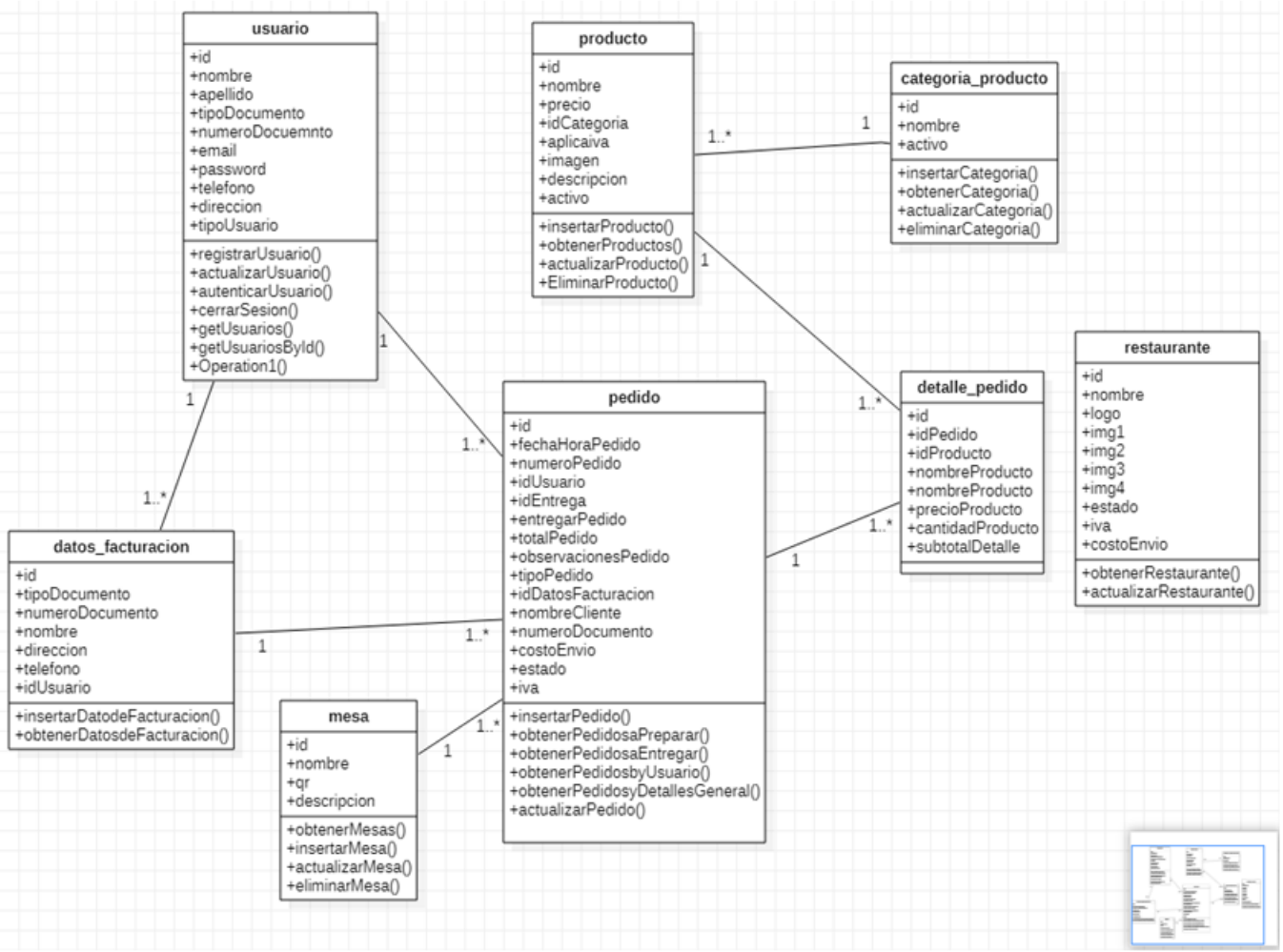


Figura 5.27: Diagrama de clases
Elaborado por: Juan Mesias

3.5.3 Desarrollo del software

Detalles de funcionalidad

La comunicación entre el frontend y el backend de la aplicación se basa en la arquitectura de comunicación mediante una API-REST. De esta manera garantiza en funcionamiento de la aplicación en cualquier dispositivo. El aplicativo garantiza que el acceso a la información de cada usuario y de configuración del restaurante esté protegido mediante el inicio de sesión por tokens. Una vez realizado el inicio de sesión por tokens se debe considerar el perfil de cada usuario para mostrar la información correspondiente. Cada usuario poseerá un rol el cual le dará funciones específicas dentro de la aplicación para el manejo de datos. La biblioteca para diseñar las interfaces de la aplicación es Reactjs. A través de esta fue posible manejar los datos que se reciben a través de solicitudes HTTP de manera óptima.

Axios

Es denominado un cliente HTTP para el manejo de solicitudes hacia una API-REST. Figura 5.28.

```
1        import axios from "axios";
2
3        const clienteAxios = axios.create({
4                baseURL: "http://192.168.1.11:8000/api",
5        });
6
7        export default clienteAxios;
```

Figura 5.28: Axios
Elaborado por: Juan Mesías

REDUX

La implementación de REDUX dentro de una aplicación diseñada con Reactjs es altamente eficiente, REDUX se basa en el manejo de un store interno por cada aplicación, que permite la inserción modificación y eliminación de registros a un nivel interno, el store puede ser modificado luego de obtener el resultado de una solicitud HTTP realizada a la API-REST a través de un cliente HTTP(axios) en caso de utilizarlo, de esta manera el store interno de la aplicación tendrá los mismos datos después de ser modificados por la API-REST. El store de REDUX se ve implementado de esta manera en una clase llamada store. El store de la aplicación se representa en la figura 5.29.

```
import { createStore, applyMiddleware, compose } from "redux";
import thunk from "redux-thunk";
import reducer from "./reducers";
const store = createStore(
  reducer,compose(applyMiddleware(thunk)));
export default store;
```

Figura 5.29: Redux
Elaborado por: Juan Mesias

Internamente al store se le asocia uno o varios REDUCERS para el manejo de datos. Figura 5.30.

```
export default combineReducers ({
restaurante: restauranteReducer,
    categorias: categoriasReducer,
productos: productosReducer,
    mesas: mesasReducer,
    pedidos: pedidosReducer,
    usuario: usuarioReducer,
logeo: logeoReducer,
});
```

Figura 5.30: Redux-Reducers
Elaborado por: Juan Mesias

El store solamente puede ser modificado por REDUCERS, que son acciones que trabajan internamente en el store de la aplicación, los responsables de disparar los REDUCERS para la modificación del store son denominados ACTIONS.

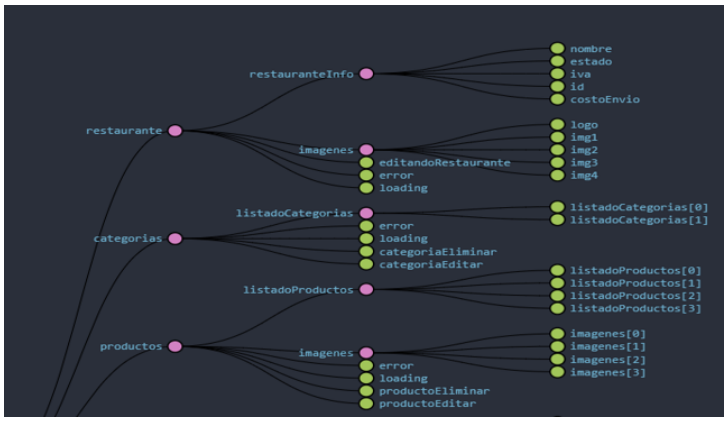


Figura 5.31: State de Redux
Elaborado por: Juan Mesias

Los ACTIONS son funciones que pueden conectar con la API-REST a través de un cliente HTTP y dependiendo o no de la respuesta de la solicitud realizada a la API-REST se dispara el REDUCER para modificar el store.

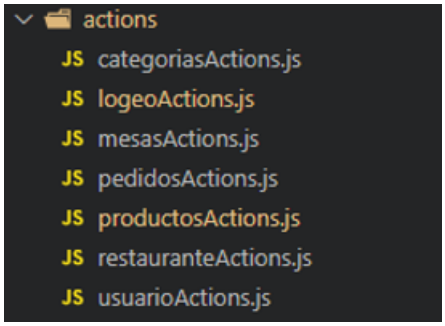


Figura 5.32: Actions
Elaborado por: Juan Mesias

```
1         export function obtenerCategoriasAction() {
2                 return async (dispatch) => {
3         dispatch(comienzaDescargaCategorias());
4
5         try {
6         const respuesta = await clienteAxios.get("/categorias");
7         dispatch(descargaCategoriasExito(respuesta.data.categorias))
              ;
8         } catch (error) {
9                 dispatch(descargaCategoriasError());
10        }
11  };
12}
```

Figura 5.33: Definición de Actions
Elaborado por: Juan Mesias

La función denominada obtener categorías actions se encarga de consultar a la API-REST todas las categorías disponibles en la base de datos, la misma que responde con un array de categorías el cual será enviado como payload y almacenado en el store de la aplicación a través de un reducer. El dispatch indicado en la función es el encargado de accionar las funciones que indicaran la acción representada por un TYPE y los parámetros representados por PAYLOAD que modificara el store en un reducer.

```
const comienzaDescargaCategorias = () => ({
  type: COMENZAR_DESCARGA_CATEGORIAS,
  payload: true,
});
const descargaCategoriasExito = (categorias) => ({
  type: DESCARGAR_CATEGORIAS_EXITO,
  payload: categorias,
});
const descargaCategoriasError = () => ({
  type: DESCARGAR_CATEGORIAS_ERROR,
  payload: true,
});
```

Figura 5.34: Actions de categorías
Elaborado por: Juan Mesias

El reducer trabaja en función a los TYPES ejecutados, los types son variables de dos o más palabras que describen la acción realizada en el reducer, son palabras para llevar un registro de las actividades realizadas dentro del reducer. También el reducer recibe como parámetro el PAYLOAD que viene a ser el paso de parámetros hacia la función. Los types son los encargados de identificar la función que se ejecutara dentro del reducer.

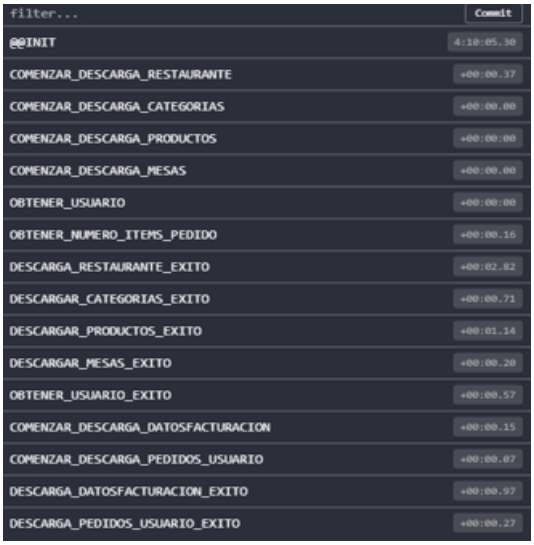


Figura 5.35: Types-Redux
Elaborado por: Juan Mesias

La declaración de los TYPES en la aplicación son como en la figura 5.36.

```
export const COMENZAR_DESCARGA_CATEGORIAS = "
    COMENZAR_DESCARGA_CATEGORIAS";
export const DESCARGAR_CATEGORIAS_EXITO = "
    DESCARGAR_CATEGORIAS_EXITO";
export const DESCARGAR_CATEGORIAS_ERROR = "
    DESCARGAR_CATEGORIAS_ERROR";
```

Figura 5.36: Declaración de Types
Elaborado por: Juan Mesias

La definición de los reducers varía según el criterio del programador, en el proyecto los reducers creados fueron de acuerdo a la figura 5.37

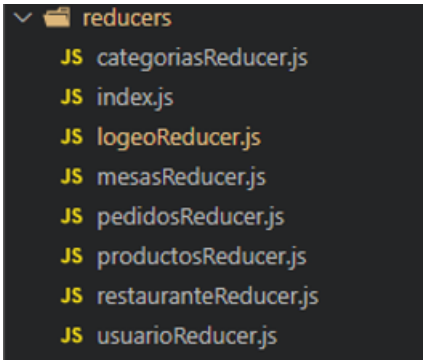


Figura 5.37: Reducers de la Aplicación
Elaborado por: Juan Mesias

Cada reducer debe tener definido su propio estado inicial como en la figura 5.38.

```
const initialState = {
listadoCategorias: [],
error: null,
loading: false,
categoriaEliminar: null,
categoriaEditar: null,
};
```

Figura 5.38: Definición del estado inicial de un Reducer
Elaborado por: Juan Mesias

El reducer tiene la estructura de un switch el cual evalua el TYPE enviado por el ACTION como indica la figura 5.39.

```
export default function (state = initialState, action) {
switch (action.type) {
case COMENZAR_DESCARGA_CATEGORIAS:
      return {
        ...state,
        loading: action.payload,
      };
    case DESCARGAR_CATEGORIAS_ERROR:
  return {
        ...state,
        loading: false,
        error: action.payload,
  };
    case DESCARGAR_CATEGORIAS_EXITO:
  return {
        ...state,
        loading: false,
    error: null,
        listadoCategorias: action.payload,
  };

    default:
  return state;
  }
}
```

Figura 5.39: Cuerpo de un Reducer
Elaborado por: Juan Mesias

La aplicación consigue adaptarse a múltiples tamaños de pantallas definidos como Extra Small(xs), Small(sm), Medium(md) y Large(lg) contenidos por una tabla que modifica el comportamiento de cada componente detallado en la figura 5.40.

```
<Grid item xs={12} md={4} lg={3}>
    //..
  </Grid>
  <Grid item xs={12} md={8} lg={6}>
    //..
  </Grid>
```

Figura 5.40: Código de la pantalla principal de la aplicación
Elaborado por: Juan Mesias

