UNIVERSIDAD POLITÉCNICA SALESIANA

SEDE CUENCA

CARRERA DE INGENIERÍA DE SISTEMAS

DESARROLLO E IMPLEMENTACIÓN DE UNA APLICACIÓN MÓVIL PARA EL CONTROL DE RUTAS DE MOTORIZADOS MONITOREADOS EN TIEMPO REAL

Trabajo de titulación previo a la obtención del título de Ingeniero de Sistemas

AUTORES: JUAN FERNANDO CÓRDOVA ARÉVALO
TELMO SEBASTIÁN ROCANO ORTEGA
TUTOR: ING. MIGUEL ARTURO ARCOS ARGUDO, PHD.

Cuenca - Ecuador
2022

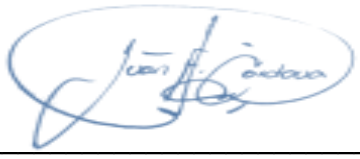
CERTIFICADO DE RESPONSABILIDAD Y AUTORÍA DEL TRABAJO DE TITULACIÓN

Nosotros, Juan Fernando Córdova Arévalo con documento de identificación N° 0105822209 y Telmo Sebastián Rocano Ortega con documento de identificación N° 0105093884; manifestamos que:

Somos los autores y responsables del presente trabajo; y, autorizo a que sin fines de lucro la Universidad Politécnica Salesiana pueda usar, difundir, reproducir o publicar de manera total o parcial el presente trabajo de titulación.

Cuenca, 11 de marzo del 2022

Atentamente,



Juan Fernando Córdova Arévalo
0105822209

Telmo Sebastián Rocano Ortega
0105093884

CERTIFICADO DE CESIÓN DE DERECHOS DE AUTOR DEL TRABAJO DE TITULACIÓN A LA UNIVERSIDAD POLITÉCNICA SALESIANA

Nosotros, Juan Fernando Córdova Arévalo con documento de identificación N° 0105822209 y Telmo Sebastián Rocano Ortega con documento de identificación N° 0105093884, expresamos nuestra voluntad y por medio del presente documento cedemos a la Universidad Politécnica Salesiana la titularidad sobre los derechos patrimoniales en virtud de que somos autores del Proyecto técnico: "Desarrollo e implementación de una aplicación móvil para el control de rutas de motorizados monitoreados en tiempo real", el cual ha sido desarrollado para optar por el título de: Ingeniero de Sistemas, en la Universidad Politécnica Salesiana, quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente.

En concordancia con lo manifestado, suscribimos este documento en el momento que hacemos la entrega del trabajo final en formato digital a la Biblioteca de la Universidad Politécnica Salesiana.

Cuenca, 11 de marzo del 2022

Atentamente,

Juan Fernando Córdova Arévalo
0105822209

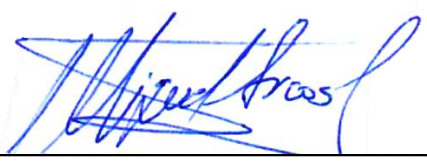
Telmo Sebastián Rocano Ortega
0105093884

CERTIFICADO DE DIRECCIÓN DEL TRABAJO DE TITULACIÓN

Yo, Miguel Arturo Arcos Argudo con documento de identificación N° 0103958021, docente de la Universidad Politécnica Salesiana, declaro que bajo mi tutoría fue desarrollado el trabajo de titulación: DESARROLLO E IMPLEMENTACIÓN DE UNA APLICACIÓN MÓVIL PARA EL CONTROL DE RUTAS DE MOTORIZADOS MONITOREADOS EN TIEMPO REAL, realizado por Juan Fernando Córdova Arévalo con documento de identificación N° 0105822209 y por Telmo Sebastián Rocano Ortega con documento de identificación N° 0105093884, obteniendo como resultado final el trabajo de titulación bajo la opción Proyecto técnico que cumple con todos los requisitos determinados por la Universidad Politécnica Salesiana.

Cuenca, 11 de marzo del 2022

Atentamente,



Ing. Miguel Arturo Arcos Argudo, PhD
0103958021

DEDICATORIA

Este proyecto le dedico a Dios por darme la fortaleza de cumplir con mis metas, a mis padres Galo Daniel Cordova y Manuela Arevalo, por su amor incondicional y por qué día a día siempre lucharon, para darme la mejor educación, a mi esposa Anita por no dejarme vencer y apoyarme incondicionalmente y por su amor infinito que día a día me brinda, a mis hijas Rafaela y Renata, que ellas fueron y serán el motor principal que me permita seguir cumpliendo las metas que me proponga, a mi hermana Verónica y mi cuñado Armando que siempre estuvieron para apoyarme y aconsejarme, por ellos que son y serán mi gran alegría y bendición, Dios les bendiga siempre.

Juan Fernando Córdova Arévalo

Primeramente dedico este proyecto a Dios, quien me dio las fuerzas para cumplir con cada uno de mi propósitos, dedico este proyecto a mis padres Manuel Rocano y Carmita Ortega, por su esfuerzo que hicieron todo este tiempo, por su amor incondicional, les doy las gracias por estar conmigo en todo el transcurso de la carrera Universitaria, por no dejarme solo en los momentos difíciles y apoyarme en todo, por sus concejos y a no darme por vencido y darme las fuerzas necesarias para seguir y a no rendirme nunca.

También a mis Hermanos, por tenerme paciencia en los momentos difíciles y apoyarme cuando sentía que ya no podía más, les agradezco por darme motivos para seguir adelante. A mis sobrinas que día a día me sacan una sonrisa y me dan fuerzas y motivos de seguir esforzándome cada día para ser una mejor persona.

Telmo Sebastián Rocano Ortega

AGRADECIMIENTO

Un agradecimiento especial a la Universidad Politécnica Salesiana, y a sus distinguidos docentes por tan valiosa formación educativa que recibimos, en especial al Ing. Pablo Gallegos PhD. por enseñarnos que la vida laboral es un mundo diferente y prepararnos con temas que ahora nos están sirviendo, al Ing. Gabriel León PhD. por enseñarnos que nunca debemos dejar de aprender eh inspirarnos por investigar las nuevas tecnologías como él siempre lo hace, a nuestro tutor Ing. Miguel Arturo Arcos Argudo, PhD. por su paciencia y dedicación para guiarnos en nuestro trabajo, a nuestras familias, compañeros, amigos que estuvieron siempre con nosotros en el difícil camino hacia nuestra meta, con toda sinceridad un agradecimiento profundo a cada uno de ustedes por su colaboración que Dios les bendiga.

Telmo Sebastián Rocano Ortega
Juan Fernando Córdova Arévalo

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

RESUMEN

El objetivo del proyecto de titulación denominado “Desarrollo e implementación de una aplicación móvil para el control de rutas de motorizados monitoreados en tiempo real” consiste en apoyar a los establecimientos comerciales (por ejemplo: restaurantes, farmacias, etc.) que se han visto afectados por la pandemia la cual fue provocada por el SARS-Cov-2 durante los años 2020, 2021 y hasta la actualidad, los mismos que han buscado la manera más eficaz de solventar los pedidos que sus clientes realizan día a día. Además, los usuarios por temor a salir a las calles han visto la necesidad de tomar otras medidas de compra para adquirir en este caso los servicios de los proveedores. Por este motivo hemos desarrollado una aplicación en Flutter para móviles tanto Android como IOS, esto para que ayude al establecimiento comercial a: receptar y despachar los pedidos a domicilio, y a la misma vez que el cliente pueda visualizar en tiempo real en donde se encuentra en repartidor que en este caso será el motorizado. Para el desarrollo de la aplicación se han utilizado las siguientes herramientas: Android Studio, Flutter, Node JS, PostgreSQL, entre otras. Dichas herramientas permiten crear una aplicación para los diferentes sistemas operativos de cada móvil. Así mismo, se ha informado al dueño del negocio, a los clientes y a los motorizados sobre los beneficios y alcance de la aplicación. Se han definido los requerimientos necesarios, la metodología de desarrollo de la aplicación, la etapa de pruebas y la validación de la aplicación. Posteriormente, se presentan algunos resultados: la recepción de varios pedidos los cuales deben ser claros y sobre todo precisos en el seguimiento, el incremento que existe para las solicitudes de servicio a domicilio, el monitoreo del propio repartidor (motorizado) y la geolocalización que permite visualizar la ubicación

en tiempo real del repartidor. La aplicación permite definir diferentes roles que básicamente consisten en: establecimiento, repartidor y cliente, los mismos que podrán realizar diferentes acciones.

- Palabras Claves: Flutter, entrega a domicilio, Delivery, motorizado, aplicación móvil, pandemia, geolocalización.

I. Abstract

The objective of the titling project called "Development and implementation of a mobile application for the control of motorized routes monitored in real-time" is to support commercial establishments (e.g., restaurants, pharmacies, etc.) that have been affected by the pandemic caused by SARS-Cov-2 during the years 2020, 2021 and until today, the same that have sought the most effective way to solve the orders that their customers make every day. In addition, users for fear of going out to the streets have been forced to take alternative ways of purchase to acquire in this case the services of suppliers. For this reason, we have developed an application in Flutter for both Android and IOS mobiles, this help the commercial establishment

to: receive and dispatch home orders, and at the same time that the customer can view in real-time where the delivery man is, in this case, will be the motorized. The following framworks have been used for the development of the application: Android Studio, Flutter, Node JS, PostgreSQL, among others. These frameworks allow you to create an application for the different operating systems of each mobile. Likewise, the business owner, customers, and motorized people have been informed about the benefits and scope of the application. The requirements, the application development methodology, the testing stage, and the validation of the application have been defined. Subsequently, some results are presented: the reception of multiple orders, clear and precise orders, the increase that exists for delivery service requests, the monitoring of the delivery person (motorized), and the geolocation that allows visualizing the delivery person's location in real-time. The application allows defining different roles that consist of establishment, delivery person, and customer, who can perform different actions.

- Keys words: Flutter, delivery service, motorized, mobile application, pandemic, geolocation.

INTRODUCCIÓN

Durante los años 2020 y 2021 el mundo ha sido testigo de los cambios suscitados en su entorno provocados por la pandemia. Con estos cambios, empresas como restaurantes (y cualquier establecimiento comercial), se han visto en la obligación de tomar otros medios de atención al cliente para así asegurar la salud del consumidor (usuario final) y su propia subsistencia. De la misma manera, los clientes buscan nuevas formas de proveerse de productos y que les sean entregados en las puertas de su domicilio, sin necesidad de salir de casa, lo cual supone también una manera de aumentar su confort. Por ello el siguiente proyecto de titulación consiste en desarrollar e implementar una aplicación móvil para el control de rutas de motorizados monitoreados en tiempo real, especialmente dirigido a restaurantes quienes entregarán sus productos a sus clientes mediante un motorizado.

Con la siguiente propuesta se busca conocer la necesidad y factibilidad de crear una App que permita la automatización de los procesos para la gestión de los pedidos que realizan los clientes a los establecimientos de comida, ofreciendo el servicio de entregas mediante motorizados quienes son los encargados de llevar el producto hasta la puerta del consumidor.