El código para cargar los datos iniciales de la aplicacion se realiza mendiante la implementacion de ACTIONS y de los invoca de la manera mostrada en la figura 5.41

```
1         useEffect(() => {
2
3             const cargarRestaurante = () => dispatch(
                  obtenerRestauranteAction());
4             cargarRestaurante();
5             const cargarCategorias = () => dispatch(
                  obtenerCategoriasAction());
6         cargarCategorias();
7             const cargarProductos = () => dispatch(
                  obtenerProductosAction());
8             cargarProductos();
9             const cargarMesas = () => dispatch(obtenerMesasAction())
                  ;
10            cargarMesas();
11
12            dispatch(getUsuarioAutenticado());
13
14  }, []);
```

Figura 5.41: Código para cargar la información inicial de la aplicación
Elaborado por: Juan Mesias

Para diferenciar los roles y mostrar el menú correspondiente al usuario se valida al mismo con operadores ternarios partiendo de su propiedad tipo de usuario.

```
1         {
2         tipousuario === 'usuario'?
3         ( <MenuUsuario handleCerrarMenu={handleCerrarMenu}/> )
4         :
5         ( <MenuCliente handleCerrarMenu={handleCerrarMenu}/> )
6         }
```

Figura 5.42: Código de roles del usuario
Elaborado por: Juan Mesias

Se propuso dos opciones para la definición del lugar de entrega de el pedido por medio de mapas elijiendo un lugar de entrega y por medio de el escaneo de un código QR perteneciente a una mesa dentro del restaurante. La implementación de la herramienta para leer códigos Qr se realizo a travéz del componente react-qr-reader.

```
                <QrReader
            ref={componentRef}
            delay={300}
            onError={handleError}
            onScan={handleScan}
            style={{ width: "100%" }}
            showViewFinder={false}
            /* className={classes.paper} */
          />
```

Figura 5.43: Código de QR
Elaborado por: Juan Mesias

Una de las partes fundamentales de la aplicación son los módulos Pedidos a preparar y Pedidos a entregar que trabajan bajo el servicio de WebSockets, lo cual permiten a los módulos trabajar en tiempo real al recibir y enviar información, este servicio fue implementado con anterioridad el Laravel, para acceder a esta conexión y poder transmitir información se debe realizar la configuración en el cliente que en este caso es Reactjs como muestra la figura 5.44.

```
        window.Pusher = Pusher;
        export default window.Echo = new Echo({
        broadcaster: "pusher",
        key: "ASDASD2121",
        wsHost: "192.168.1.11",
        wsPort: 6001,
        disableStats: true,
        //encrypted: true
        });
```

Figura 5.44: Configuración de Websockets
Elaborado por: Juan Mesias

Para poder escuchar la emisión de acciones por parte de la configuración realizada, se declaró una clase denominada listener, que es específicamente encargada de ejecutar acciones dependiendo del evento emitido por parte de Laravel.

```
        echo.channel("pedidos").listen("NuevoPedido", (ev) =>
      agregarPedidoDetalleLive(ev.pedido, ev.detalles)
    );

        echo.channel("pedidoPreparado").listen("PedidoPreparado", (
            ev) => {
        agregarPedidoAEntregar(ev.pedido, ev.detalles);
        });

        echo.channel("pedidoCancelado").listen("PedidoCancelado", (
            ev) =>
      borrarPedidoCancelado(ev.pedido, ev.detalles)
        );

        echo.channel("pedidoEntregado").listen("PedidoEntregado", (
            ev) =>
      borrarPedidoEntregado(ev.pedido, ev.detalles)
    );
```

Figura 5.45: Listeners de Websocket
Elaborado por: Juan Mesias

La aplicación permite la impresion de dos elementos que son el QR generado para la identificación de la mesa y la factura generada a partir de un nuevo pedido, la implementación de la impresión se realizo con el componenete "react to print" como se mueysra en la figura 5.46

```
<ReactToPrint
          trigger={() => (
            <Button variant="contained" color="primary" fullWidth>
              <Typography variant="h5"> Imprimir </Typography>
            </Button>
          )}
          content={() => componentRef.current}
        />
```

Figura 5.46: Código de cerrar sesión
Elaborado por: Juan Mesias

La opción de cerrar sesión esta definida para cualquier tipo de usuario dentro de la aplicación ya que permite al usuario terminar la sesión y borrar sus datos de referencia en el store interno de la aplicación.
Action realizada para cerrar la sesión y modificar el reducer.

```
export function cerrarSesionAction() {
        return async (dispatch) => {
        dispatch(cerrarSesion());
        };
}
```

Figura 5.47: Código de cerrar sesión
Elaborado por: Juan Mesias

Caso en el reducer que modifica el store y el LocalStorage de la aplicación para eliminar el token.

```
  case CERRAR_SESION:
localStorage.removeItem("*********");
return {
  ...state,
  token: null,
  autenticado: null,
  usuarioInfo: null,
  mensaje: null,
  datosFacturacion: [],
  pedidos: [],
  detalles: [],
};
```

Figura 5.48: Código para eliminar token
Elaborado por: Juan Mesias

3.6 Pruebas de funcionalidad

3.6.1 Evaluación de usabilidad del prototipo terminado

Ingreso a la aplicación

Al iniciar la aplicación, independientemente del inicio de sesión, se mostrará la información esencial figura 6.49 para comenzar con el proceso de realizar un pedido. La información que muestra la aplicación son las opciones para empezar con la generación de una nueva orden y la lista de categorías y productos que están disponibles para su venta.

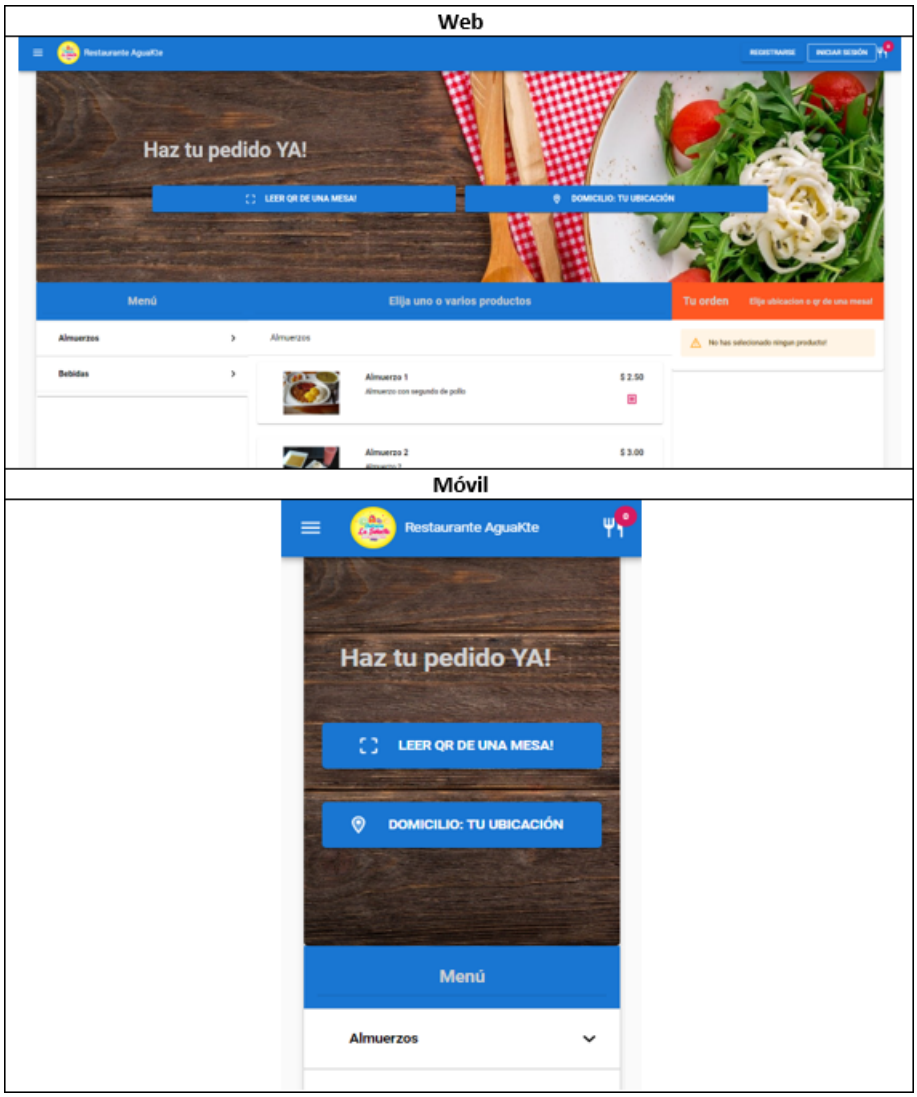


Figura 6.49: Pantalla principal de la aplicación
Elaborado por: Juan Mesias

Inicio de Sesión

Una vez iniciada la aplicación y mostrada la información relevante, el usuario tiene la opción de registrarse o simplemente iniciar sesión. En caso de que exista un token de una sesión previa el aplicativo tratara de iniciar sesión automáticamente.Figura 6.50.

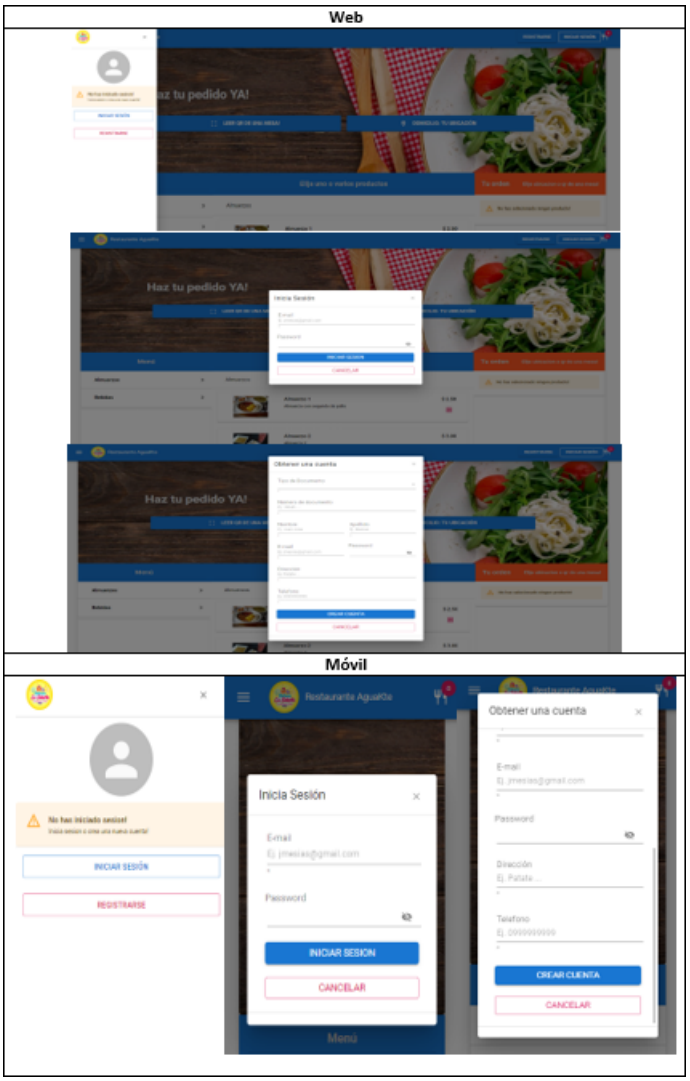


Figura 6.50: Pantalla de inicio de sesión
Elaborado por: Juan Mesias

Una vez que el usuario haya iniciado sesión se modificará el encabezado y el menú lateral de la aplicación, mostrando información y submenús que le permitirán navegar dentro de la aplicación. Como indica la figura 6.51.

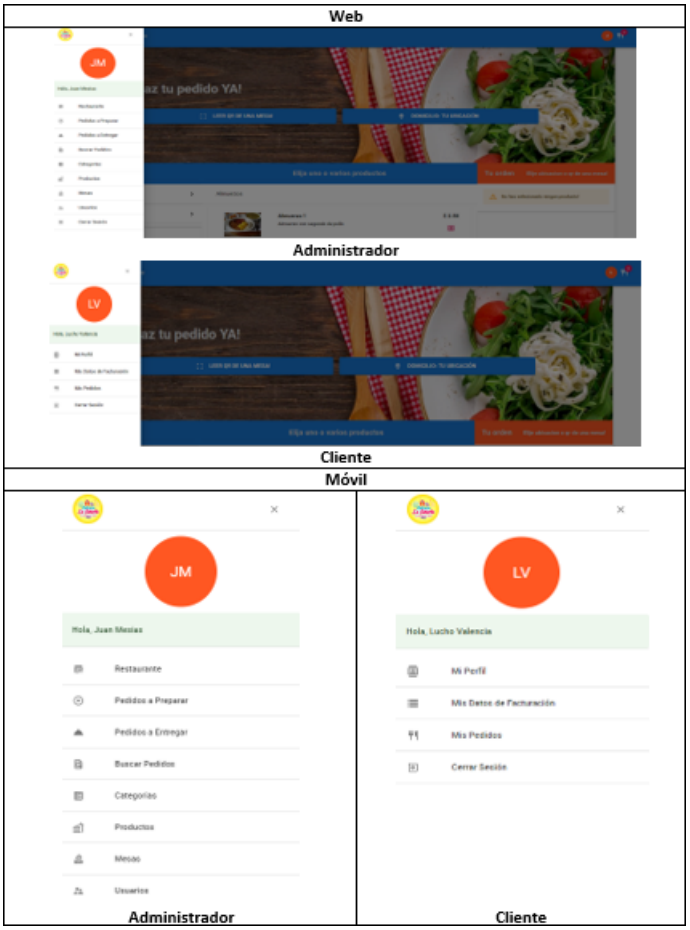


Figura 6.51: Pantalla de inicio del usuario
Elaborado por: Juan Mesias

Realizar un Pedido

El usuario registrado como cliente podrá hacer uso de la funcionalidad de la aplicación para realizar pedidos mediante la misma, de tal manera que escaneando el código QR perteneciente a una mesa dentro del restaurante definirá el lugar de entrega de su pedido. La pantalla para escanear el código QR se indica en la figura 6.52

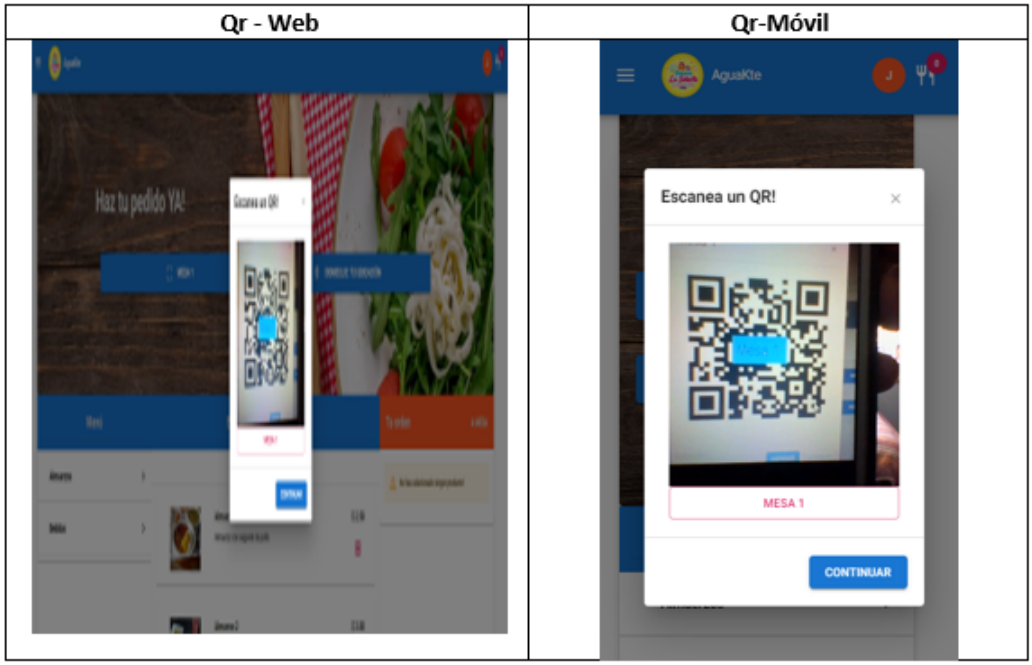


Figura 6.52: Pantalla lectura de QR
Elaborado por: Juan Mesias

Agregar pedido

Una vez elegido el método de entrega se procede a agregar productos a su pedido de acuerdo a las figuras 6.53, 6.54, 6.55, 6.56 en ese orden.

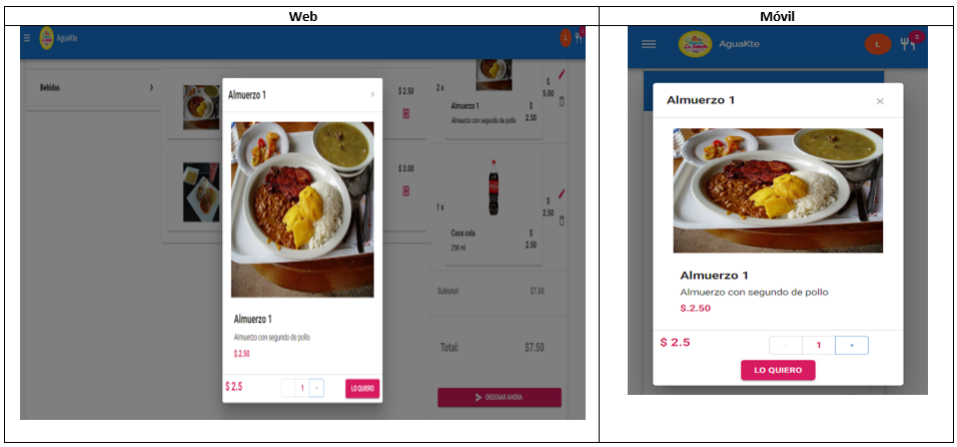


Figura 6.53: Agregar productos
Elaborado por: Juan Mesias

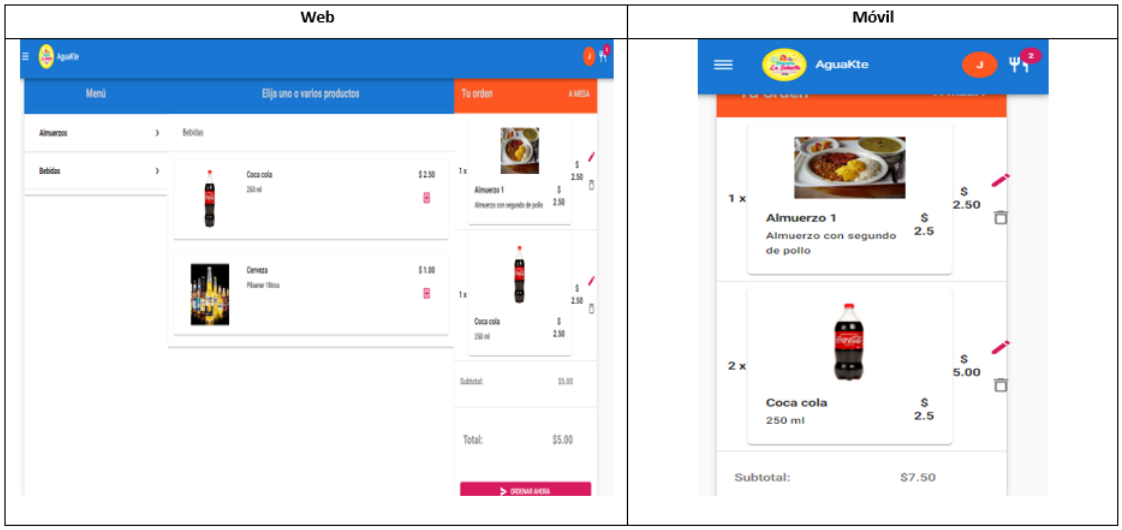


Figura 6.54: Confirmación de pedidos
Elaborado por: Juan Mesias

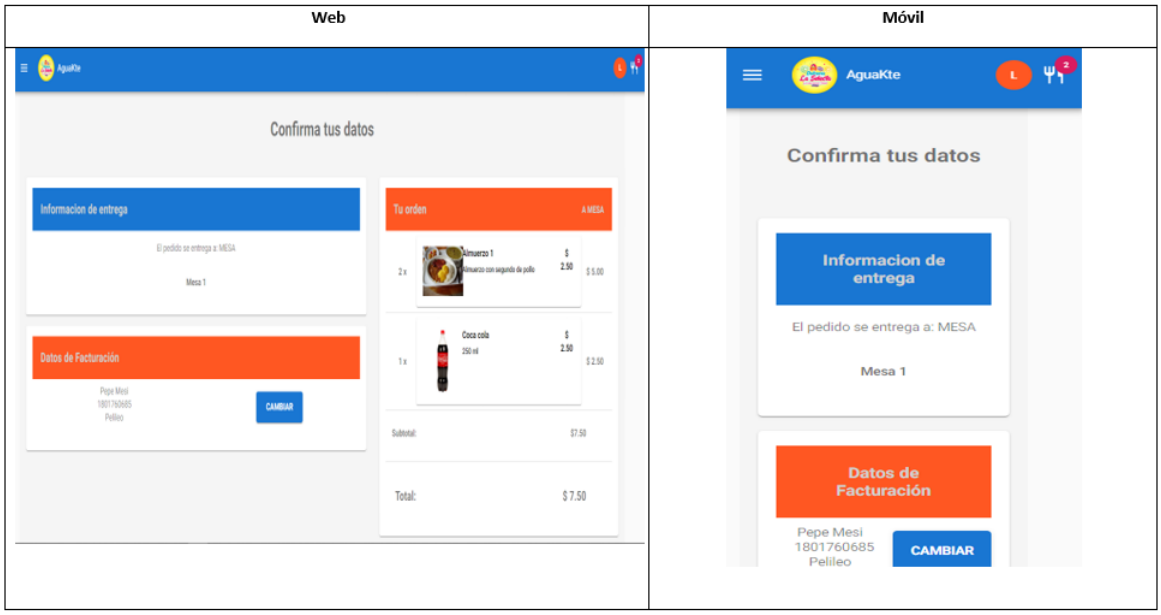


Figura 6.55: Ordenar pedido
Elaborado por: Juan Mesias

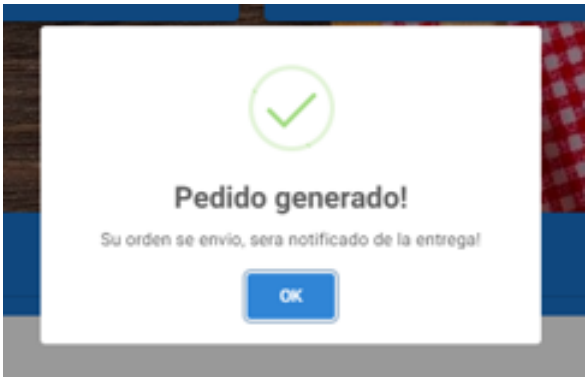


Figura 6.56: Mensaje de pedido ordenado
Elaborado por: Juan Mesias

Menú del cliente

- **Mi Perfil.** El submenú mi perfil permite al usuario con el rol de cliente modificar algunos de los datos personales que pueden variar con el tiempo como indica la digura 6.57, para la actualización de la información del usuario se realiza la estructura propuesta a través de la API-REST, y así en todos los casos de modificación e inserción de datos en la aplicación.

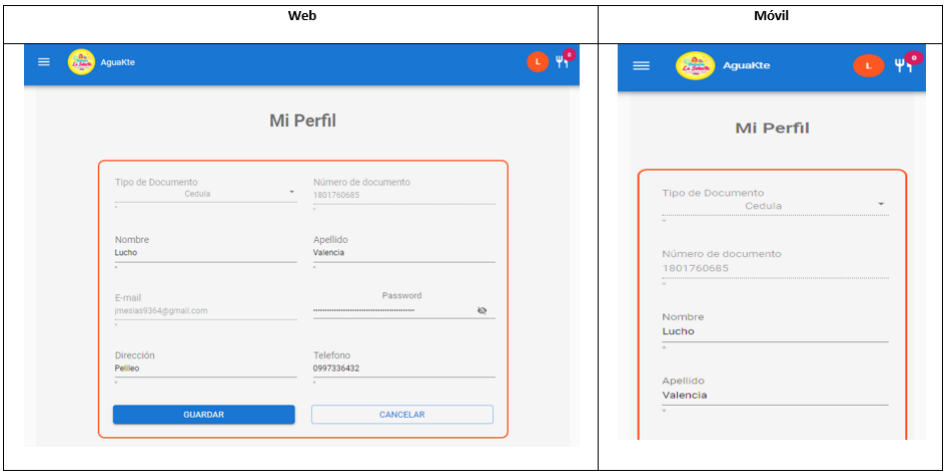


Figura 6.57: Pantalla de "Mi perfil"
Elaborado por: Juan Mesias

- **Mis datos de facturación.** En le submenú mis datos de facturación el usuario podrá solamente ingresar nuevos datos de facturación y observar los ya registrados con anterioridad. Figuras 6.58 y 6.59.

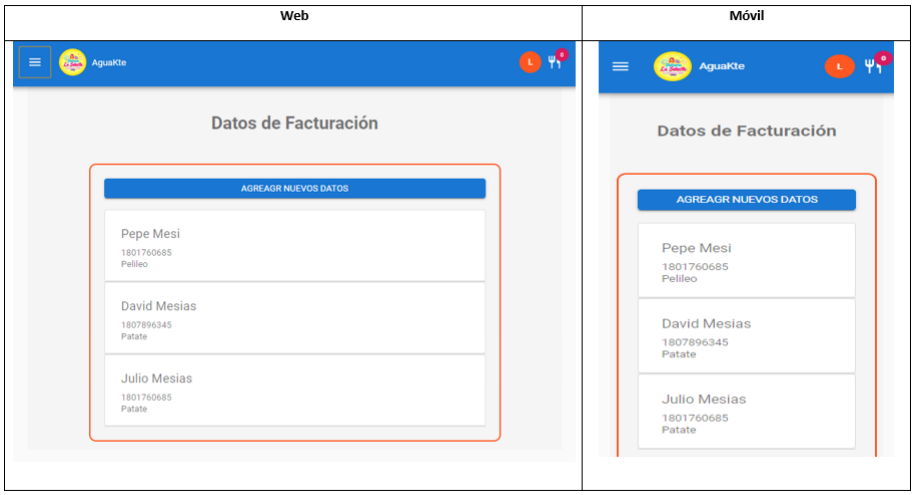


Figura 6.58: Pantalla de "Datos de facturación"
Elaborado por: Juan Mesias

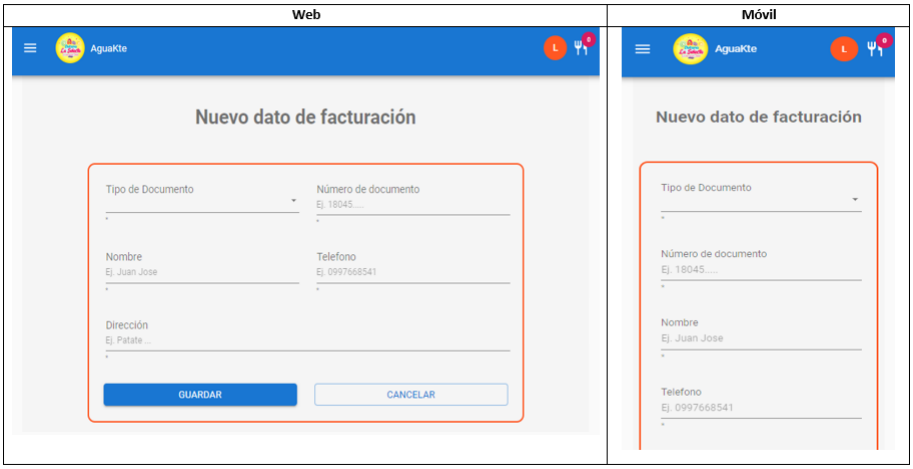


Figura 6.59: Pantalla de "Nuevos datos de facturación"
Elaborado por: Juan Mesias

- **Mis pedidos.** El submenú mis pedidos muestra los pedidos realizados del usuario logeado su estado su detalle y la factura. Se puede observar las figuras 6.60 y 6.61.