Los establecimientos de comida buscan áreas, métodos y nuevas oportunidades para incursionar en el mundo de la digitalización y atraer a sus clientes, y al mismo tiempo ofrecerles una alternativa de continuar ofreciendo su servicio minimizando el riesgo de contagio de COVID. Una de estas alternativas es el uso de aplicaciones móviles que pongan a disposición de los clientes una gran variedad de opciones para comer, precios y que sean entregados a domicilio.

1.1 APP’s de entrega a domicilio

Hoy por hoy existen muchas aplicaciones que se han creado para el servicio de entregas a domicilio, a continuación, se detallan algunas de ellas

Uber Eats: es una plataforma de pedidos a domicilio, es muy popular y conocido del mundo, esta plataforma esta activa en cientos de países alrededor del mundo, tiene su funcionamiento de la siguiente manera; los restaurantes que se registran o que están registrados en la plataforma solo tienen que preparar los pedidos, lo demás lo realiza la plataforma. El cliente puede o hace un pedido al Restaurante a través de la aplicación, por lo tanto, el restaurante acepta y prepara el pedido, por lo cual el Uber encargado en el repartir lo recoge y lo entrega físicamente al cliente.
En comisiones y pagos; Uber Eats, realiza sus comisiones para cada pedido. El porcentaje de comisión que el restaurante debe de pagar a la aplicación va a variar de acuerdo con la base de su contrato, a la cantidad de locales y/o ubicaciones, entre otras. Adicionalmente Uber cobra

una comisión por el servicio realizado, que les permite ayudar a cubrir los costos de soporte, marketing y comisiones por tarjetas de crédito, etc.

PedidosYa; Es una tecnología líder en Q-Commerce y Delivery en Latinoamérica. Es una plataforma de pedidos digital muy fácil de utilizarla, practica y no tiene costos adicionales esta plataforma permite a sus usuarios elegir una gran variedad de opciones disponible y realizar a su pedido a través del sitio web y también mediante aplicaciones para Android y iOS. Esta plataforma cumple los pedidos de los usuarios que realicen sus diferentes pedidos de una forma rápida y sencilla de utilizar de esta manera se satisface las duras expectativas de los usuarios. Hoy por hoy es una de las plataformas de pedidos con mayor experiencia y alcance geográfico en Latinoamérica y busca el liderazgo en cada mercado que están presentes. Uno de sus beneficios que tiene esta plataforma, es realizar pedidos de forma práctica, segura y sin consto adicional, además encontrar miles de restaurante al alcance de la mano, evita llamadas telefónicas y líneas ocupadas, entre otras.

Móvil foods; Es una plataforma para pedir comida a domicilio desde cualquier restaurante favorito y que lo hace al menos de 30 minutos, los pedidos son recibidos de manera eficaz y simple, algunas de las ventajas de estas apps son: El precio de envió de su pedido, además el restaurante notifica el tiempo de llegada del pedido con el tiempo de entrega incluido, puede tener un plato favorito y calificarlo.

KFC; Cuenta con una aplicación disponible para teléfonos Android y iOS, de esta manera el cliente ya puede realizar el pedido que desee, y a la misma vez, se puede obtener cupones de descuento. Esta aplicación se permite hacer los pedidos a domicilio u ordenarlos para que sean recogidos en el local ha sido seleccionado.

Domino's Pizza; Tiene su aplicación en Latinoamérica, Su cobertura incluye países como Ecuador, Colombia, Panamá y Perú. Con esta app se puede visualizar diferentes promociones y permite elegir si desea hacer la orden a domicilio, o en si pagar para poder recogerlo. Es decir, el consumidor hace el pedido y selecciona el local donde puede recogerlo.

Picker; Es una aplicación Delivery, fundado en Ecuador en la que se puede pedir comidas de restaurantes, pero también permite realizar encomiendas, hacer comprar en tiendas e incluso hacer compras en diferentes mercados municipales.

Estas son algunas de las aplicaciones que están vigentes en el Ecuador, y que relazan el servicio de entregas a domicilio.

Diferencia entre estas Aplicaciones, con la propuesta de trabajo realizado

A continuación, se detallan las diferencias que existe en la propuesta de trabajo con las demás aplicaciones que ya existen hoy por hoy en el mercado.

- La aplicación nos permite crear cada una de las ordenes que el restaurante tiene en ese momento disponible.
- Cuenta con tres estados que le permite saber tanto al Restaurante como al cliente el estado del pedido, estados son, despachado, en camino y entregado, lo que permite ser visualizado desde la aplicación
- El cliente sabrá la ubicación exacta de donde se encentra el repartidor, es decir si el repartidor toma otra ruta, deferente a la que se marcó primero, el cliente sabrá por donde está el repartidor que se le asigno para la entrega y el tiempo exacto.
- Si el repartidor se encuentra lejos, es decir, no se encuentra cerca del restaurante o está demasiado lejos no podrá realizar la entrega, por lo que el restaurante debe de buscar a otro repartidor que este más cerca posible.
- Se le establecerá una ruta donde haya menor tráfico al repartidor para hacer la entrega.

Existen algunas aplicaciones que han tenido la misma idea de implementar el servicio de entregas a domicilio, han fracasado, tal es el caso de empresas como ***Glovo, MEGABIT y Domicilios***.

Glovo, fracaso por malos resultados económicos, por no poder retener el market share (Cuota de mercado) que lo habían logrado, lo que hizo perder cuando llegaron más competidores al mercado, estas empresas que hicieron que Glovo fracasara fueron PedidosYa, Rappi UberEats y entre otras.

La aplicación de **Glovo** realizaba entregas de comida, bebidas alcohólicas, medicamentos, regalos, y también realiza cualquier tipo de trámite.

Domicilios.com esta aplicación dejo de funcionar con la aparición de IFood luego de haber adquirido la compañía. Domicilios siguió en 2019, cuando Glovo vendió sus operaciones en Perú y Ecuador, hasta 2020. La plataforma IFood brasileña anuncio su intención de hacerse dueño de domicilios y trato de fortalecer su arremetido contra Rappi. Con la aparición de Rappi, domicilios.com dejo de funcionar.

1.2 El Comercio Electrónico

Los negocios a nivel general, se alternan en las formas de hacer negocios. El cambio invariable de la exceptiva de los usuarios, como el aumento de la capacidad. A vista de ello, el comercio mundial está cambiando su forma de actuar y se ve en la obligación de adaptarse a estos nuevos cambios.

Los servicios comerciales se realizan de diferentes maneras, por ejemplo, el comercio dentro de la empresa, las diferentes interacciones y, últimamente, comercio electrónico a través de la red internet, a esto se debe su crecimiento en gran medida.

1.3 Comercio Electrónico durante la pandemia SARS COV 2

Muchas empresas debido a la pandemia COVID-19 cambiaron su forma de hacer negocio y en sí, tuvieron que adaptarse a los cambios que les han obligado a socializar con la tecnologia.

"Los cambios que ha sufrido la humanidad en este tiempo debido a la pandemia han hecho prácticamente la forma obligatoria para poder mitigar y afrontar la situación actual por parte del sector empresarial, los compradores se han visto impactados al momento de adquirir un bien tangible o intangible. Estos cambios, a los cuales han sido sometido la micro, pequeñas, medianas y grandes empresas, se han visto apoyadas y beneficiados de medios y herramientas tecnológicas digitales a través de internet, esto para poder promocionar y vender sus productos y servicios, impulsando a un gran segmento del mercado a realizar compras y ventas de forma online, siendo el comercio electrónico el gran protagonista ante la situación actual, el cual últimamente ha tenido un gran realce e impacto tanto para el empresario, como para el consumidor, estas herramientas se han convertido un gran apoyo ante el confinamiento que se le ha exigido a la población. Con eso se logra salvaguardar su salud y de las demás, de esta formar se logra erradicar el contagio; con la nueva enfermedad del COVID-19." (Barría, F. C., de Tyler, C. R., & Jiménez, T. G, 2021)

Según la Organización Mundial de la Salud (2021) "El COVID-19, es la enfermedad infecciosa causada por el coronavirus que se ha descubierto más recientemente. Tanto este muevo virus como la enfermedad que provoca eran desconocidos antes de que estallara el brote de Wuhan () China) en diciembre de 2011, Actualmente, la COVID-19 es una pandemia que ha afectado a todo el mundo"

Siempre que una aplicación es lanzada al mercado debe tener un grado de aceptación con los usuarios, esa aceptación dependerá en gran medida de las diferentes características que cada usuario al utilizar dicha aplicación considere importante. Una de las principales características desde la Ingeniería del Software debe ser siempre la calidad. Para los desarrolladores de software lo que deben tener en cuenta es saber cómo medir y como realizar las aplicaciones para satisfacer a los usuarios. (Enriquez, J. G., & Casas, S. I., 2014)

"En la actualidad la tecnología móvil ha cambiado la forma en la que vemos las cosas, la forma en la que vivimos y mucho más en la forma en la que trabajamos y sobre todo la forma en la que nos comunicamos, esto afecta a todas las esferas de nuestra vida. Como comentó en

una ocasión *Benedict Evan*, ejecutivo de la consultora *Andreessen/Horowitz* (un observador indispensable del cambio digital a través de los medios de comunicación), los móviles se están comiendo al mundo. Oficialmente, existe una gran cantidad de dispositivos móviles en el mundo, con diferentes sistemas operativos que personas en el mundo. Según el *Global System Mobile Association* (GSMA sus siglas en ingles), los móviles están a punto de llegarse a los 7 422 millones de conexiones diarias que se realizan día a día, mientras que el censo de población en todo el mundo es de 7 228 millones. Sin embargo, en el año 2014 por primera vez supero el tiempo y el número de acceso a la web desde los dispositivos móviles respecto a los accesos a Internet efectuados desde las PC's de escritorio, hasta el punto de que no hay distinción entre el futuro de la telefonía móvil y el futuro de la tecnología. Son lo mismo; en el plazo de unos años el adjetivo móvil será prescindible cuando hablamos de tecnologías, porque la mayoría de las tecnologías en su mayoría serán dispositivos móviles" (Arevalo & Canelo, 2017).

"La geolocalización se ha tornado una de las herramientas más monopolizadas por los geógrafos, lo que les permite ubicar a las personas u objetos en el espacio mediante sus coordenadas y que ha a partir de la aparición de Internet ha cobrado una nueva dimensión y de los dispositivos móviles. Frente a esto se ha desarrollado el fenómeno de colaborar con la información desde cada lugar e individuo en los medios sociales. A la unión de estos aspectos se le ha denominado **SoLoMo**, acrónimo de ***social, local y móvil,*** donde la geolocalización sirve como una herramienta de comunicación entre el mundo digital y físico, desde lo global a lo local, desde el lugar a la nube" (López, 2015).

Por medio de la implementación de nuevos tipos de tecnologías en los restaurantes y establecimientos comerciales que se encuentran en las diferentes ciudades del país, será posible presentar mejores perfiles publicitarios, para la obtención de ventajosos resultados como es: el incremento de productividad y así mismo disminuir los costos de publicidad, esto gracias a novedosas ofertas que presentan las aplicaciones que son descargadas en los dispositivos móviles y sobre todo fortalecer la comodidad y facilidad de los usuarios a quienes se orientan (García & Vélez ,2014).

Las aplicaciones móviles han explotado sus recursos para poder obtener información sobre la ubicación, los mismos que se ven plasmados en documentos cartográficos que tiene diferentes orígenes y que son denominados como sistema de coordenadas, estos sirven para mostrar como resultado los puntos específicos para referenciar un determinado lugar.

Para el desarrollo del proyecto se utilizará la sistemática Scrum, este sistema prácticamente consiste en procesos, conjunto de tareas, con el objetivo del trabajo en equipo.

Empleando este sistema trabajo, nuestro objetivo es alcanzar resultados insuperables de nuestro proyecto. Las prácticas con la metodología Scrum se retroalimentan unas con otras y tiene su origen en un estudio de cómo coordinar con los equipos para ser competitivos.

PROBLEMA

2.1 Definición del problema

El aislamiento social provocado por el confinamiento y el temor al contagio del virus SARS-Cov-2 ha implicado que todo el mundo entre en una crisis financiera perjudicando las ventas y los ingresos de todo tipo de negocios, especialmente a la pequeña y mediana empresa, ya que sus clientes han tenido que evitar salir de casa. Como consecuencia de este fenómeno se ha evidenciado que las ventas se han realizado utilizando medios digitales que, hasta el momento, han sido poco explotados. Uno de esos medios son las aplicaciones móviles que ofrecen la posibilidad de que un comercio se registre en ella y ofrezca sus productos a sus clientes con un proceso muy simple: la venta se realiza en el entorno web y el producto se entrega utilizando un medio de transporte físico. Uno de los medios de transporte más eficaz y económicos son las motocicletas, ya que consumen poco combustible y son fáciles de estacionar, además de ser fáciles de adquirir en comparación a un vehículo. Sin embargo, existen problemas que aún se pueden resolver de manera más eficiente, por ejemplo: la insatisfacción que siente un cliente que no tiene su pedido a tiempo, lo cual causa malestar y por ende la negativa de volver a comprar en el mismo sitio web; es por ello que resulta interesante que un motorizado que realiza las entregas tenga un control de ruta en tiempo real, de esta manera, tanto el cliente como el dueño del negocio estará seguro de que la entrega se realice en los tiempos establecidos ayudando en gran medida a negocios. El presente proyecto contribuye a la solución de esta problemática mediante una aplicación informática que permita a un cliente seleccionar un proveedor, realizar la compra, asignar la entrega a un motorizado y monitorizar el recorrido del repartidor hasta llegar a su destino. Si bien es cierto que la aplicación que se ha desarrollado puede ser utilizada por un proveedor de cualquier bien, ha sido probada de manera especial en negocios gastronómicos (restaurantes, soda bar, picanterías, etc.) ya que constituyen el tipo de negocio que más utiliza este tipo de herramientas para realizar sus ventas en medio de la pandemia.

OBJETIVOS

3.1 Objetivo general

Desarrollar una App Móvil que permite al usuario realizar compras en línea y asigne un motorizado que se encargue de recoger y entregar el pedido monitorizando en tiempo real la ruta más cercana.

3.2 Objetivos Específicos

- Utilizar herramientas de desarrollo que permitan implementar geolocalización para visualizar la ruta que va a utilizar el motorizado.
- Contribuir con una posible solución a los usuarios que realizan compras en línea.
- Contribuir a mantener el distanciamiento social mediante el uso de la App que permitirá a los usuarios efectuar sus compras, sin salir de casa en tiempos de pandemia.
- Utilizar herramientas que permitan realizar transacciones en una BD mediante una Aplicación Web.

REVISIÓN DE LA LITERATURA O FUNDAMENTOS TEÓRICOS

Debido a la pandemia del COVID 19, el miedo de las personas de salir a la calle para realizar sus actividades normales ha ido aumentando día a día, es por ello que muchos restaurantes tuvieron que adaptarse a la necesidad de la gente para no perder sus ingresos buscando repartidores que puedan ayudar a entregar sus productos. Si bien esta solución ha causado un mayor ingreso para los dueños de restaurantes también ha causado el malestar y sobre todo la pérdida de interés por parte de los usuarios que realizan los pedidos, el motivo radica en la falta de control y monitorización para las personas en este caso los motorizados que realizan las entregas a domicilio. La utilidad de esta implementación está principalmente en el control de los motorizados con la finalidad de mejorar las entregas en cuanto al establecimiento de rutas y tiempos, de esta manera se logrará mejorar el interés del usuario por realizar los diferentes pedidos y por ende mejorar el rendimiento de entregas satisfaciendo la necesidad del dueño del negocio de mantener sus ventas y la del cliente de adquirir el producto que desea.

4.1 Estado del arte y la práctica

4.1.1 Comercio electrónico

Es cualquier forma de transacción comercial donde las partes interactúan electrónicamente, por lo tanto, el comercio electrónico es el uso de las tecnologías de las telecomunicaciones y de la informática.

En la figura 4.1.2.1. Se puede visualizar que las operaciones del comercio electrónico las cuales giran en torno al usuario, lo primero que se debe hacer es atraer la atención del cliente al sitio web o aplicación, mediante la publicidad o por promociones en internet, ya que el cliente conoce el sitio lo que se debe de hacer es interactuar con él.

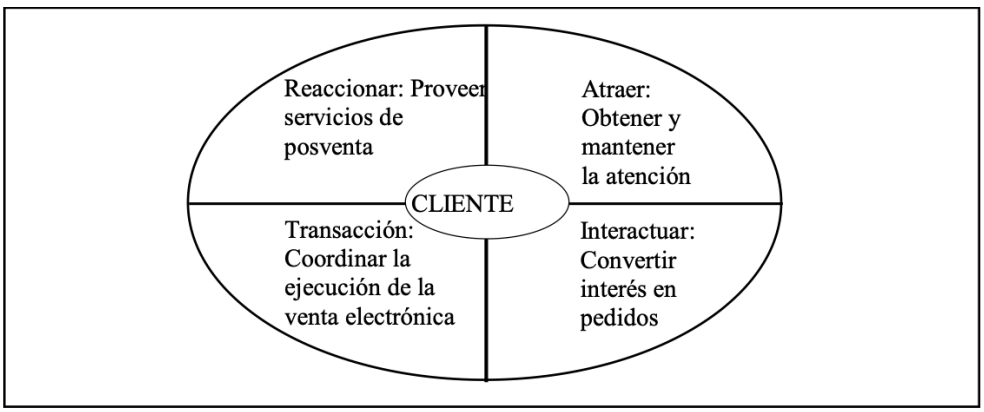


Figura 4.1.2.1 Ciclo del Comercio Electrónico.

4.1.2 Diferencia entre e-businnes y e-commerce.

Como se mencionó anteriormente existen aplicaciones móviles tanto nacionales e internaciones que realizan entregas a domicilio. Empresas como lo son UberEats, PedidosYa, Mobil Foods, KFC, Dominós, entre otras, de las cuales alguna de esas empresas no ha tenido éxitos tanto nacionalmente como internacional. En Ecuador varios negocios están usando multoles medios digitales para promocionar sus productos tales como: WhatsApp, Telegram, Redes sociales y Aplicaciones móviles, para tomar pedidos de los clientes, esto a fin de apegarse a las órdenes dadas por el gobierno Nacional a cuenca de la Emergencia sanitaria que esto viviendo nuestro país y todo el mundo como lo es el Coronavirus.

En los últimos años se han visto el incrementando el uso y la variedad de aplicaciones que realizan entregas de comida a domicilio en Ecuador.

Uno de los factores que ha permitido que las aplicaciones de entrega a domicilio sigan en crecimiento ha sido la pandemia del COVID-19, ya que muchas personas se han apegado de las recomendaciones de quedarse en casa, para así evitar exponerse al virus la base de los usuarios aumento hasta en un 300% durante los meses más críticos que vivió el Ecuador.

En el mercado nacional e internacional no solo están aplicaciones en las que se ofrecen las opciones de diferentes locales de comida, sino que también has propios restaurantes que han desarrollado su propia aplicación.

Si bien la pandemia aun no acaba, pero la mayoría de las personas lo han tomado como una enfermedad a la cual toca aprender a vivir con eso, pero existen muchas personas o empresas que aun realizan sus labores desde la comodidad de sus hogares, por lo que, al momento de no poder salir de sus casas, también optan por la opción de realizar pedidos a través de diferentes aplicaciones móviles que existen.

Una App o aplicación móvil, es un programa o software informático que está actualmente diseñado para que funcione en teléfonos o dispositivos inteligentes (smartphones), tables.

“No obstante, inicialmente las Apps fueron pensadas como herramientas de trabajo y brindaba información general, como el calendario o el correo electrónico, ha tenido un rápido incremento de su variabilidad y su desarrollo, con el progreso de nuevas tecnologías, y con los avances tecnológicos, el porcentaje de la población que tiene teléfonos inteligentes u otros dispositivos capaces de soportar estas aplicaciones es mucho mayor, comparado con anteriores años. Etas aplicaciones o Apps, pueden ser de pago o gratuitas” (Arévalo & Canelo, 2017).

A continuación, se describen brevemente los conceptos y herramientas que se manejarán en el desarrollo de este proyecto, con la finalidad de proporcionar la información fundamental que facilitarán la comprensión del presente trabajo.

4.1.3 Aplicaciones Móviles.

Las Aplicaciones Móviles (*App*) son herramientas digitales que se ejecutan en dispositivos móviles. El usuario puede obtener múltiples beneficios con su funcionalidad, sin importar el lugar en que este se encuentre (Tubón, 2020).

Las aplicaciones móviles se apartan de los sistemas de software integrado, a diferencia de las aplicaciones que son desarrolladas o diseñadas para computadoras de escritorio. Las aplicaciones móviles proporcionaran una funcionalidad limitada y aislada. Por ejemplo, puede ser, un navegador web móvil, un juego o una calculadora (Herazo, 2020).

4.1.4 Características de las Apps.

Para que una App sea rentable, se deberá de considerar ciertas características:

- **Interfaz Simple.** Debe de ser amigable en cuanto la presentación o el entorno con cualquier usuario, que lo use sin problema o, en ciertas ocasiones, ser capacitado.

- **Seguridad.** Proteger la información y a su vez mantener la privacidad de los usuarios que la usen.

- **Actualizaciones Periódicas.** La App deberá de ser actualizada cada cierto tiempo, esto con el fin de corregir errores o realizar mejoras que permitan cumplir con las necesidades y requerimientos de quienes la usan.