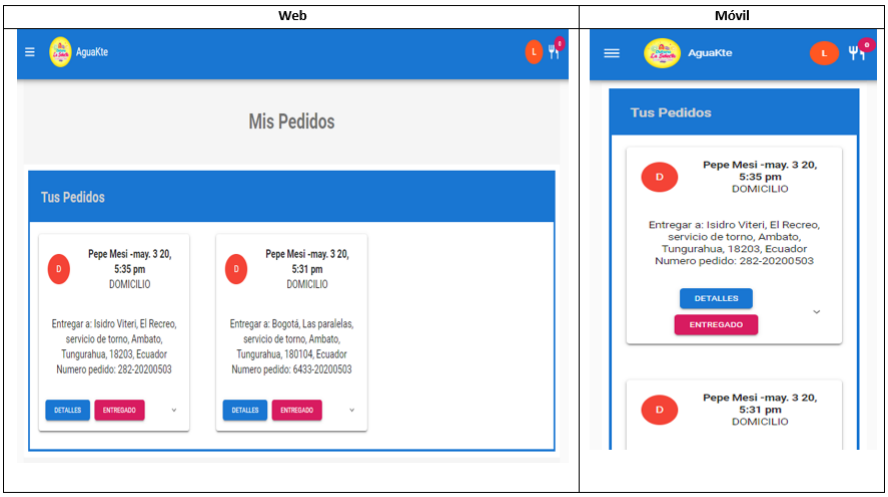


Figura 6.60: Pantalla de "Mis pedidos"
Elaborado por: Juan Mesias

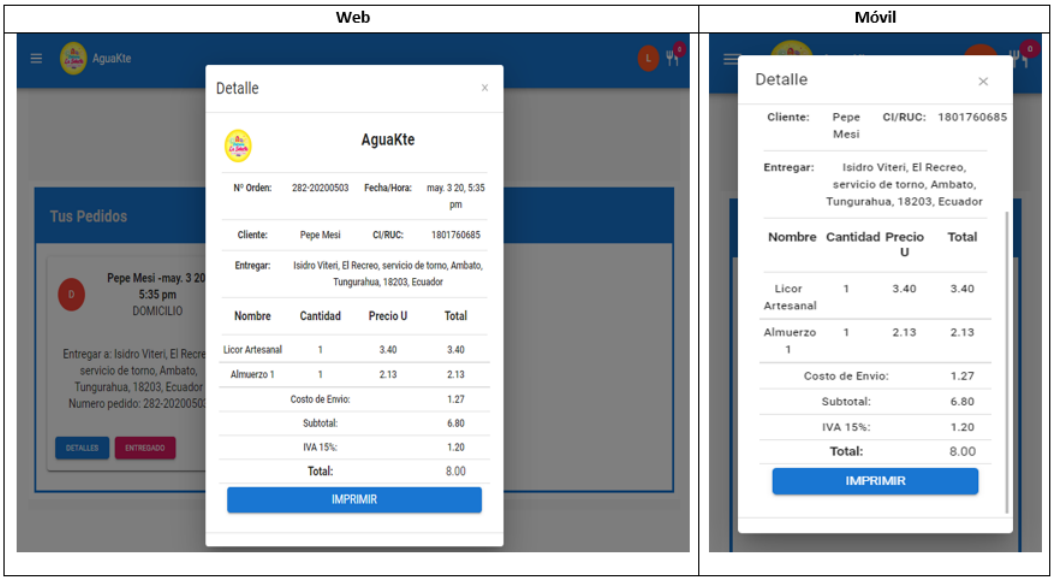


Figura 6.61: Pantalla de "Factura"
Elaborado por: Juan Mesias

Menú del Administrador

- **Restaurante.** El submenú indicado permite la actualización de la información del restaurante que se manejara para la presentación de la aplicación 6.62.

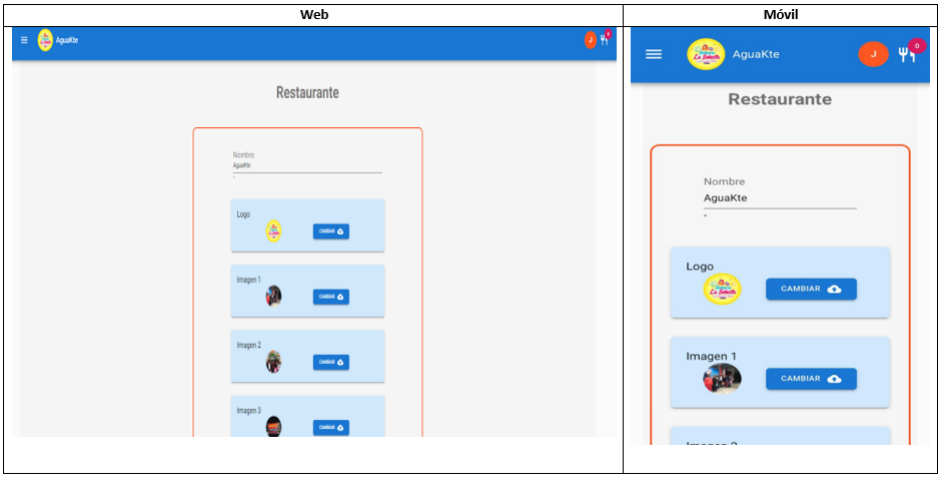


Figura 6.62: Pantalla de "Restaurante"
Elaborado por: Juan Mesias

- **Pedidos a preparar.** Esta pantalla contiene todos los pedidos que han sido recientemente añadidos por parte del cliente que estan listos para ser preparados, la pantalla muestra la información necesaria para que el pedido sea preparado y sea marcado como preparado lo que indicara que está listo para ser entregado al cliente. La pantalla esta dividida en tres secciones

como muestra la figura 6.63 y un mensaje de confirmación para marcar los pedidos como indica la figura 6.64. Al mismo tiempo el usuario será notificado por correo electrónico que su pedido está listo para ser entregado un ejemplo es la figura 6.65.

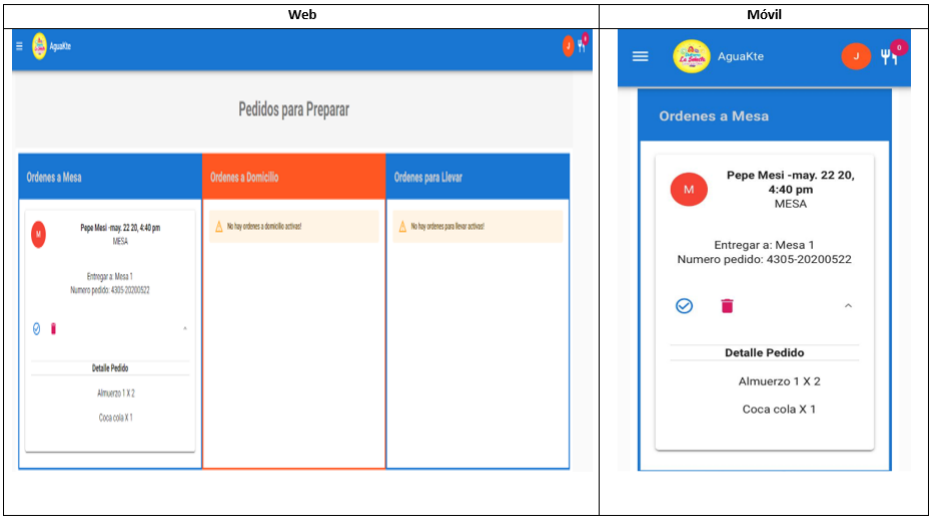


Figura 6.63: Pantalla de "Pedidos a preparar"
Elaborado por: Juan Mesias

Figura 6.64: Pantalla de "Confirmación de pedido marcado como preparado"
Elaborado por: Juan Mesías

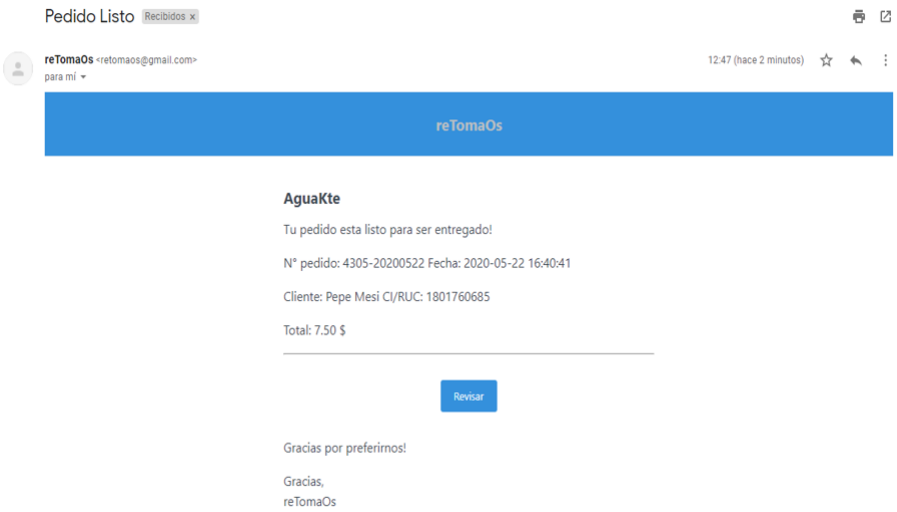


Figura 6.65: Correo electrónico enviado por el sistema al cliente
Elaborado por: Juan Mesias

- **Pedidos a Entregar.** El submenú indicado da la posibilidad de obtener los pedidos marcados como preparados listos para ser entregados al usuario y marcarlos como entregado lo que dará fin a el seguimiento que la aplicación hace a los pedidos. De la misma manera la pantalla se divide en tres secciones para diferenciar los pedidos como indica la figura 6.66 y un mensaje de confirmación para marcar el pedido como entregado en la figura 6.67.

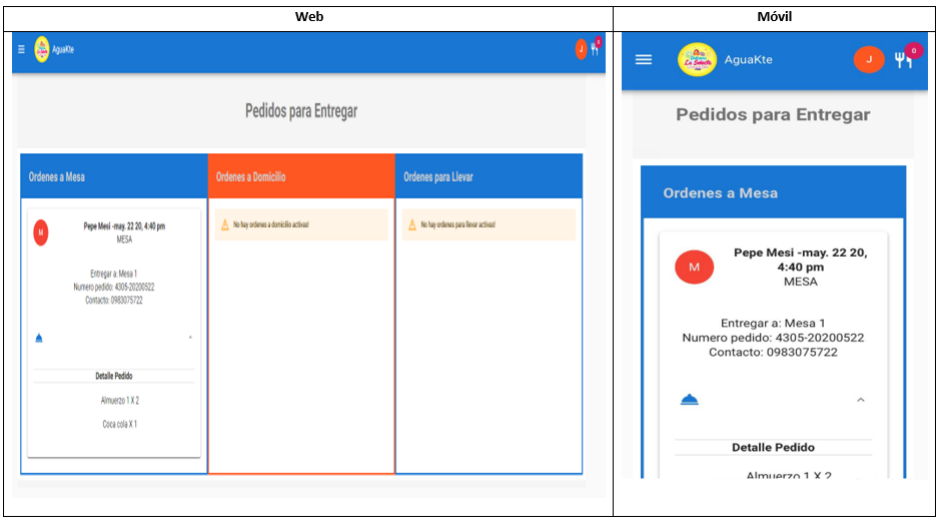


Figura 6.66: Pantalla de "Pedidos a entregar"
Elaborado por: Juan Mesias

Figura 6.67: Mensaje de "Confirmación de pedido marcado como entregado"
Elaborado por: Juan Mesias

- **Buscar Pedidos.** El submenú indicado permite al administrador de la aplicación buscar la información detallada de cada pedido. Se muestra en la figura 6.68.

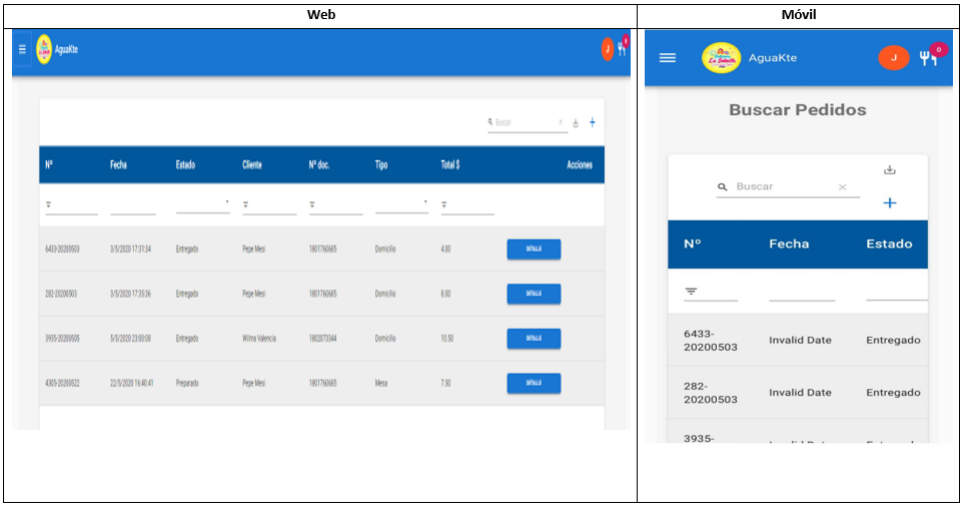


Figura 6.68: Pantalla "Buscar pedidos"
Elaborado por: Juan Mesias

- **Categorías.** El submenú permite administrar las categorías que contendrán a los productos, realizando las acciones principales inserción, actualización y lectura. La pantalla 6.69 indica una vista general de las categorías y la figura 6.70 indica la pantalla disponible para ingresar una nueva categoría.