- **Servicios Web.** La World Wide Web Consortium la define como "...un sistema de software diseñado para soportar interacción interoperable máquina a máquina sobre una red. Estos, se diferencian por tener una interfaz descrita en un formato procesable por una máquina (específicamente WSDL). Otros sistemas interactúan con el servicio web en una manera prescrita por su descripción usando mensajes SOAP, típicamente enviados usando HTTP con una serialización XML en relación con otros estándares relacionados con la web" (Morales, 2010).

4.1.5 Sistemas Operativos Móviles.

- **Android. "**Plataforma software y sistema operativo basado en Linux, para dispositivos móviles, No es muy habitual que usen este sistema operativo, tablets, incluso PC 's. Los sistemas operativos Android permite programar aplicaciones en un ambiente de trabajo (framework) de Java sobre una máquina virtual Dalvik (variación de la máquina de Java con compilación en tiempo de ejecución). Además, lo que lo diferencia de otros sistemas operativos, es que

cualquier persona que sepa programar puede crear nuevas aplicaciones, *widgets*, o incluso, modificar el propio sistema operativo, dado que Android es de código libre, por lo que utilizando los conocimientos necesarios en lenguaje Java es muy fácil programar en esta plataforma" (Báez et al., 2019).

- **IOS. "**Lanzado y utilizado por Apple, iOS es un sistema operativo. Su nombre proviene de iPhone OS. Es decir, iPhone Operating System o Sistema Operativo de iPhone" (García, 2021).

4.1.6 Geolocalización.

"La geolocalización es una tecnología que permite encontrar un dispositivo sin importar el lugar en donde se encuentre, utilizando sus coordenadas geográficas: latitud y longitud. Estas coordenadas se pueden obtener mediante señales GPS, Wi-Fi, torres telefónicas y radiofrecuencia" (Del Médico, 2021).

- **Google Maps. "**Es una herramienta muy poderosa que proporciona mapas digitales de cualquier punto del planeta, en la que, además que permite de buscar deferentes direcciones o calcular itinerarios, hace posible ubicar lugares mediante marcas de posición o delimitar áreas; así, resulta un recurso de enorme potencial didáctico" (López, 2012).

4.1.7 Mercado de Aplicaciones (App Stores)

"A nivel mundial el mercado de aplicaciones móviles experimenta un excesivo crecimiento impulsado por la penetración masiva de dispositivos inteligentes, los desarrolladores y comercializadores de diferentes aplicaciones móviles se enfrentan a desafíos relacionados con la retención de usuarios, la monetización y el fraude publicitario" (López, 2019).

Con el incremento del acceso digital a través de teléfonos inteligentes y tablets, ha aumentado considerablemente las aplicaciones dentro del app market para diferentes situaciones y necesidades.

El teléfono celular dejó de ser un aparato que sólo recibe y hace llamadas telefónicas, convirtiéndose en un verdadero compartimiento de aplicaciones descargadas para otras soluciones de necesidades diversas de las personas (Rank MyApp, 2018).

- **App Store.** Es una tienda de aplicaciones destinada únicamente para dispositivos móviles Apple, estas aplicaciones estarán ejecutadas dentro del sistema operativo IOS.

Figura 4.1.7.1: Logo de App Store

Fuente: https://www.apple.com/v/app-store/b/images/overview/icon_appstore__ev0z770zyxoy_large_2x.png

- **Google Play.** Esta tienda en línea fue creada por Google e inicialmente se llamó Android Market, luego cambió su nombre para dar a conocer que no solo se venden aplicaciones móviles para Android sino para todo tipo de dispositivos y que los usuarios pueden navegar y descargar aplicaciones de pago y gratuitas (Quimbaya, 2014).

Figura 4.1.7.2: Google Play

Fuente: https://play.google.com/store?hl=es_EC&gl=US

4.1.8 Requisitos necesarios para publicar una App

A continuación, se describirán los requisitos que las tiendas de aplicaciones exigen para publicarlas:

Tabla 4.1.8.1: Requisitos de una App a publicar

Tienda / Requisitos	Google play	App Store
Cuenta de desarrollador	Si	Si
Costo por crear cuenta	25$	99$/anual
Versiones beta	No	No
Costo por publicar app	No	-
Costo por descargar	30%	30%
Formato	APK	IPA
Dispositivo con sistema operativo de la compañía	No	Si
Disponibilidad del país	Indefinida	Indefinida

Fuente: (Cando & Antony, 2020)

En la tabla 4.1.8.1 se puede apreciar que los requisitos varían de acuerdo a la tienda en la cual se va a publicar la App. Google Play es la más beneficiosa por un costo de $25. A su vez facilita publicar cualquier tipo de App de modo ilimitada y no necesita que el equipo sea una marca específica.

Se presentarán los principales conceptos necesarios para el desarrollo e implementación del proyecto.

4.2 Plataformas de desarrollo.

- **Framework Flutter. "**Es un Framework multidesarrollo, desarrollado por Google utilizando su propio lenguaje de programación llamado DART. Con Flutter se pueden desplegar aplicaciones Android y iOS y además con la versión dos del framework también se puede desarrollar software para diversas plataformas como Linux, Windows, web y Mac OS. A pesar de encontrarse en otras plataformas" (Moya, 2021).
  Para la ejecución de este proyecto se trabajará con una aplicación la cual será implementada para Android y iOS.

Figura 4.2.1: Logotipo de flutter

Fuente: https://flutter.dev

- **Android Studio.** "Es el entorno de desarrollo integrado (IDE) para el desarrollo de apps utilizado para Android y está basado en IntelliJ IDEA. Es un potente editor de códigos y las herramientas para desarrolladores de IntelliJ, Android Studio ofrece incluso más funciones que aumentan tu productividad cuando desarrollas apps para Android" (Developers, 2021)

Figura 4.2.2: Logo de Android Studio

Fuente: https://developer.android.com/studio

- **XCode.** El código X 13 es una tecnología que añade nuevas y potentes funciones de desarrollo de equipos, perfectas para trabajar con Nube de Xcode, así como con las funciones de colaboración de GitHub, Bitbucket y GitLab (Developer, 2021).

Figura 4.2.3: Logo de XCode

Fuente: https://developer.apple.com/assets/elements/icons/xcode-12/xcode-12-96x96.png

- **Postman.** "Aplicación utilizada para realizar pruebas API. Es un cliente HTTP que da la posibilidad de testear 'HTTP requests' a través de una interfaz gráfica de usuario, por medio de la cual se obtienen diferentes tipos de respuesta que posteriormente deberán ser validados" (Moreno, 2021).

Figura 4.2.4: Logo de Postman

Fuente: https://miro.medium.com/max/1000/0*Ij4wyJ4yMq_0Vm_U.png

4.3 Lenguajes de Programación

"En términos generales, los *lenguajes de programación* son herramientas que permite desarrollar o ejecutar programas para computadoras o *software*. Son empleados para implementar y diseñar programas encargados de definir y administrar el comportamiento de los dispositivos lógicos y físicos dentro del entorno de diferentes computadores. Esto se consigue mediante la creación e implementación de algoritmos utilizados como una forma de interacción

entre la computadora y el ser humano. Una de las funciones principales de los lenguajes de programación consta en escribir programas permitiendo de esta manera la comunicación usuario máquina" (Ceballos, 2004)

- **JavaScript.** "Es un lenguaje de programación utilizado especialmente para la creación de páginas web dinámicas. Las páginas Web dinámicas son aquellos que incorporan efectos a los textos que aparecen y desaparecen, acciones que se activan al pulsar botones, animaciones, y las ventanas que muestran mensajes de aviso para el usuario" (Perez,2019).

4.4 Editores de Texto.

- **Visual Studio Code. "**Es un editor de código fuente autónomo, este editor puede ser ejecutado en sistemas operativos como Linux, Windows, y macOS. La selección principal para desarrolladores web y Java, con multitud de extensiones para admitir casi cualquier lenguaje de programación" (Microsoft, 2021).

4.5 Servidores Locales

- **Node JS (BACKEND)**

  Arquitectura. "Node (llamado también: *Node.js*) es un entorno de código abierto que trabaja en tiempo de ejecución, sobre todo es un entorno multiplataforma, que permite a los desarrolladores la creación de herramientas de lado servidor y aplicaciones en JavaScript. Está considerada que la a ejecución en tiempo real pueda usarse fuera del contexto de un explorador web que consiente en ejecutarse directamente en un sistema operativo de servidor o en una computadora. Como tal, el entorno omite las APIs de JavaScript específicas del explorador web y añade soporte para APIs de sistema operativo más tradicionales que incluyen HTTP y bibliotecas de sistemas de ficheros" (MDN contributors, 2022)

  "El lenguaje común de la plataforma es JavaScript. Este es un lenguaje básico de scripting del lado del cliente para páginas web. Se utiliza en la construcción de páginas web dinámicas y ha recibido muchas mejoras" (Chitra & Satapathy, 2017).

  Rendimiento. El rendimiento de una app está determinado por más de una medida. El rendimiento se refiere a la también a la fluidez de la UI y la falta de fluidez, y avece a la velocidad en bruto. Otros ejemplos de rendimiento incluyen I/O velocidad de red.

  "Node.js proporciona un ambiente en tiempo de ejecución, al igual que la forma en que la JVM está presente para Java. Node.js está implementado en C + +. V8 es rápido debido a las características como la compilación a tiempo con código nativo y la gestión de la memoria. Esto se ocupa de los programas que se están ejecutando en el hilo de eventos. Internamente consiste en un grupo de subprocesos como si las

aplicaciones que están funcionando bajo E/S no bloqueante" (Chitra & Satapathy, 2017).

Bibliotecas y módulos. Una biblioteca o módulo es una clase rediseñada para realizar una función específica dentro del desarrollo de aplicaciones web, también lo usamos para la conexión con nuestro framework. Existen varios módulos o bibliotecas que no necesitan una instalación adicional, un lenguaje de programación sin bibliotecas es inútil para aplicaciones prácticas. Por lo tanto, Node.js tiene una API para operaciones de E/S, redes básicas, módulos de servidor HTTP, compresión y para muchas otras tareas requeridas. Las bibliotecas de Node.js que están disponibles se pueden ampliar con paquetes adicionales. Los paquetes están disponibles a través de registros de paquetes públicos o privados. Se pueden instalar utilizando el administrador de paquetes NPM y están organizados de acuerdo con el formato de paquete Common JS. (Chitra & Satapathy, 2017).

4.6 Base de Datos

Una base de datos contiene datos relacionados entre sí, los mismos que se encuentran estructurados o agrupados; además que permiten ser manipulados por programas conocidos actualmente como sistema de gestión de bases de datos (SGBD) (Ordonez, Rios & Castillo, 2017).

PostgreSQL: PostgreSQL es un gestor de bases de datos orientadas a objetos (SGBDOO o ORDBMS) es muy conocido y usado en varios entornos de software libre porque cumple los estándares SQL92 y SQL99, y también por el conjunto de funcionalidades avanzadas que soporta, lo que lo sitúa al mismo o a un mejor nivel que muchos SGBD comerciales (Ginestà & Mora, 2012).

4.7 Delivery

Entregas a domicilio por parte de una empresa, a usuarios que compren mediante vía online, puede ser por medio de aplicaciones móviles o por medio de páginas web.