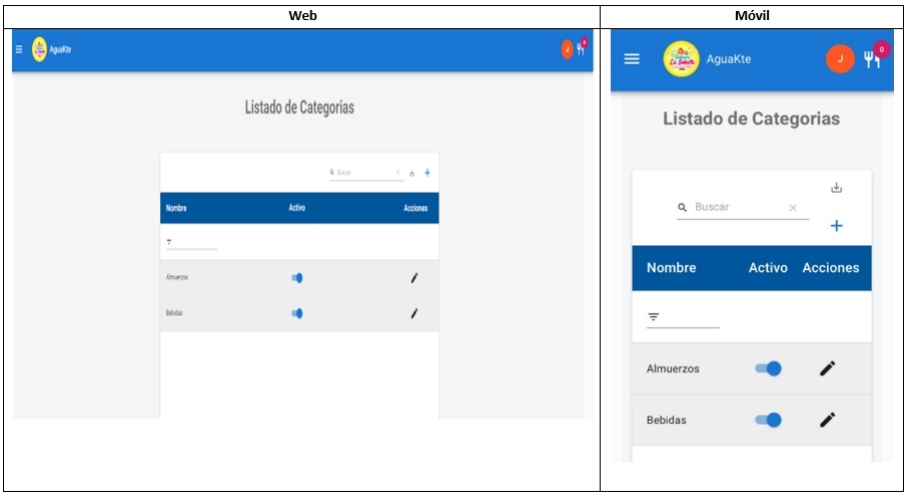


Figura 6.69: Pantalla "Categorías"
Elaborado por: Juan Mesias

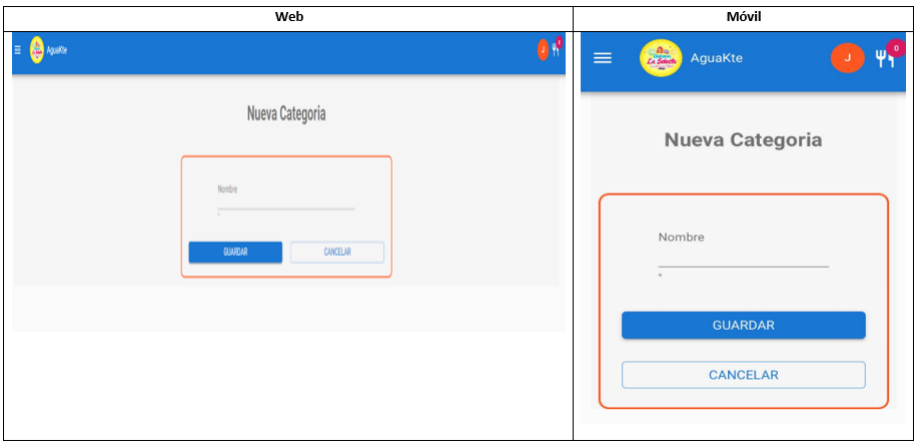


Figura 6.70: Pantalla "Nueva categoría"
Elaborado por: Juan Mesias

- **Productos.** El submenú permitirá administrar los productos que se quieren poner a disposición del cliente, realizando las acciones principales de inserción, actualización y lectura. La pantalla general de pedidos se muestra en la figura 6.71 y la pantalla para ingresar un nuevo producto en la figura 6.72.

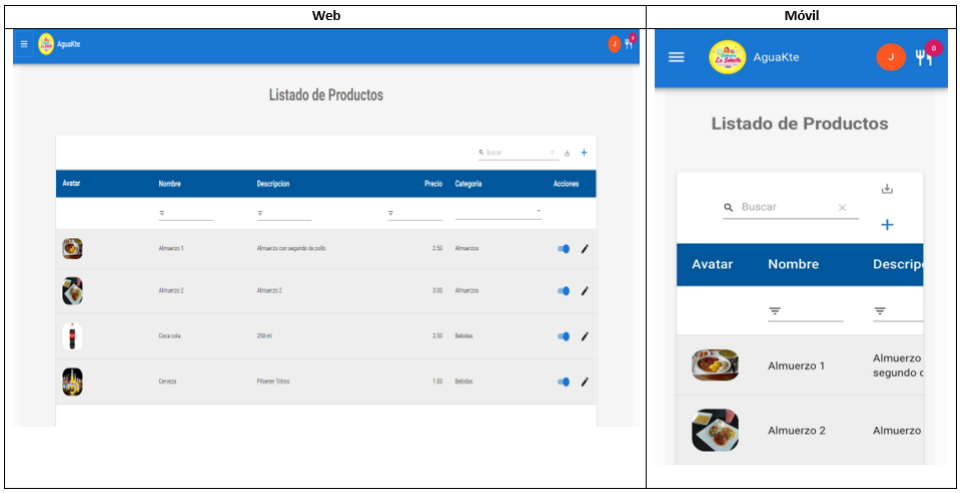


Figura 6.71: Pantalla "Listado de productos"
Elaborado por: Juan Mesias

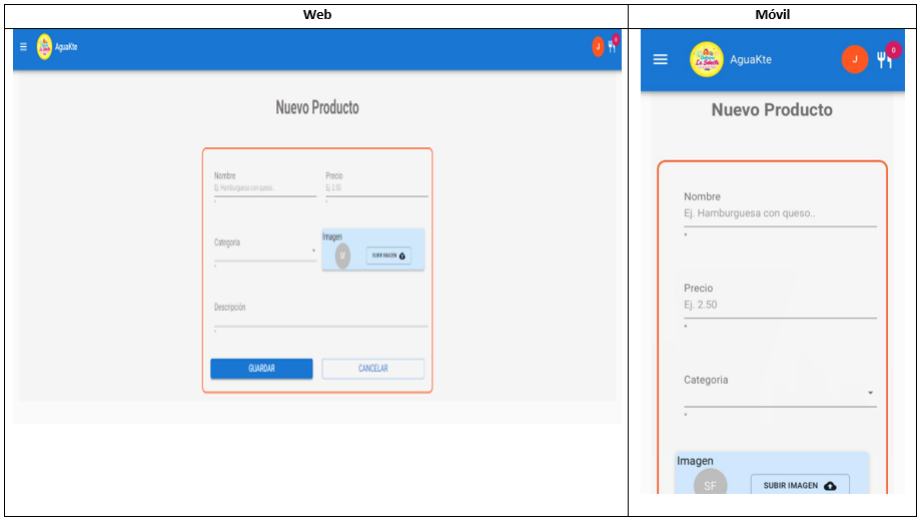


Figura 6.72: Pantalla "Nuevo producto"
Elaborado por: Juan Mesias

- **Mesas.** El submenú permite administrar las mesas creadas y realizar las acciones principales, también permite la impresión correspondiente del código QR de cada mesa como muestran las figuras 6.73,6.74, 6.75 .

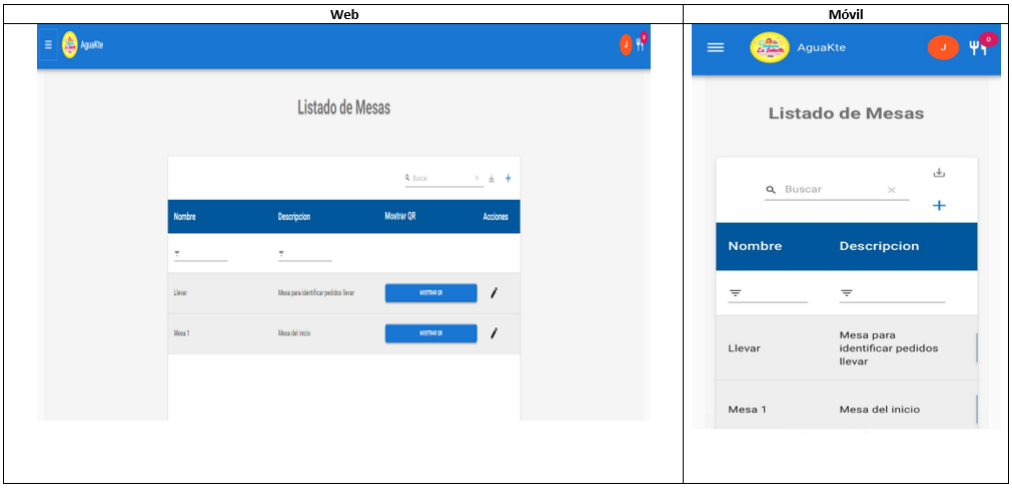


Figura 6.73: Pantalla “Mesas”
Elaborado por: Juan Mesias

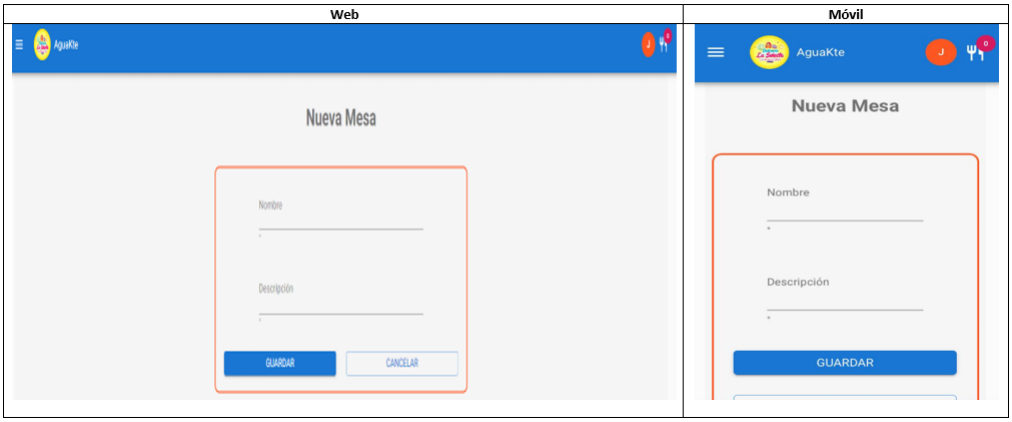


Figura 6.74: Pantalla “Nueva mesa”
Elaborado por: Juan Mesias

Figura 6.75: Pantalla "QR de una mesa"
Elaborado por: Juan Mesias

- **Usuarios.** El submenú usuarios permite al administrador manipular toda la información de usuarios registrados en la aplicación. También por medio del submenú el administrador está en la capacidad de crear nuevos usuarios con diferentes roles de la aplicación. La pantalla de general de usuarios representa la figura 6.76 y la pantalla de un nuevo usuario la figura 6.77

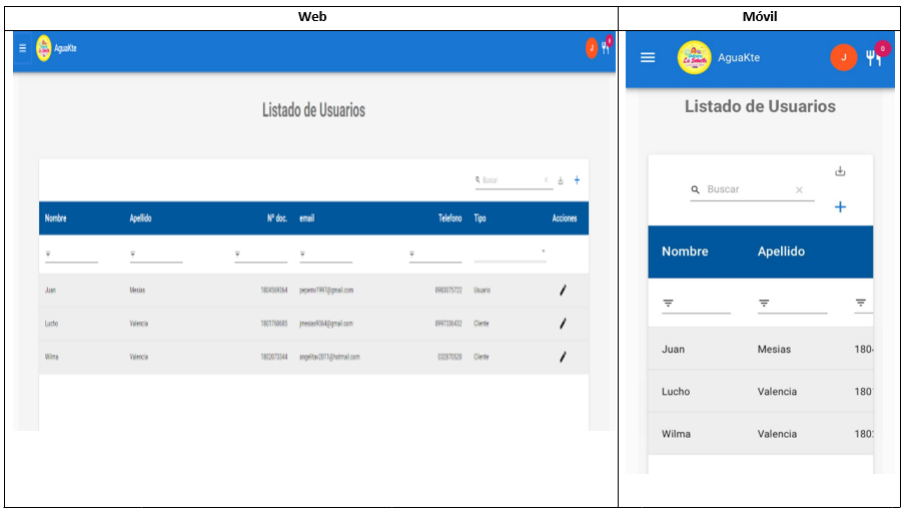


Figura 6.76: Pantalla "Usuarios"
Elaborado por: Juan Mesias

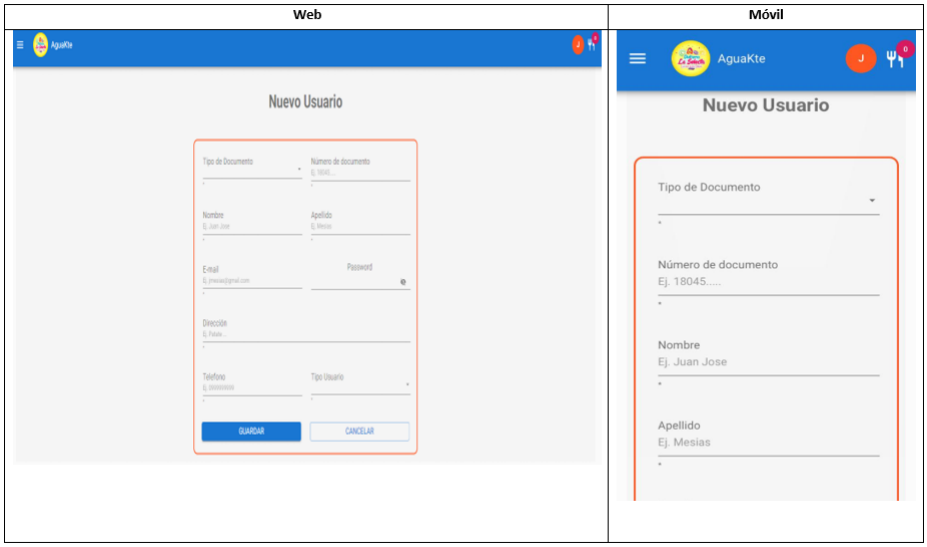


Figura 6.77: Pantalla “Nuevo usuario”
Elaborado por: Juan Mesias

Una vez revisada la funcionalidad en su totalidad se decidió establecer parámetros para evidenciar la calidad del software. El modelo propuesto para determinar la calidad es las dimensiones de calidad de Garvín, inicialmente no fue diseñado para determinar la calidad del software, pero sus medidas se ajustan perfectamente. Se basa el en dimensionamiento de ocho parámetros los cuales serán evaluados a continuación con una valoración de 1 a 5 entendiendo que por máximo valor se le considera al 5 y su mínimo valor a 1, esto se realizó bajo su debida justificación.