MARCO METODOLÓGICO

En el siguiente capítulo presentaremos el sistema o metodología que hemos utilizado para el desarrollo del proyecto.

5.1 Metodología de desarrollo de Software

5.1.1 Introducción

Con la metodología Scrum lo que se pretende es alcanzar el mejor resultado de un proyecto determinado. Las prácticas que se aplican con la metodología Scrum se retroalimentan unas con otras y la integración de las mismas tiene su origen en un estudio de cómo hay que coordinar a los equipos para ser potencialmente competitivos. (Redacción APD, 2019).

5.1.2 Marco de trabajo de Scrum

Se divide en las siguientes fases:

- El qué y el quién: dentro de los miembros de identificaran los roles y la responsabilidad de estos.
- El cuándo y dónde; es el sprint.
- El cómo y el porqué; referencia al equipo de Scrum y así mismo a la herramienta que se van a usar.

5.1.3 Roles de Scrum.

Product Owner

Persona encargada de financiar el proyecto, y también de realizar la publicación necesaria para que el proyecto sea eficiente.

Scrum Master

Es el encargado de asegurar que las bases y técnicas de la metodología están siendo aplicadas para el correcto control del desarrollo del proyecto

Equipo de desarrollo.

Son los encargados de realizar las pruebas y el desarrollo del producto.

5.1.4 Sprint: cuándo y dónde

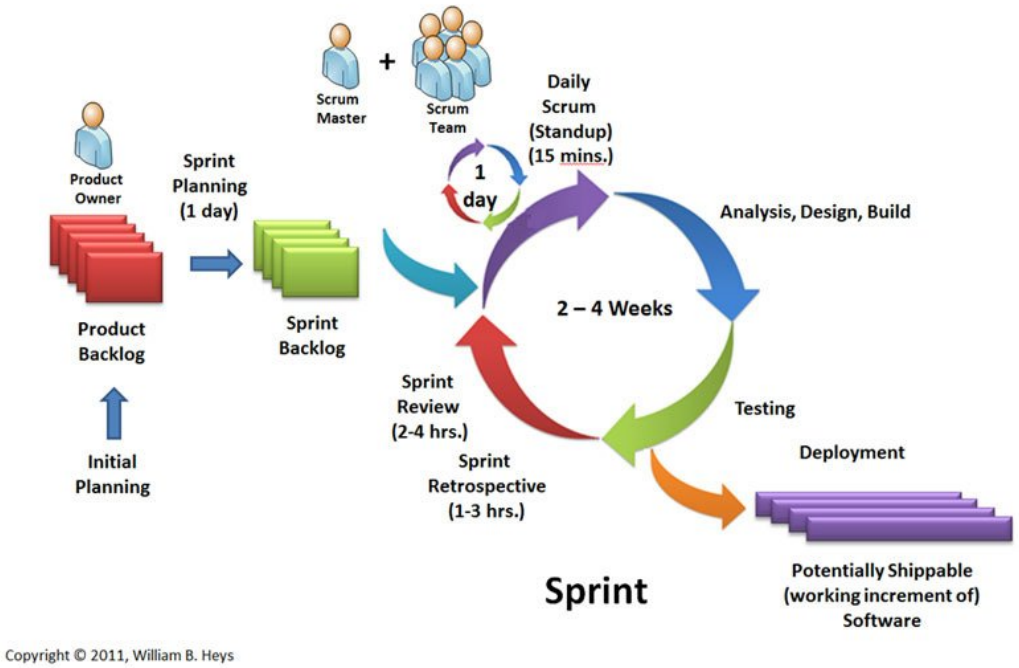


Figura 5.1.1. Sprint y modelos de desarrollo ágil.

Sprint es el corazón, tiene una duración de un mes y es donde se produce el desarrollo de un producto.

También se puede definir el Sprint como un mini proyecto, en donde el equipo de trabajo se focaliza en el desarrollo de tareas para alcanzar el objetivo que se ha definido previamente en el Sprint planning. (Redacción APD, 2019)

Dentro del sprint intervienen.

Planificación de sprint.

Son las reuniones que se llevarán a cabo para decidir los requerimientos o qué tareas serán asignadas a los miembros del equipo.

Reunión del Equipo.

Reuniones no más de 15 min en los que el equipo de desarrollo y Scrum Master.

Revisión de sprint.

Clientes y desarrolladores se reúnen para mostrar que el desarrollo del software se ha completado, mostrándoles su correcto funcionamiento.

Retrospectiva de sprint.

Reunión para evaluar cuáles fueron los procesos que se hicieron bien y de las que se hizo mal, inconvenientes que surgieron entre otros.

5.1.5 Herramientas scrum

Product backlog

Información muy general del proyecto, que a veces no son tomadas como requerimientos.

Sprint backlog

Tareas elegidas y tareas a ser destinadas y posterior a estas serán desarrolladas.

5.2 REQUERIMIENTOS

ELICITACION DE REQUERIMIENTOS

REQUERIMIENTOS

Los requerimientos nos permiten definir las funciones, capacidades o en si los tributos intrínsecos de nuestro proyecto, es decir nos ayuda a escribir como nuestro sistema debe de comportare.

Según (Aguilar, L. U. M., López, M. D. L. Á. S., & Peña, J. M. F, 2008) "Un requerimiento, es algo que debe de hacerse o como se supone o se espera, que deba de funcionar el producto que se está desarrollando, esto con el fin de satisfacer las necesidades específicas".

5.2.1 Requerimientos funcionales

En los requerimientos funcionales realizaremos las diferentes descripciones explicitas del comportamiento de esta manera tener una solución para nuestro proyecto inicial.

A continuación, se detallan los requerimientos funcionales del proyecto:

Identificador: FR01	**Nombre:** Frontend	**FUNCIONAL**
Descripción: Interacción entre el usuario y la aplicación de delivery.	**Categoría (Visible/No Visible):** NO VISIBLE	

Objetivo: Se realiza las peticiones GET, PUT, POST hacia el Backend.	
Información de entrada: Visualización de ventanas realizadas en FLUTTER	**Información de salida:** Peticiones al Backend
Condiciones de aceptación: No existe	
Condiciones previas: El usuario debe estar registrado.	
Postcondición: Presenta la información al usuario.	
Prioridad: Alta	

Tabla 1 RF Servicio de Frontend (Elaboración propia)

Identificador: FR02	**Nombre:** Backend	**FUNCIONAL**
Descripción: Procesar la información que envié el Frontend.	**Categoría (Visible/No Visible):** NO VISIBLE	
Objetivo: Acceder a la información que el usuario requiera, consultando en nuestra base de datos.		
Información de entrada: Peticiones de entrada al Backend realizada en Nodejs	**Información de salida:** Información enviada al Frontend.	
Condiciones de aceptación: No existe		
Condiciones previas: Uso de base de datos y librerías para devolver información		
Postcondición: Presenta la información al usuario.		
Prioridad: Alta		

Tabla 2 RF Servicio de Backend (Elaboración propia)

Identificador: FR03	**Nombre:** Almacenamiento de imágenes en Firebase	**FUNCIONAL**
Descripción: Acceso a las imágenes mediante almacenamiento de URL en PostgreSQL	**Categoría (Visible/No Visible):** NO VISIBLE	
Objetivo: Guardar imágenes en la nube, para luego ser llamadas mediante URL dentro de la aplicación.		
Información de entrada: Aplicación	**Información de salida:** Imagen	
Condiciones de 20captación: No existe		
Condiciones previas: Disponibilidad de las imágenes en Firebase		
Postcondición: Imágenes guardadas.		
Prioridad: Alta		

Tabla 3 RF Almacenamiento de imágenes en Firebase (Elaboración propia)

Identificador: FR04	**Nombre:** Almacenamiento en base de datos PostgreSQL	**FUNCIONAL**
Descripción: Acceso a la información almacenada por medio de peticiones	**Categoría (Visible/No Visible):** NO VISIBLE	

Objetivo: Desplegar información detallada en la base de datos.	
Información de entrada: Base de datos creada	**Información de salida:**
Condiciones de aceptación: No existe	
Condiciones previas: Conexión con Node.js y flutter	
Postcondición: Tener activada la conexión a la base de datos desde Node.js	
Prioridad: Alta	

Tabla 4 RF Almacenamiento en base de datos PostgreSQL (Elaboración propia)

USUARIOS.

Identificador: FR01	**Nombre:** Ingreso de nuevos usuarios	**FUNCIONAL**
Descripción: Los nuevos usuarios podrán registrarse en la aplicación (Restaurante, repartidor, cliente)	**Categoría (Visible/No Visible):** Visible	
Objetivo: Asegurar el ingreso de los usuarios en la aplicación		
Información de entrada: Datos del Usuario. • ID, • Email • nombre, • Apellido, • Teléfono, • Contraseña	**Información de salida:**	
Condiciones de aceptación: Ninguna		
Condiciones previas: Identificare como el rol a que pertenece		
Postcondición: La aplicación deberá guardar los usuarios creados		
Prioridad: Alta		

Tabla 5 RF Ingreso de nuevos usuarios (Elaboración propia)

Identificador: FR02	**Nombre:** Detalle de usuarios registrados	**FUNCIONAL**
Descripción: Los nuevos usuarios podrán registrarse en la aplicación (Restaurante, repartidor, cliente)	**Categoría (Visible/No Visible):** Visible	
Objetivo: Una vez el cliente registrado podrá acceder al menú donde seleccionará que rol cumple.		

Información de entrada: Listar los usuarios creados (Restaurante, cliente, repartidor)	**Información de salida:** Listade los del Usuario registrados. • ID, • Email • nombre, • Apellido, • Teléfono, • Contraseña, • Imagen
Condiciones de aceptación: Usuarios creados	
Condiciones previas: Previamente debe existir usuarios creados	
Postcondición: La aplicación deberá de mostrar los usuarios creados	
Prioridad: Alta	

Tabla 6 RF Detalle de usuarios registrados (Elaboración propia)

Identificador: FR03	**Nombre:** Identificarse con el Rol al que pertenece	**FUNCIONAL**
Descripción: Una vez registrado el cliente de identificar con el Rol (Restaurante, repartidor, cliente)	**Categoría (Visible/No Visible):** Visible	
Objetivo: Seleccionar el rol al cual pertenece		
Información de entrada: Selección del Rol	**Información de salida:** Listar los Roles para el usuario. • Cliente • Restaurante • Repartidor	
Condiciones de aceptación: Rol relacionado		
Condiciones previas: El usuario deberá de estar registrado para seleccionar el rol		
Postcondición: La aplicación muestra el rol que se debe seleccionar.		
Prioridad: Alta		

Tabla 7 RF Identificarse con el Rol al que pertenece (Elaboración propia)

Identificador: FR07	**Nombre:** Registro de categorías	**FUNCIONAL**
Descripción: Con el rol de restaurante se podrá crear nuevas categorías.	**Categoría (Visible/No Visible):** Visible	
Objetivo: El rol que cuenta para el restaurante se podrá crear nuevas categorías que permitan ser visualizados por el cliente**.**		
Información de entrada: • ID • Nombre • Descripción	**Información de salida:**	
Condiciones de aceptación: Ninguna		

Condiciones previas: Para crear categoría deberá pertenecer al rol del Restaurante
Postcondición: La aplicación deberá de guardar la nueva categoría.
Prioridad: Alta

Tabla 8 RF Registro de Categorías (Elaboración propia)

Identificador: FR07	**Nombre:** Detalle de categorías	**FUNCIONAL**
Descripción: Con el rol de restaurante puede visualizar las categorías creadas	**Categoría (Visible/No Visible):** Visible	
Objetivo: Garantizar que se han creado nuevas categorías.		
Información de entrada:	**Información de salida:** • ID • Nombre • Descripción	
Condiciones de aceptación: Categoría creada.		
Condiciones previas: Deben existir nuevas categorías creadas.		
Postcondición: Mostar en la aplicación las categorías		
Prioridad: Alta		

Tabla 9 RF Detalle de categorías (Elaboración propia)

Identificador: FR07	**Nombre:** Registro de Productos	**FUNCIONAL**
Descripción: El restaurante podrá crear nuevos productos	**Categoría (Visible/No Visible):** Visible	
Objetivo: Registrar nuevos productos dentro de la aplicación.		
Información de entrada: • ID • Nombre • Descripción Precio	**Información de salida:**	
Condiciones de aceptación: Ninguna		
Condiciones previas: Debe de cumplir con el rol de Restaurante		
Postcondición: Se deberá de registrar los nuevos productos.		
Prioridad: Alta		

Tabla 10 RF Registro de Productos (Elaboración propia)

Identificador: FR08	**Nombre:** Dirección de usuario con Google Maps en Flutter	**FUNCIONAL**
Descripción: Se registra en el mapa las direcciones de los usuarios para referencia usada por los repartidores.	**Categoría (Visible/No Visible):** Visible	
Objetivo: Detallar la dirección exacta con coordenadas de latitud y longitud.		

Información de entrada: • Dirección • Referencia • Punto de referencia	**Información de salida:** Ubicación en el mapa
Condiciones de aceptación: Dirección registrada	
Condiciones previas: Se puede visualizar las direcciones registradas	
Postcondición: No se podrá realizar un pedido sin tener una dirección registrada.	
Prioridad: Alta	

Tabla 11 RF Dirección de usuario con Google Maps en Flutter (Elaboración propia)

Identificador: FR09	**Nombre:** Pedido despachado	**FUNCIONAL**
Descripción: Cuando el producto es pedido por el cliente, este pedido será despachado por el restaurante eligiendo un repartidor.	**Categoría (Visible/No Visible):** Visible	
Objetivo: Seguir un orden para que la entrega sea satisfactoria al cliente.		
Información de entrada: • Información del repartidor • Información del pedido • Información de la dirección a entregar el pedido	**Información de salida:** Pedido despachado en el rol cliente.	
Condiciones de aceptación: Despacho entregado a repartidor para su entrega		
Condiciones previas: Tener un repartidor registrado		
Postcondición: No se podrá realizar un pedido sin tener una dirección registrada y un repartidor asignado		
Prioridad: Alta		

Tabla 12 RF Pedido despachado (Elaboración propia)

Identificador: FR09	**Nombre:** Pedido en camino	**FUNCIONAL**
Descripción: El repartidor ya asignado realizara el pedido para realizar la entrega en la dirección del cliente.	**Categoría (Visible/No Visible):** Visible	
Objetivo: Realizar el pedido al cliente, con la condición de realizar la entrega si se encuentra a menos de 100 metros del cliente, caso contrario la aplicación no le permitirá realizar la entrega.		

Información de entrada: •Ubicación del repartidor • Ubicación del cliente • Información del cliente • Teléfono del cliente	**Información de salida:** Seguimiento del repartidor en tiempo real
Condiciones de aceptación: El repartidor tiene que estar a menos de 100m para poder realizar la entrega del pedido.	
Condiciones previas: Tener la dirección del cliente	
Postcondición: No se podrá realizar la entrega si el cliente se encuentra más de 100m de distancia del repartidor.	
Prioridad: Alta	

Tabla 13 RF Pedido en camino (Elaboración propia)

Identificador: FR09	**Nombre:** Pedido entregado	**FUNCIONAL**
Descripción: El pedido será entregado siempre y cuando el cliente haya llegado al lugar indicado.	**Categoría (Visible/No Visible):** Visible	
Objetivo: Realizar el pedido al cliente, con la condición de realizar la entrega si se encuentra a menos de 100 metros del cliente, caso contrario la aplicación no le permitirá realizar la entrega.		
Información de entrada: •Ubicación del repartidor • Ubicación del cliente • Información del cliente • Teléfono del cliente	**Información de salida:** Visualización del repartidor en el lugar de entrega del pedido.	
Condiciones de aceptación: El repartidor tiene que estar a menos de 100m para poder realizar la entrega del pedido.		
Condiciones previas: Tener la dirección del cliente, teléfono y datos		
Postcondición: No se podrá realizar la entrega si el cliente se encuentra más de 100m de distancia del repartidor.		
Prioridad: Alta		

Tabla 14 RF Pedido entregado (Elaboración propia)

Identificador: FR10	**Nombre:** Seguimiento del repartidor en tiempo real.	**FUNCIONAL**
Descripción: El cliente podrá visualizar la ubicación del repartidor en tiempo real hasta que llegue a su destino.	**Categoría (Visible/No Visible):** Visible	
Objetivo: Saber la ubicación del repartidor en tiempo real.		

Información de entrada: •Ubicación del repartidor en el mapa	**Información de salida:** Ubicación del repartidor en el mapa
Condiciones de aceptación: El repartidor tiene que estar a menos de 100m para poder realizar la entrega del pedido.	
Condiciones previas: Tener la dirección del cliente, teléfono y datos	
Postcondición: No se podrá realizar la entrega si el cliente se encuentra más de 100m de distancia del repartidor.	
Prioridad: Alta	

Tabla 15 RF Seguimiento del repartidor en tiempo real. (Elaboración propia)

Identificador: FR10	**Nombre:** Búsqueda de Productos	**FUNCIONAL**
Descripción: El cliente podrá realizar la búsqueda del producto que desea comprar	**Categoría (Visible/No Visible):** Visible	
Objetivo: Facilitar la compra al usuario realizando búsquedas de lo que desea comprar		
Información de entrada: •Nombre del producto a comprar	**Información de salida:** Visualización del producto para su compra	
Condiciones de aceptación: Tener productos registrados en la base de datos.		
Condiciones previas: Tener productos registrados en la base de datos		
Postcondición: En la búsqueda aparecerán los productos registrados con el nombre con el que se realizó la búsqueda.		
Prioridad: Alta		

Tabla 16 RF Búsqueda de Productos. (Elaboración propia)

5.2.2 Requerimientos no funcionales

Identificación del requerimiento	RNF01
Clase del requerimiento	Librerías
Descripción del requerimiento	Las librerías de la aplicación no deben ser actualizadas y trabajar en conjunto con las demás librerías para evitar incompatibilidad.
Importancia	Evitar actualizaciones erróneas eh incompatibles.

Tabla 17 RNF01 (Elaboración propia)

Identificación del requerimiento	RNF02
Clase del requerimiento	Base de datos
Descripción del requerimiento	La base de datos de nuestra aplicación debe tener un respaldo o una réplica para evitar pérdidas de información.
Importancia	Proteger la información de la base de datos.

Tabla 17 RNF02 (Elaboración propia)

5.3 DISEÑO DE LA ARQUITECTURA

5.3.1 Arquitectura y componentes de la arquitectura

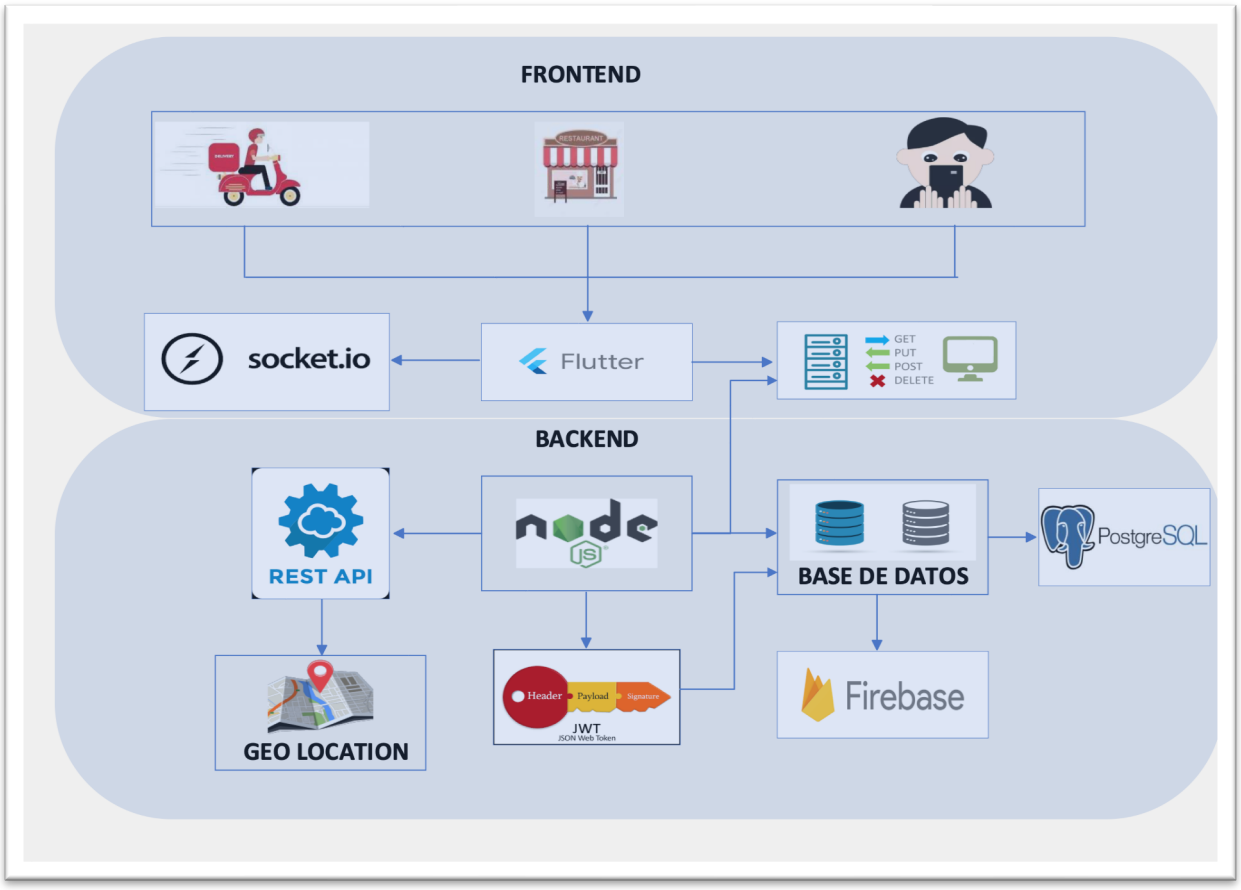


Fig.5.3.1.1 Arquitectura genera (Esquema Propio)

Dentro de los objetivos que nos establecimos desde un principio el poder contribuir con una posible solución a los usuarios que realizan compras en línea y contribuir a mantener el distanciamiento social mediante el uso de la App que permitirá a los usuarios efectuar sus compras, sin salir de casa en tiempos de pandemia, podemos determinar las siguientes soluciones.