Parámetro	Valor	Justificación
Calidad de desempeño	4	El desempeño de la aplicación bajo la concurrencia establecida es el esperado y óptimo.
Calidad de características	4	El software esta diseñado bajo los principios de adaptabilidad a diferentes dispositivos por lo que contiene componentes con características especiales.
Confiabilidad	4	La aplicación de cifrado hash y la utilización de JWT para la comunicación entre cliente y servidor brindan una gran seguridad a la aplicación.
Conformidad	4	Al utilizar la metodología de desarrollo prototipo evolutivo los requerimientos se cumplieron a cabalidad con diferentes prototipos presentados.
Durabilidad	4	El software esta diseñado bajo el concepto de componentes como su tecnología de desarrollo lo permite, lo cual permite un fácil mantenimiento y escalabilidad del mismo.
Servicio	3	El software esta apto para entrar en un tiempo de mantenimiento breve en caso de ser necesario.
Estética	4	El diseño de la interfaz de la aplicación es un diseño simple basado en la filosofía menos es mas de esta manera la interfaz muestra al usuario lo necesario para guiarse y desenvolverse en la misma.

Figura 6.78: Pantalla “Parametros Garvín”
Elaborado por: Juan Mesias

3.6.2 Pruebas de estrés

El software seleccionado para realizar las pruebas de estres fue ApacheBench(ab), por ser una herramienta que viene incluida en el servidor de Apache, ademas de ser muy sencilla de usar por medio de lineas de comandos lo cual es adecuado para la experimentación de usuarios principiantes e intermedios.

Las pruebas de estres fueron diseñadas en dos etapas tomando encuenta el despliegue de la aplicación en una red local sobre la ip 192.168.1.11:3000 y sobre el despliegue de la aplicacion en netlify, plataforma que permite el hospedaje de aplicaciones javascript gratuitamente.
En la figura 6.79 se puede observar el resultado de un numero de cien solicitudes con una concurrencia de diez hacia la aplicación en la red local.

```
# ab -n 100 -c 10 http://192.168.1.11:3000/
This is ApacheBench, Version 2.3 <$Revision: 1843412 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking 192.168.1.11 (be patient).....done


Server Software:
Server Hostname:        192.168.1.11
Server Port:            3000

Document Path:          /
Document Length:        2878 bytes

Concurrency Level:      10
Time taken for tests:   0.065 seconds
Complete requests:      100
Failed requests:        0
Total transferred:      312600 bytes
HTML transferred:       287800 bytes
Requests per second:    1527.44 [#/sec] (mean)
Time per request:       6.547 [ms] (mean)
Time per request:       0.655 [ms] (mean, across all concurrent requests)
Transfer rate:          4662.87 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   0.5      0       2
Processing:     3    6   2.0      5      11
Waiting:        0    5   1.7      4       9
Total:          3    6   2.1      6      12

Percentage of the requests served within a certain time (ms)
  50%      6
  66%      7
  75%      7
  80%      8
  90%      9
  95%     10
  98%     11
  99%     12
 100%     12 (longest request)
```

Figura 6.79: Pantalla “Prueba a 192.168.1.11:3000”
Elaborado por: Juan Mesias

En la figura 6.80 se puede observar la misma prueba realizada hacia el dominio jjmv.netlify.app

```
# abs -n 100 -c 10 https://jjmv.netlify.app/
This is ApacheBench, Version 2.3 <$Revision: 1843412 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking jjmv.netlify.app (be patient).....done


Server Software:        Netlify
Server Hostname:        jjmv.netlify.app
Server Port:            443
SSL/TLS Protocol:       TLSv1.2,ECDHE-RSA-AES256-GCM-SHA384,2048,256
Server Temp Key:        X25519 253 bits
TLS Server Name:        jjmv.netlify.app

Document Path:          /
Document Length:        3013 bytes

Concurrency Level:      10
Time taken for tests:   15.608 seconds
Complete requests:      100
Failed requests:        0
Total transferred:      336214 bytes
HTML transferred:       301300 bytes
Requests per second:    6.41 [#/sec] (mean)
Time per request:       1560.833 [ms] (mean)
Time per request:       156.083 [ms] (mean, across all concurrent requests)
Transfer rate:          21.04 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:      374  877 281.2    852    1469
Processing:   138  570 296.3    568    1408
Waiting:      129  341 211.9    278    1061
Total:        696 1447 135.6   1442    1783

Percentage of the requests served within a certain time (ms)
  50%   1442
  66%   1504
  75%   1518
  80%   1521
  90%   1540
  95%   1579
  98%   1644
  99%   1783
 100%   1783 (longest request)
```

Figura 6.80: Pantalla “Prueba a jjmv.netlify.app”
Elaborado por: Juan Mesias

Se puede observar una leve variación en el tiempo de respuesta y en el tiempo que se tarda en realiza el test ya que en la red local la respuesta siempre será un poco mas rápida, a comparación de realizar un test hacia un servidor remoto.
La concurrencia de las peticiones no afecta en su respuesta ya que el test de ambos casos devuelve cien de cien peticiones realizadas y cero peticiones fallidas.
A continuacion se muestra la misma prueba con una varicación en el número de peticiones a mil y la concurrencia a cien. La figura 6.81 muestra el resultado de la prueba hacia la red local y la figura 6.82 hacia netlify.

Despues de ejecutar el comando "ab -n 1000 -c 100 http://192.168.1.11" se obtuvo el siguiente resultado.

```
Server Software:
Server Hostname:        192.168.1.11
Server Port:            3000

Document Path:          /
Document Length:        2878 bytes

Concurrency Level:      100
Time taken for tests:   0.494 seconds
Complete requests:      1000
Failed requests:        0
Total transferred:      3126000 bytes
HTML transferred:       2878000 bytes
Requests per second:    2025.48 [#/sec] (mean)
Time per request:       49.371 [ms] (mean)
Time per request:       0.494 [ms] (mean, across all concurrent requests)
Transfer rate:          6183.25 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   0.4      0       1
Processing:    19   46   9.6     48      67
Waiting:        1   37   7.5     37      58
Total:         19   47   9.6     48      67

Percentage of the requests served within a certain time (ms)
  50%     48
  66%     50
  75%     53
  80%     54
  90%     59
  95%     62
  98%     65
  99%     66
 100%     67 (longest request)
```

Figura 6.81: Pantalla "Prueba a 192.168.1.11:3000"
Elaborado por: Juan Mesias

Con el comando "abs -n 1000 -c 100 https://jjmv.netlify.app" se obtuvo el siguiente resultado.

```
Server Software:        Netlify
Server Hostname:        jjmv.netlify.app
Server Port:            443
SSL/TLS Protocol:       TLSv1.2,ECDHE-RSA-AES256-GCM-SHA384,2048,256
Server Temp Key:        X25519 253 bits
TLS Server Name:        jjmv.netlify.app

Document Path:          /
Document Length:        3013 bytes

Concurrency Level:      100
Time taken for tests:   173.239 seconds
Complete requests:      1000
Failed requests:        0
Total transferred:      3362382 bytes
HTML transferred:       3013000 bytes
Requests per second:    5.77 [#/sec] (mean)
Time per request:       17323.930 [ms] (mean)
Time per request:       173.239 [ms] (mean, across all concurrent requests)
Transfer rate:          18.95 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:      383 7983 3691.1   8181   17748
Processing:   133 8065 3709.1   7905   16483
Waiting:      126 3339 1743.0   3302   13430
Total:      11197 16047 906.2  15817   19104

Percentage of the requests served within a certain time (ms)
  50%  15817
  66%  16108
  75%  16372
  80%  16691
  90%  17444
  95%  17825
  98%  17936
  99%  18433
 100%  19104 (longest request)
```

Figura 6.82: Pantalla "Prueba a jjmv.netlify.app"
Elaborado por: Juan Mesias

Las pruebas realizadas muestran que el tiempo de respuesta en realizar el test es variante pero el porcentaje de solicitudes exitosas y fallidas sigue siendo el óptimo.

CAPÍTULO IV

Conclusiones y recomendaciones

4.1 Conclusiones

La implementación de este proyecto permitió obtener una solución adaptable a todo tipo de dispositivos con un rendimiento óptimo. El aplicativo permite la toma y gestión de órdenes realizadas por un usuario dentro del restaurante al escanear un código QR perteneciente a una mesa; de la misma manera se gestiona las órdenes que son solicitadas a domicilio. Las principales conclusiones a las que se llego después de realizar este trabajo son:

- Se ha propuesto un nuevo modelo de negocio para la gestión de órdenes en restaurantes que permite a los usuarios realizar sus pedidos de manera ágil y reducir sus tiempos de espera. Al utilizar un aplicativo para realizar los pedidos la interacción entre personas dentro de un mismo espacio se ve reducida además de brindarle al usuario una experiencia nueva en lo que se refiere a la gestión de un restaurante.

- Al utilizar una tecnología como Reactjs, el desarrollo de la aplicación se vio reflejado en una arquitectura ordenada, basada en componentes que se comunican entre sí. Además de permitir un desarrollo ágil, la asociación de los datos con las vistas es de manera directa. Consecuentemente, si algún dato sufre alguna modificación, esto se ve reflejado automáticamente en la vista del usuario. Como resultado, el tiempo de programación se vio reducido debido a la reutilización de componenetes y a la actualización de vistas asociadas con los datos de manera automática.

- El desarrollo e implementación de una aplicación para la gestión de órdenes en restaurantes se logró de manera satisfactoria al aplicar los conocimientos obtenidos durante la carrera y gracias a la disponibilidad de herramientas para el desarrollo de software.

4.2 Recomendaciones

- Para futuras investigaciones se sugiere el estudio de React como motor para el desarrollo de aplicaciones nativas tanto para Android como iOS. De esta manera la aplicación podrá ser descargada e instalada directamente de una tienda oficial.

- Para la optimización de la aplicación, probar nuevos componentes de React para la generación de códigos QR y para la lectura de los mismos.

- Se sugiere la investigación de Nodejs como herramienta para el desarrollo del lado del servidor por su semejanza con javascript ya que facilita el aprendizaje.

- Con un mayor conocimiento sobre el diseño web enfocado hacia dispositivos móviles se puede optimizar el funcionamiento de la aplicación y la presentación de la información en la misma.

- La investigación del funcionamiento y aplicación de WebSockets en diferentes plataformas podría ayudar a comprender el trabajo que estos realizan dentro de varias plataformas que utilizamos diariamente y además de poder realizar una implementación propia semejante o mejorada en nuestros proyectos.

- Incentivar el desarrollo de aplicaciones de comercio electrónico debido a la disponibilidad de herramientas de libre acceso para su desarrollo.

Bibliografía

[1] Ministerio de Turismo. Ecuador cuenta con un Plan Nacional Gastronómico para promover el turismo. [Online] https://www.turismo.gob.ec/ecuador-cuenta-con-un-plan-nacional-gastronomico-para-promover-el-turismo-en-el-pais/, 2018.

[2] Marco Antonio Muñoz Torrealva. *Desarrollo de una aplicación móvil para la realización de reservas y toma de órdenes en el restaurante LongHorn.* PhD thesis, Universidad Inca Garsilazo de la Vega, 2017.

[3] Diana Cristina Altamirano Andrade. *Aplicación móvil con realidad aumentada como estrategia de marketing 2.0 para el menú del restaurante chimichurri moros & menestras en la ciudad de Ambato.* PhD thesis, Universidad Tecnica de Ambato, 2017.

[4] El Heraldo. Restaurantes asociados mejoran servicio. [Online] https://www.elheraldo.com.ec/restaurantes-asociados-mejoran-servicio/, 2018.

[5] Jesús Segarra-Saavedra, Tatiana Hidalgo-Marí, and Eliseo Rodríguez-Monteagudo. La gastronomía como Industria Creativa en un contexto digital. Análisis de webs y redes sociales de los restaurantes españoles con estrella Michelin. *adComunica*, (10):135–154, 2015.

[6] GALO FERNANDO PEÑAHERRERA MORÁN and BRIAN ANTONIO TORRES REYNOSO. *Automatización de la gestión de órdenes de pedidos para restaurantes con servicio a la mesa y a domicilio.* PhD thesis, Universidad de Guayaquil, 2016.

[7] PAUL ANDRES OCHOA QUINTEROS. *Aplicación Interactiva Para Gestión De Órdenes Y Pedidos En Restaurantes.* PhD thesis, Universidad del Azuay, 2014.

[8] Mike Wolcott. What Is Web 2.0? - CBS News. [Online] https://www.cbsnews.com/news/what-is-web-20/ http://www.cbsnews.com/8301-505125_162-51066094/what-is-web-20/, 2007.

[9] Goodwill Community Fundation. Informática Básica: ¿Qué son las aplicaciones web? [Online] https://edu.gcfglobal.org/es/informatica-basica/que-son-las-aplicaciones-web/1/.

[10] Google. Tu primera Progressive Web App — Web Fundamentals — Google Developers. [Online] https://developers.google.com/web/fundamentals/codelabs/your-first-pwapp?hl=es, 2020.

[11] Apache Friends. XAMPP Installers and Downloads for Apache Friends. [Online] https://www.apachefriends.org/es/index.html, 2017.

[12] P. DuBois. *MySQL*. Developer's Library. Pearson Education, 2008.

[13] Red Hat. ¿Qué es una API? [Online] https://www.redhat.com/es/topics/api/what-are-application-programming-interfaces, 2014.

[14] About Us - About Us LEGO.com. [Online] https://material-ui.com/es/company/about/ http://www.lego.com/en-gb/aboutus?ignorereferer=true.

[15] Bootstrap. Bootstrap · La biblioteca de HTML, CSS y JS más popular del mundo. [Online] https://getbootstrap.com/, 2019.

[16] React. React – Una biblioteca de JavaScript para construir interfaces de usuario. [Online] https://es.reactjs.org/, 2019.

[17] React. ¿Por qué usar React Redux? · Reaccionar Redux. [Online] https://react-redux.js.org/introduction/why-use-react-redux.

[18] A. DE DIEGO MORILLO. *Operaciones auxiliares de almacenaje*. Ediciones Paraninfo, S.A., 2018.

[19] Jwt.io. Introducción al token web de JSON. [Online] https://jwt.io/introduction/, 2019.

[20] WebSockets - Web API reference — MDN. [Online] https://developer.mozilla.org/es/docs/Web/API/WebSockets_API.

[21] JavaScript — MDN. [Online] https://developer.mozilla.org/es/docs/Web/JavaScript.