- Una vez que nuestra aplicación esté terminada, analizada, diseñada, implementada y desplegada en cualquier tienda móvil ayudaremos a que los usuarios al momento de descargar la aplicación puedan realizar sus compras, no únicamente compras de alimentos, en este caso utilizamos un ejemplo de tienda de comida, pero en general la aplicación ayudara a que los usuarios realicen cualquier tipo de pedido y este sea entregado en sus domicilios, de esta manera facilitaremos en gran medida la compra de productos y los usuarios finales serán beneficiados al no salir de casa para obtener la compra que deseen.
- También nos enfocamos principalmente en los diferentes empleados o empleadores de la ciudad, cuando uno de ellos desee realizar alguna compra y se encuentren en horarios de oficina será muy difícil para ellos salir a buscar o comprar su producto en este caso también es una buena idea instalar en su móvil nuestra aplicación para solucionar sus problemas de compras de productos.
- Con el difícil momento que pasamos por el COVID, es mejor tener varias alternativas o tiendas en línea que nos faciliten el trabajo de realizar compras, en este caso existen varias aplicaciones que ofrecen el servicio de pedidos a domicilio, en este caso nuestra aplicación forma parte de una de las mejores alternativas que tiene el usuario para poder realizar pedidos del cualquier producto que desee, enfocándonos principalmente en no monopolizar un mercado con un solo aplicativo móvil si no dar a conocer varias alternativos para que el usuario final sea el que elija su preferida, y de esta manera guardar un distanciamiento entre nosotros los consumidores.

Con estas respuestas que pudimos obtener para nuestros objetivos principales, podemos destacar una arquitectura como lo podemos observar en Fig.5.3.1.1 llamada Arquitectura genera, en donde podemos observar un proceso que conlleva a la solución de una compra en línea con seguimiento del repartidor en tiempo real.

1. Comenzamos desarrollando el frontend, utilizando herramientas como Flutter, con esta gran herramienta podemos obtener librerías y eventos con los cuales pudimos realizar seguimientos en tiempo real con librerías que ayudan a la visualización del frontend y librerías que ayudan a la conexión de nuestro Backend por medio de Node.js llamado Socket.io.

2. Procedemos a realizar nuestro Backend con el cual administraremos todos los datos que tengamos en nuestra base de datos y poderlos entregar directamente a nuestro frontend para que el usuario final pueda realizar sus diferentes consultas o pedidos.
3. Dentro de nuestro Backend trabajamos con bases de datos llamados Postgres y Firebase los cuales están asociados un JWT para generar tokens de ingreso que facilita la identificación de los usuarios.
4. Para poder realizar la geolocalización utilizamos una arquitectura llamada API REST la cual por medio de peticiones podemos determinar diferentes ubicaciones.

5.4 Programas utilizados para el desarrollo de la aplicación.

El software utilizado se detalla a continuación:

5.4.1 Para el desarrollo en un Sistema Operativo Windows.

- Flutter (SDK creado por Google)
- Android Studio,
- Visual Studio Code,
- Base de datos (PostgreSQL),
- Postman,
- Servidor local (Node Js),
- Correr la aplicación es un dispositivo físico Android.

5.4.2 Para el desarrollo en un Sistema MAC.

- Flutter (SDK creado por Google)
- Android Studio,
- Xcode,
- Visual Studio code,
- Base de datos (PostgreSQL),
- Postman,
- Servidor local (Node Js).

Para poder realizar una simulación en nuestro emulador IOS, debemos tener instalado un programa llamado Xcode es decir un IDE (Entorno de Desarrollo Integrado), puesto que cuenta con múltiples herramientas creadas por Apple con el fin de desarrollar aplicaciones móviles en MacOS, para lo cual debemos tener instalada la última versión y posterior a esto realizar configuraciones internas dentro del terminal del ordenador.

Cuando las configuraciones estén realizadas correctamente, obteniendo las licencias firmadas por el propietario del ordenador, se procede a ejecutar y probar el emulador IOS ya con la aplicación Flutter.

Cuando el simulador de IOS está ejecutado correctamente, en el programa Android Studio se visualiza la opción para trabajar con dicho simulador.

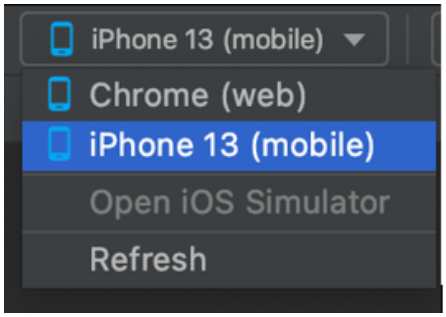


Figura 5.4.2.1 Opciones de Simulador en Android Studio.

Figura 5.4.2.2: Emulador IOS (MAC)

Figura 5.4.2.3: Emulador Android (Windows)

Estos son requerimientos necesarios que se necesitará al momento de desarrollar la App en un Sistema Operativo MAC.

5.5 Diseño de la aplicación

Para comenzar con el desarrollo de un nuevo proyecto en Flutter, se inicia trabajando en las configuraciones del archivo **main.dart,** el mismo que es generado al momento de crear un nuevo proyecto, este servirá a su vez para trabajar con el diseño de diferentes pantallas con las cuales contará la App.

5.5.1 Etiquetas

Dentro de las etiquetas se puede encontrar la forma en la cual está estructurado cada una de las pantallas que serán visualizadas. Todas las pantallas deben estar estructuradas dentro del framework Flutter, las etiquetas son de mucha utilidad y de mucha importancia.

El usuario al ingresar sus datos, ingresara directamente a su Rol a cuál ha sido asignado, por lo tanto, si el usuario permanece o es un Restaurante, tendrá una pantalla diferente a la del cliente o repartidor.

Un cliente que se registre en nuestra aplicación podrá pertenecer al rol que desee, incluso puede llegar a tener los 3 roles como cliente, restaurante o repartidor, dándole la opción al momento de ingresar que el usuario pueda escoger el rol que desee.

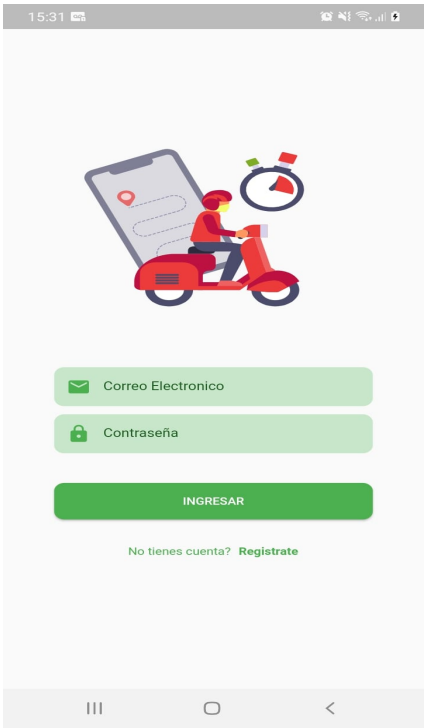


Figura 5.5.2.1: Pantalla de Inicio

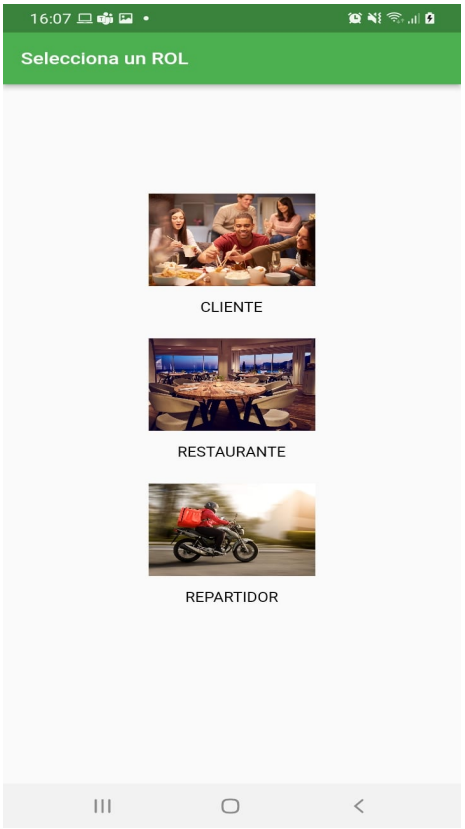


Figura 5.5.2.2: Pantalla de Roles

El mismo proceso se aplica para el repartidor o cliente.

5.5.3 Widgets

"Son construidos usando un moderno entorno de trabajo (framework) inspirado en React. Los Widgets describen la vista de la aplicación, dada su configuración y estado actuales. Cuando el estado de un widget cambia, el widget reconstruye su descripción, ya que el framework difiere de la descripción anterior para determinar los cambios mínimos necesarios en el árbol de renderizado subyacente para la transición de un estado al siguiente" (Flutter, 2022).

Para el desarrollo de la aplicación se crearán automáticamente nuevos Widgets que vienen a ser subclases de Stateless Widget o Stateful Widget, esto dependerá si el widget está o no gestionando algún estado.

Se detallan a continuación algunos de los widgets usados en el desarrollo de la aplicación:

- **Text:** Permite crear una cadena de texto utilizando diferentes estilos estilo.

- **Row, Column:** Permiten crear layouts flexibles, estos pueden ser horizontales como verticales (Row, Column), estos diseños están basados en el modelo de layouts flexbox de la web
- **Stack:** Se basan en el modelo de layout de posicionamiento absoluto dentro de la web, permite aplicar los widgets uno encima de otro, el orden de posición lo define el programador, se pueden utilizar el widget Positioned en los hijos de un Stack para que estos sean posicionados en relación con el borde superior, derecho, inferior o izquierdo del stack.
- **Container:** Crear un elemento visual rectangular. Los contenedores pueden ser decorados utilizando una herramienta llamada Box Decoration, como un borde, una sombra o un borde. Los contenedores tienen los siguiente atributos márgenes, relleno interno y restricciones aplicadas a su tamaño. Además, un Contenedor puede transformarse en un espacio tridimensional gracias a la ayuda de las matrices.

5.5.4 Animaciones

Un objeto Animation en Flutter, son clases que van a generar secuencias de número interpolados entres dos valores durante un lapso de tiempo. El objeto Animation puede ser lineal, curva, por función de pasos o por cualquier otro mapeado. Dependientemente de cómo se controle el objeto, podría ejecutarse en modo inverso, o incluso cambiar la dirección en el medio (Flutter, 2022).