[22] Miguel Angel Alvarez. Librería Axios: cliente HTTP para Javascript. [Online] https://desarrolloweb.com/articulos/axios-ajax-cliente-http-javascript.html, 2018.

[23] Osterwalder Alexander and Yves Pigneur. Generacion de modelos de negocio. *Journal of Product Innovation Management*, 13(2):180–181, 1996.

[24] Flor Díaz Piraquive. Gestión de procesos de negocio BPM (Business Process Management), TIC y crecimiento empresarial : ¿Qué es BPM y cómo se articula con el crecimiento empresarial? *Universidad & Empresa*, 10(15):151–176, 2008.

[25] Rodolfo F. Schmal and Teresa Y. Olave. Optimización del Proceso de Atención al Cliente en un Restaurante durante Períodos de Alta Demanda. *Informacion Tecnologica*, 25(4):27–34, 2014.

[26] Rocío A Rodríguez, Pablo M Vera, M Roxana Martínez, and Fernando A Parra Beltrán. Aplicaciones Web Progresivas Impulsadas por el Avance de los Estándares Web. page 5, 2019.

[27] Pablo Berbel Marín. Desarrollo de un frontend en ReactJS. 2017.

[28] Xianjun Chen, Zhoupeng Ji, Yu Fan, and Yongsong Zhan. Restful API Architecture Based on Laravel Framework. *Journal of Physics: Conference Series*, 910(1), 2017.

[29] Tailor Otwell. Introduction - Laravel - The PHP Framework For Web Artisans. [Online] https://laravel.com/docs/4.2/introduction.

[30] Django Software Foundation. The Web framework for perfectionists with deadlines — Django. [Online] https://www.djangoproject.com/, 2018.

[31] What is ASP.NET? — .NET. [Online] https://dotnet.microsoft.com/learn/aspnet/what-is-aspnet, 2020.

[32] Angular.io. Introduction to Angular concepts. [Online] https://angular.io/guide/architecture.

[33] Evan You. Introducción - Vue.js. [Online] https://vuejs.org/v2/guide/ https://vuejs.org/v2/guide/index.html#What-is-Vue-js, 2016.

[34] JP Erkkilä. Websocket security analysis. *Aalto University School of Science*, 2012.

Anexo A

Desarrollo de la API-REST

Laravel soporta la conexión a diferentes bases de datos. Para el presente proyecto se ha seleccionado la base de datos MySQL con Maria DB, conexión que se configuró en el archivo .env, tal y como se muestra en la figura 0.1.

```
1    DB_CONNECTION=mysql
2    DB_HOST=localhost
3    DB_PORT=3306
4    DB_DATABASE=propuesta1db
5    DB_USERNAME=**********
6    DB_PASSWORD=*********
```

Figura 0.1: Archivo .env
Elaborado por: Juan Mesías

Para comenzar a crear la API-REST en Laravel es fundamental tener claro el modelo de base de datos ya que el mismo es representado bajo los modelos y controladores que se crean en la aplicación.
Los modelos son los encargados de contener la información específica de las tablas en la base de datos, lo que permite actuar sobre las mismas. Los modelos creados en el proyecto se definieron bajo del directorio app/, como lo muestra la figura 0.2

Los modelos pueden ser alterados una vez creados dentro de la aplicación para definir el comportamiento de los datos que se van a consultar o modificar en la base de datos. Por ejemplo, el modelo de la entidad Usuario denominado como "User", al cual se le agregó algunas características para mantener la integridad de los datos personales del usuario, propiedades de notificación y características para la autenticación mediante JWT, como se muestra en la figura 0.3.

Una vez definido los modelos de la aplicación se procede a crear los controladores bajo el directorio.

`app/Http/Controllers`

Los cuales contienen los métodos para modificar los datos basandose en la estructura de cada modelo ,ya que cada controlador extiende su funcionalidad a partir de uno o varios modelos. Los controladores creados para las interacciones de los datos se muestran en la figura 0.4

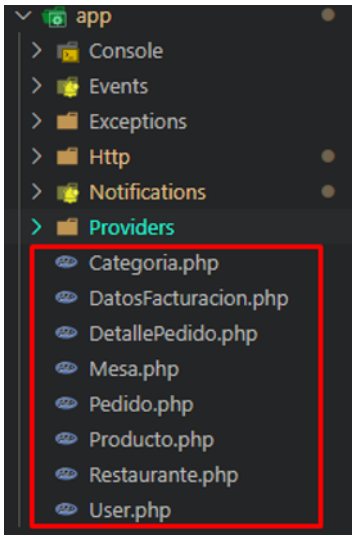


Figura 0.2: API-REST en Laravel
Elaborado por: Juan Mesías

```php
<?php
use Illuminate\Notifications\Notifiable;
use Illuminate\Foundation\Auth\User as Authenticatable;
use Tymon\JWTAuth\Contracts\JWTSubject;
class User extends Authenticatable implements JWTSubject
{
use Notifiable;
    public $timestamps = false;
protected $table = 'usuario';
    protected $fillable = [
    'nombre', 'email', 'password', 'apellido',
    ];

    protected $hidden = [
    'password', 'remember_token',
    ];

    public function getJWTIdentifier()
        {
        return $this->getKey();
        }
    public function getJWTCustomClaims()
        {
        return [];
        }
}
```

Figura 0.3: Modelo "User"
Elaborado por: Juan Mesías

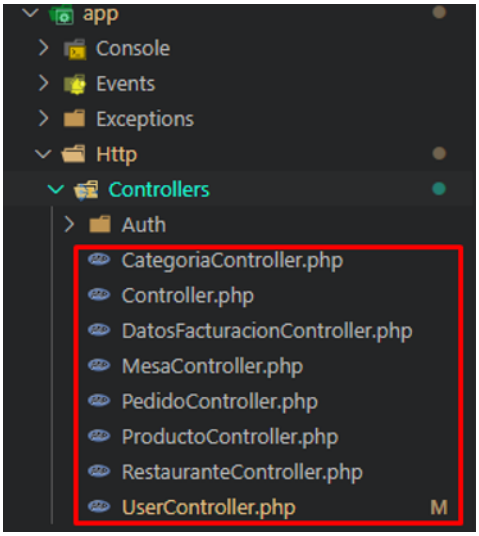


Figura 0.4: Controllers en Laravel
Elaborado por: Juan Mesías

Los controladores son ejecutados en dependencia de cada solicitud, la que utiliza un método específico GET, POST, PUT o DELETE. Algunos controladores adoptan funcionalidades extra como la de generación de tokens para la autenticación, las de envio de notificaciones y las de accionadores de eventos.
El controlador del usuario definido como "UserController" contiene métodos en los que se importan otras clases requeridas para la implementación de las funcionalidades que se requieren. El controlador del usuario implementa la autenticación mediante JWT. La biblioteca de JWT para PHP es utilizada incluyendo el paquete llamado "tymon/jwt-auth". Una vez instalado el paquete, se debe agregar la configuración en el archivo config/app.php de la forma que se muestra en la figura 0.5.

```
'providers' => [ ..., Tymon\JWTAuth\Providers
\LaravelServiceProvider::class,]

'aliases' => [ ..., 'JWTAuth' => Tymon\JWTAuth\Facades\JWTAuth::class,
        'JWTFactory' => Tymon\JWTAuth\Facades\JWTFactory::class,]
```

Figura 0.5: Paquete 'tymon/jwt-auth"
Elaborado por: Juan Mesías

Una vez que se agregó la biblioteca de JWT para la autenticación del usuario, se procedió a crear el controlador que contiene los métodos de registro, actualización de datos del usuario, obtención de listado de usuarios, obtención de usuario por id, autenticación y cerrar sesión.

El método register permite al usuario crear un nuevo usuario dentro de la aplicación. El registro del nuevo usuario en la base de datos se realiza despues de validar los campos necesarios y transformar la contraseña a un "hash", tal y como se puede comprobar en la figura 0.6.

```
1         public function register(Request $request)
2     {
3
4         $validation = Validator::make(
5             $request->all(),
6             [
7                 'nombre' => 'required',
8                 'apellido' => 'required',
9                 'tipoDocumento' => 'required',
10                'numeroDocumento' => 'required',
11                'email' => 'required|email',
12                'password' => 'required',
13                'telefono' => 'required',
14                'direccion' => 'required',
15                'tipoUsuario' => 'required',
16            ]
17        );
18        $user = new User();
19        //...
20        $user->password = Hash::make($request->password);
21        //..
22        $user->save();
23
24        $token = JWTAuth::fromUser($user);
25
26        return response()->json(
27            [
28                'usuario' => $user,
29                'token' => $token,
30                'HttpResponse' => [
31                    'tittle' => 'Correcto',
32                    'message' => 'Usuario creado!',
33                    'status' => 200,
34                    'statusText' => 'success',
35                    'ok' => true
36                ],
37            ],
38            201
39        );
40    }
```

Figura 0.6: Método "register"
Elaborado por: Juan Mesías

El método actualizar usuario permite modificar ciertos datos del usuario registrado. El proceso de actualización inicia verificando la existencia del usuario a modificar y validando los campos necesarios 0.7.

```
1         public function actualizarUsuario(Request $request, $id){
2         $usuario = User::find($id);
3         if (!$usuario) {
4             return response()->json([
5                 'HttpResponse' => [
6                     'tittle' => 'Error',
7                     'message' => 'No se encontro el usuario!',
8                     'status' => 400,
9                     'statusText' => 'error',
10                    'ok' => true
11                ]
12            ]);
13        }
14
15                //...
16
17        $usuario->save();
18
19        return response()->json(
20            [
21                'usuario' => $usuario,
22                'HttpResponse' => [
23                    'tittle' => 'Correcto',
24                    'message' => 'Usuario actualizado!',
25                    'status' => 200,
26                    'statusText' => 'success',
27                    'ok' => true
28                ],
29            ],
30            201
31        );
32    }
```

Figura 0.7: Método “actualizarUsuario”
Elaborado por: Juan Mesías

El método authenticate permite al usuario crear un token para manejar su sesión dentro de la aplicación 0.8.

```
        public function authenticate(Request $request)
    {
        $credentials = $request->only('email', 'password');
        try {
            if (! $token = JWTAuth::attempt($credentials)) {

                return response()->json(
                    [
                        'HttpResponse' => [
                            'tittle' => 'Error',
                            'message' => 'Credenciales invalidas!',
                            'status' => 200,
                            'statusText' => 'error',
                            'ok' => true
                        ],
                    ],
                    400
                );
            }
        } catch (JWTException $e) {
            return response()->json(
                [
                    'HttpResponse' => [
                        'tittle' => 'Error',
                        'message' => 'No se pudo crear el acceso!',
                        'status' => 200,
                        'statusText' => 'error',
                        'ok' => true
                    ],
                ],
                500
            );
        }

        return response()->json(
            [
                'token' => $token,
                'HttpResponse' => [
                    'tittle' => 'Correcto',
                    'message' => 'Logeado!',
                    'status' => 200,
                    'statusText' => 'success',
                    'ok' => true
                ],
            ],
            201
        );
    }
```

Figura 0.8: Método “authenticate”
Elaborado por: Juan Mesías

El método logout permite al usuario cerrar la sesión, indicando que el token generado con anterioridad sea inválido 0.9.

```
1         public function logout()
2     {
3         JWTAuth::invalidate();
4         return response()->json(
5             [
6                 'HttpResponse' => [
7                     'tittle' => 'Correcto',
8                     'message' => 'Ya no esta logeado!',
9                     'status' => 200,
10                    'statusText' => 'success',
11                    'ok' => true
12                ],
13            ],
14            201
15        );
16    }
```

Figura 0.9: Método "logout"
Elaborado por: Juan Mesías

El método index realiza la consulta de todos los usuarios para su administración 0.10.

```
1         public function index()
2     {
3         return response()->json(
4             [
5                 'usuarios' => User::all(),
6                 'HttpResponse' => [
7                     'status' => 200,
8                     'statusText' => 'OK',
9                     'ok' => true
10                ]
11            ],
12            201
13        );
14    }
```

Figura 0.10: Método "index"
Elaborado por: Juan Mesías

El método mostrado en la figura 0.11 consulta los datos de un usuario en especifico.

```
1         public function getUserByid(Request $request){
2         $usuario = User::find($request->id);
3         if (!$usuario) {
4             return response()->json([
5                 'HttpResponse' => [
6                     'tittle' => 'Error',
7                     'message' => 'No se encontro el usuario!',
8                     'status' => 400,
9                     'statusText' => 'error',
10                    'ok' => true
11                ]
12            ]);
13        }else{
14            return response()->json(
15                [
16                    'usuario' => $usuario,
17                    'HttpResponse' => [
18                        'status' => 200,
19                        'statusText' => 'success',
20                        'ok' => true
21                    ]
22                ],
23                201
24            );
25        }
26    }
```

Figura 0.11: Método “getUserByid”
Elaborado por: Juan Mesías

Uno de los controladores más importantes en el desarrollo de la aplicación fue el controlador de los pedidos denominado “PedidoController”; este controlador es el encargado de gestionar la información de los pedidos realizados en la tabla pedidos y detalle de pedido dentro de la base de datos. Además de esto, controla el desencadenamiento de eventos “Events” que trabajan bajo WebSockets, para llevar un control en tiempo real del estado del pedido en la aplicación. El controlador también es el encargado de enviar una notificación de ”Pedido Preparado” al usuario dependiendo del estado en el que se encuentra usando las denominadas “Notifications”. Laravel incluye la característica de notificaciones por diferentes canales sin necesidad de instalación de paquetes externos.
Los WebSokets son definidos como un canal de comunicación bidireccional en tiempo real sobre el protocolo TCP. La principal característica de los WebSockets es que no siguen el protocolo tradicional de las peticiones HTTP. Siempre que un cliente abra una conexión hacia el servidor mediante WebSockets, este canal de comunicación permanecerá abierto para el envío y recepción de información. La conexión permanecerá abierta hasta que el cliente o el servidor cierren dicha conexión [34].

La herramienta en Laravel para trabajar con WebSockets es Laravel WebSockets, una biblioteca que debe ser instalada como un paquete adicional con el comando:

```
\composer require beyondcode/laravel-websockets"
```

Una vez instalado el paquete se publica el archivo de configuración con el comando

```
\php artisan vendor:publish
--provider="BeyondCode\LaravelWebSockets
\WebSocketsServiceProvider" --tag="config" "
```

El archivo publicado se encuentra en la ruta config/websockets.php. Este fichero contiene toda la configuración del paquete. La configuración de credenciales y de la conexión también se lo puede hacer en el archivo .env (ver figura 0.12).

```
1    PUSHER_APP_ID=12345
2    PUSHER_APP_KEY=**********
3    PUSHER_APP_SECRET=**********
4    PUSHER_APP_CLUSTER=mt1
```

Figura 0.12: Archivo .env del Web Socket
Elaborado por: Juan Mesías

Despues de la configuración correcta la implementacion de WebSockets se realizó a través de la creación de eventos "Events" asi llamados en Laravel, los cuales son los encargados de la transmision bidireccional de datos en tiempo real a través de un canal definido.
Los eventos creados para el manejo del estado de los pedidos en la aplicación se crearon bajo el directorio "app/Events", con las definiciones correspondientes, tal y como se muestra en la figura 0.13.

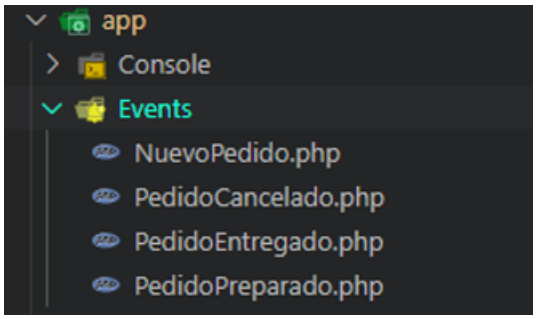


Figura 0.13: Eventos
Elaborado por: Juan Mesías

La ejecución del evento se da al momento de crear un pedido y actualizar el estado del mismo.
Evento Nuevo Pedido: Se encarga de enviar, a la parte administrativa que maneja los pedidos, el nuevo pedido generado por parte de un cliente. De esta manera, la sección de "Pedidos a Preparar", tendrá la información de los pedidos recientemente realizados en tiempo real. El constructor del evento establece los parametros que seran transmitidos y tambien se define el canal por el cual sera enviada la información con el método "broadcastOn" como muestra la figura 0.14.

```php
<?php

namespace App\Events;

use Illuminate\Broadcasting\Channel;
/..
class NuevoPedido implements ShouldBroadcast
{
use Dispatchable, InteractsWithSockets, SerializesModels;

    public $pedido;
    public $detalles;
    public function __construct($pedido, $detalles)
    {
    $this->pedido = $pedido;
        $this->detalles = $detalles;
    }

public function broadcastOn()
    {
        return new Channel('pedidos');
    }
}
```

Figura 0.14: Evento “NuevoPedido”
Elaborado por: Juan Mesías

Evento Pedido Preparado: Es el encargado de enviar la modificación del estado del pedido el cual cambiará a "preparado". De la misma manera, los demás eventos son los encargados de modificar los diferentes estados del pedido en la aplicación.

```
1        <?php
2
3        namespace App\Events;
4
5        use Illuminate\Broadcasting\Channel;
6        //.
7
8        class PedidoPreparado implements ShouldBroadcast
9        {
10       use Dispatchable, InteractsWithSockets, SerializesModels;
11
12           public $pedido;
13           public $detalles;
14
15           public function __construct($pedido, $detalles)
16       {
17               $this->pedido = $pedido;
18               $this->detalles = $detalles;
19       }
20
21       public function broadcastOn()
22           {
23               return new Channel('pedidoPreparado');
24       }
25       }
```

Figura 0.15: Evento "PedidoPreparado"
Elaborado por: Juan Mesías

Laravel Notifications es un paquete propio del framework el cual se lo puede encontrar en la documentación oficial. Los canales de notificaciones que proporciona Laravel son varios; entre ellos: correo electrónico, SMS, slack y por una serie de canales diferentes creados por la comunidad de desarrollo de Laravel. Para el proyecto se eligió crear una notificación mediante correo electrónico. Las notificaciones se encuentran en el directorio: "app/Notifications".

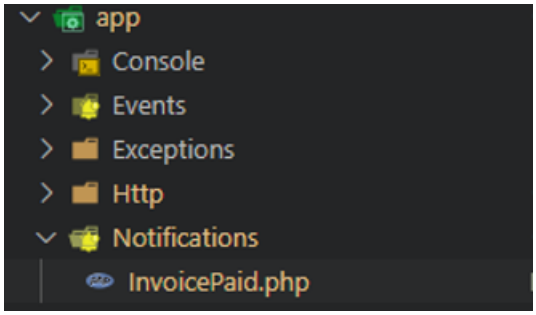


Figura 0.16: Notificaciones
Elaborado por: Juan Mesías

La configuración del mail del cual se va a enviar la notificación se la puede realizar en el archivo. env.

```
1        MAIL_DRIVER=smtp
2        MAIL_HOST=smtp.gmail.com
3        MAIL_PORT=587
4        MAIL_USERNAME=retomaos@gmail.com
5        MAIL_PASSWORD=*********
6        MAIL_FROM_NAME=reTomaOs
```

Figura 0.17: Archivo .env para datos de email
Elaborado por: Juan Mesías

Notificación InvoicePaid: Se encarga de enviar la notificación por correo electrónico al usuario indicado. El método indicado en la figura 0.18 declara su constructor solicitando como parametro el id del pedido del cual se notifica su estado, tambien se define el método "via" en este se declara el canal por el cual se enviara la notificaión en nuestro caso sera por email y en el método "toMail" se define la estructura del mensaje que se envia por correo electrónico.

```
class InvoicePaid extends Notification
{
    use Queueable;

    public $id;

    public function __construct($idPedido)
    {
        $this->id = $idPedido;
    }

    public function via($notifiable)
    {
        return ['mail'];
    }

    public function toMail($notifiable)
    {
    $restaurante = Restaurante::first();
        $pedido = Pedido::find($this->id);
        //$detalles = DB::table('detalle_pedido')->where('idPedido
            ','=',$this->id)->get();

        return (new MailMessage)
            ->subject('Pedido Listo')
            ->greeting($restaurante->nombre)
            ->line('Tu pedido esta listo para ser entregado!')
            ->line('N  pedido: '.$pedido->numeroPedido.'  '.'Fecha:
                '.$pedido->fechahoraPedido)
            ->line('Cliente: '.$pedido->nombreCliente.'  '.'CI/RUC:
                '.$pedido->numeroDocumento)
            ->line('Total: '.$pedido->totalPedido.' $')
            ->line('****************************************')
            ->action('Revisar','#')
            ->line('Gracias por preferirnos!');
    }
}
```

Figura 0.18: Clase 'InvoicePaid'
Elaborado por: Juan Mesías

Una vez realizadas las implementaciones de Eventos y Notificaciones se procedió a crear el controlador “PedidoController”, el cual hace uso de ”Events” y ”Notifications”.

Para crear un nuevo pedido se definió el método store el cual permite validar los datos necesarios para el almacenamiento del nuevo pedido y accionar el evento ”NuevoPedido” como se muestra en la figura 0.19.

```
1   public function store(Request $request)
2    {
3        $validation = Validator::make(
4            $request->all(),
5            [
6                'idUsuario' => 'required',
7                //..
8            ]
9        );
10        $pedido = new Pedido();
11        //..
12        $pedido->save();
13
14       //disparo el evento
15       event(new NuevoPedido($pedido, $detalles));
16
17       return response()->json(
18           [
19               'pedido' => $pedido,
20               'detalles' => $detalles,
21               'HttpResponse' => [
22                   'tittle' => 'Correcto',
23                   'message' => 'Pedido generado!',
24                   'status' => 200,
25                   'statusText' => 'success',
26                   'ok' => true
27               ],
28           ],
29           201
30       );
31
32   }
```

Figura 0.19: Método “store”
Elaborado por: Juan Mesías

El metodo de la figura 0.20 devuelve la información de los pedidos que tienen un estado de "pedido", lo que se vera reflejado en la aplicación como pedidos para preparar.

```
1   public function getPedidosParaPreparar (){
2
3       $pedidos = DB::table('pedido')->where('estado','=','pedido')
            ->get();
4       $detalles = array();
5       foreach ($pedidos as $pedido) {
6           $detalleTemporal = DB::table('detalle_pedido')->where('
                idPedido','=',$pedido-
                >id)->get();
7           foreach ($detalleTemporal as $item) {
8               array_push($detalles, $item);
9           }
10      }
11
12      return response()->json(
13          [
14              'pedidos' => $pedidos,
15              'detalles' => $detalles,
16              'HttpResponse' => [
17                  'status' => 200,
18                  'statusText' => 'OK',
19                  'ok' => true
20              ]
21          ],
22          201
23      );
24   }
```

Figura 0.20: Método "getPedidosParaPreparar"
Elaborado por: Juan Mesías

El metodo de la figura 0.21 devuelve la información de los pedidos que tienen un estado de "preparado", lo que se vera reflejado en la aplicación como pedidos para entregar.

```
1   public function getPedidosParaEntregar (){
2
3         $pedidos = DB::table('pedido')->where('estado','=','
              preparado')->get();
4         $detalles = array();
5         //..
6         return response()->json(
7               [
8                     'pedidos' => $pedidos,
9                     'detalles' => $detalles,
10                    'HttpResponse' => [
11                          'status' => 200,
12                          'statusText' => 'OK',
13                          'ok' => true
14                    ]
15              ],
16              201
17        );
18    }
```

Figura 0.21: Método "getPedidosParaEntregar"
Elaborado por: Juan Mesías

Para que el usuario tenga la posibilidad de consultar todos sus pedidos y los detalles de cada uno de implemento el método de la figura 0.22.

```
1   public function getPedidosByUsuario($id){
2         $pedidos = Pedido::where('idUsuario', $id)->orderBy('
              fechahoraPedido', 'desc')->get();
3         //..
4         return response()->json([
5                   'pedidos' => $pedidos,
6                   'detalles' => $detalles,
7                   'HttpResponse' => [
8                         'status' => 200,
9                         'statusText' => 'OK',
10                        'ok' => true
11                  ]],201);
12    }
```

Figura 0.22: Método "getPedidosByUsuario"
Elaborado por: Juan Mesías

Para la consulta de todos los pedidos y sus detalles se implemento el método getPedidosyDetallesGeneral.

```
1   public function getPedidosyDetallesGeneral (){
2         $pedidos = Pedido::all();
3         $detalles = DetallePedido::all();
4         return response()->json(
5             [
6                 'pedidos' => $pedidos,
7                 'detalles' => $detalles,
8                 'HttpResponse' => [
9                     'status' => 200,
10                    'statusText' => 'OK',
11                    'ok' => true
12                ]
13            ],
14            201
15        );
16    }
```

Figura 0.23: Método "getPedidosByUsuario"
Elaborado por: Juan Mesías

En la figura 0.24 se implement la actualización de estado del pedido consecuentemente es el encargado de enviar la notificación al usuario y accionar uno de los eventos, estas dos acciones se realizan en dependencia al estado enviado para la actualización.

```
1       public function actualizarEstadoPedido(Request $request, $id)
           {
2       //..
3        if($request->estado == 'preparado'){
4            event(new PedidoPreparado($pedido, $detalles));
5            $user = User::find($pedido->idUsuario);
6            $user->notify(new InvoicePaid($pedido->id));
7        } else if($request->estado == 'cancelado'){
8            event(new PedidoCancelado($pedido, $detalles));
9        } else if($request->estado == 'entregado'){
10           event(new PedidoEntregado($pedido, $detalles));
11       }
12       return response()->json(
13           //..
14           );
15       }
```

Figura 0.24: Método "actualizarEstadoPedido"
Elaborado por: Juan Mesías

Para la publicación de las rutas y tener acceso a las mismas Laravel contiene la herramienta de Routing que es definida en base a los controladores creados y a los métodos existentes dentro de los mismos, se debe especificar el método de acceso GET, POST, PUT, DELETE. El archivo se encuentra en el directorio "router/api.php".

```
1   Route::get('datosFacturacion/{id}', '
        DatosFacturacionController@show');
2   Route::post('datosFacturacion', '
        DatosFacturacionController@store');
3   Route::post('usuario/login', 'UserController@authenticate');
4   Route::post('usuario/logout', 'UserController@logout');
5   Route::post('usuario/registrar', 'UserController@register');
6   Route::get('usuarios', 'UserController@index');
7   Route::put('usuarios/{id}', 'UserController@actualizarUsuario');
8   Route::get('usuariosById', 'UserController@getUserByid');
9   Route::get('restaurante', 'RestauranteController@index');
10  Route::post('restaurante/update/{id}', '
        RestauranteController@update');
11  Route::get('categorias', 'CategoriaController@index');
12  Route::post('categorias', 'CategoriaController@store');
13  Route::put('categorias/{id}', 'CategoriaController@update');
14  Route::delete('categorias/{id}', 'CategoriaController@destroy');
15  Route::get('productos', 'ProductoController@index');
16  Route::post('productos/store', 'ProductoController@store');
17  Route::post('productos/update/{id}', 'ProductoController@update'
        );
18  Route::delete('productos/{id}', 'ProductoController@destroy');
19  Route::get('mesas', 'MesaController@index');
20  Route::post('mesas', 'MesaController@store');
21  Route::put('mesas/{id}', 'MesaController@update');
22  Route::delete('mesas/{id}', 'MesaController@destroy');
23  Route::post('pedidos', 'PedidoController@store');
24  Route::get('pedidosPreparar', '
        PedidoController@getPedidosParaPreparar');
25  Route::get('pedidosEntregar', '
        PedidoController@getPedidosParaEntregar');
26  Route::get('pedidosByUsuario/{id}', '
        PedidoController@getPedidosByUsuario');
27  Route::get('pedidosGeneral', '
        PedidoController@getPedidosyDetallesGeneral');
28  Route::put('actulizarPedido/{id}','
        PedidoController@actualizarEstadoPedido');
```

Figura 0.25: 'Puntos de consulta'
Elaborado por: Juan Mesías