5.5.5 Diseño de la pantalla de registro

En la imagen muestra los datos que los usuarios deberán ingresar en la aplicación para que pueda registrarse, los datos que el usuario debe ingresar son: correo electrónico, nombres, apellidos, teléfono y una contraseña. Estos datos estarán registrados y almacenados en la base de datos, al momento que el usuario se registre, la aplicación automáticamente le asignará el rol de “Cliente”, para por optar por el rol (“Restaurante” o “Repartidor”) el cliente tendrá que comunicarse con los administradores de la aplicación, de esta manera constatamos que dicho cliente cuenta con un restaurante, o es apto para formar parte de los repartidores, todo esto se lo realizará mediante visitas al restaurante y en el caso del repartidor tendremos que obtener sus datos y los de su vehículo de transporte.

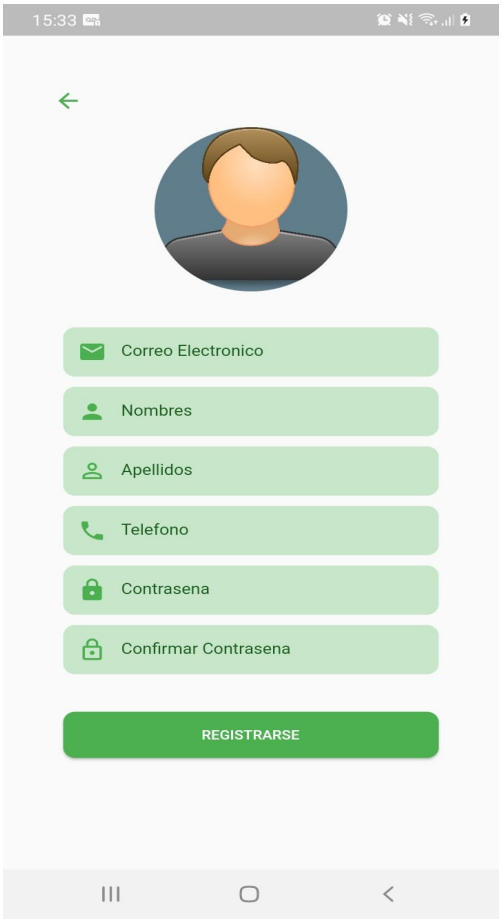


Figura 5.5.5.1: Pantalla de registro.

Cuando un usuario se registre, tendrá que esperar que los datos se almacenen en nuestra base de datos, esto tardará de 1 a 2 segundos, mientras tanto en la pantalla les aparecerá un círculo que indica el avance de la carga. Una vez que se cargue la información en la base de datos, la aplicación automáticamente lo enviará a la página de Login.

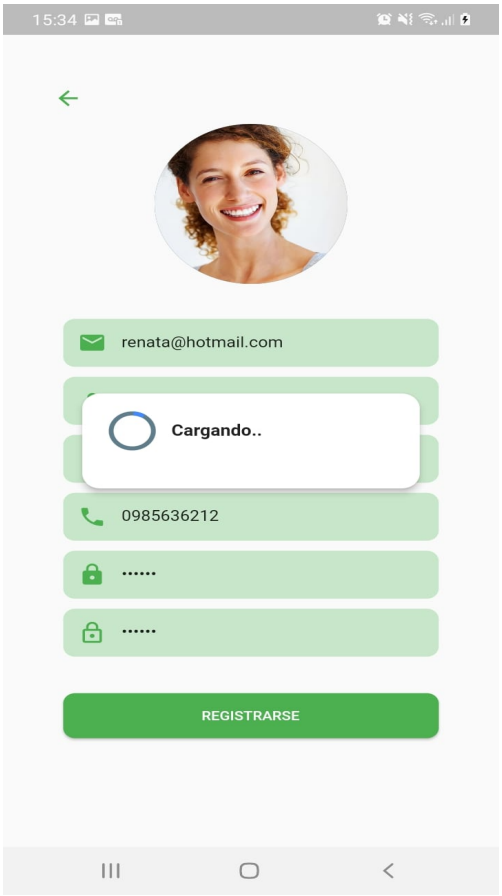


Figura 5.5.5.2: Avance de carga.

5.5.6 Servidor en Node JS

Como se mencionó anteriormente se utiliza un servidor, Node Js ya deberá de estar instalado en la computadora. En primer lugar, se debe ubicar en la carpeta donde guardar nuestro el proyecto.

Algunas de las ventajas de Node.js son:

- Tiene un excelente rendimiento.
- Código escrito en JavaScript (Lenguaje de programación nuevo y sofisticado adaptable a cualquier entorno, con excelentes diseños comparado a lenguajes tradicionales de servidores web).
- Es portable con versiones que funcionan en Microsoft Windows, OS X, Linux, Solaris, FreeBSD, OpenBSD, WebOS, y NonStop OS. (MDN contributors, 2022)

"Además, crear de forma sencilla servidores web básicos los cuales responderán cualquier petición simplemente usando el paquete HTTP de *Node*, como se muestra en la figura 2.1. Este, creará un servidor y escuchará cualquier clase de peticiones en la

URL http://localhost:8000/; cuando se reciba una petición, se responderá enviando en texto la respuesta en este ejemplo la respuesta será: ¡Hola Mundo!" (MDN contributors, 2022)

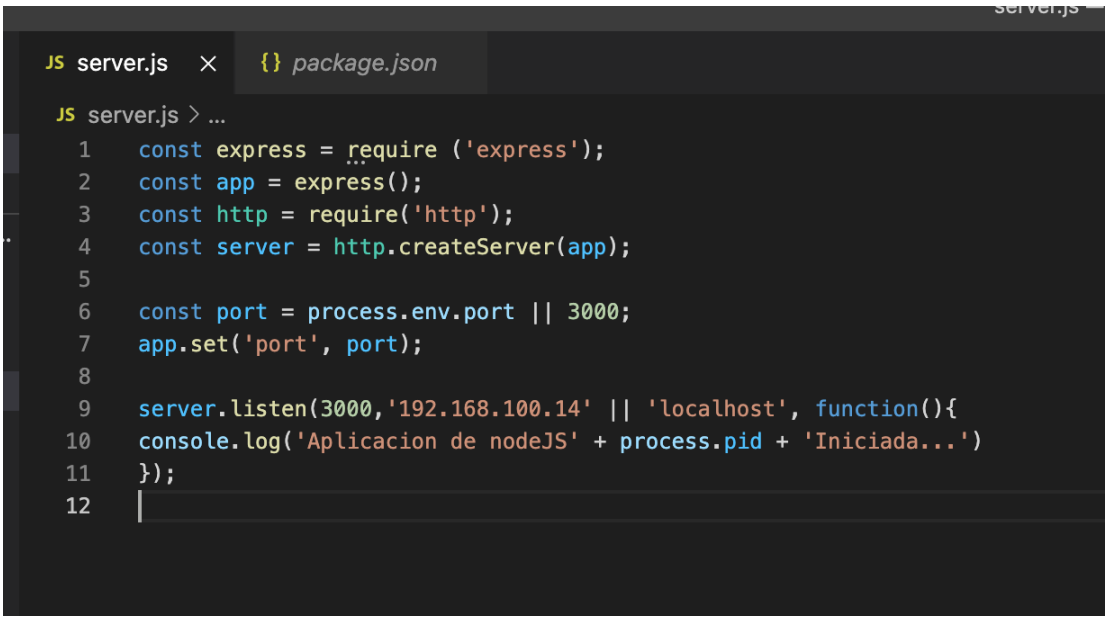


Figura 5.5.6.1: Creación de servidor Node.js

5.6 Conexión a la Base de Datos

Una vez ejecutada toda la configuración de Node.js, se procede a trabajar en la base de datos en donde crearemos las diferentes tablas para el desarrollo de la aplicación.

A continuación, se muestra el Diagrama Entidad Relación que es utilizado en el desarrollo de la aplicación.

Figura 5.6.1: Tabla Entidad Relación

5.6.1 Tablas utilizadas dentro de la aplicación.

- Usuarios
- Órdenes
- Órdenes y Productos
- Roles
- Direcciones
- Productos
- Categorías
- Usuarios y Roles

Este esquema ha sido creado en la base de datos PostgreSQL.

5.7 API REST

Las API REST están en un gran auge de popularidad puesto que la mayoría hace uso de esta tecnología para enviar y recibir datos de forma rápida y sencilla ayudando a optimizar el rendimiento de cualquier aplicación, existen tecnologías que está en crecimiento como Graph QL la cual busca reemplazar o desplazar a las API REST de esta manera serán escalables, sencillas y sobre todo robustas.

Las API REST aprovechan los métodos HTTP, desde un simple POST o GET hasta métodos personalizados, sin embargo, nosotros veremos únicamente POST, GET, PUT y DELETE en su forma más sencilla y las Headers que son para autenticación (Asfo Senior Application Developer, 2018).

5.8 Creación de nuevos usuarios

Dentro del mismo proyecto en el cual se está desarrollando la aplicación, se deberá de ejecutar una "Sequence sql".

"Una secuencia no es más que un objeto enlazado de esquema determinado por el usuario que crea una secuencia de valores numéricos de acuerdo con la descripción con la que se creó la secuencia. La secuencia de valores numéricos se crea en orden ascendente o descendente a un intervalo determinado y se puede configurar para reiniciar (ciclar) cuando se agota. Las aplicaciones se refieren a un objeto de secuencia para recuperar su siguiente valor. La relación entre secuencias y tablas está controlada por la aplicación. Las aplicaciones de usuario pueden hacer referencia a un objeto de secuencia y coordinar los valores en varias filas y tablas" (Microsoft, 2021)

Para la creación del nuevo usuario se registrará los siguientes datos:

- Email
- Nombre

- Apellido
- Teléfono
- Imagen
- Password

5.8.1 Peticiones POST desde flutter.

Para realizar peticiones POST, dentro del código se deberán de realizar diferentes configuraciones, pero todas las configuraciones servirán para ejecutarlos en diferentes proyectos a futuro.

5.8.2 Paquete HTTP.

Es un paquete que contiene una gran cantidad de funciones y clases con un gran nivel los cuales facilitaran de manera apropiada el consumo de recursos HTTP. Es multiplataforma esto quiere decir que es compatible con diferentes componentes como dispositivos móviles, computadores de escritorio y sobre todo con navegador.

Se puede encontrar diferentes paquetes o dependencias HTTP dentro del siguiente sitio web:

https://pub.dev/packages/http/install

Se necesitará usar paquetes o dependencias HTTP, ya que necesita realizar diferentes peticiones a la API, estas peticiones son “GET, POST, PUT”

5.9 Roles de Usuarios.

En la imagen#. se puede visualizar los tres tipos de roles, con las cuales cuenta la aplicación, se debe especificar que para que el usuario pueda elegir o seleccionar el rol, este deberá de ponerse en contacto con los desarrolladores de la aplicación para poder destinar un rol al cual pertenece, a excepción del cliente ya automáticamente la aplicación le designará al rol de cliente.

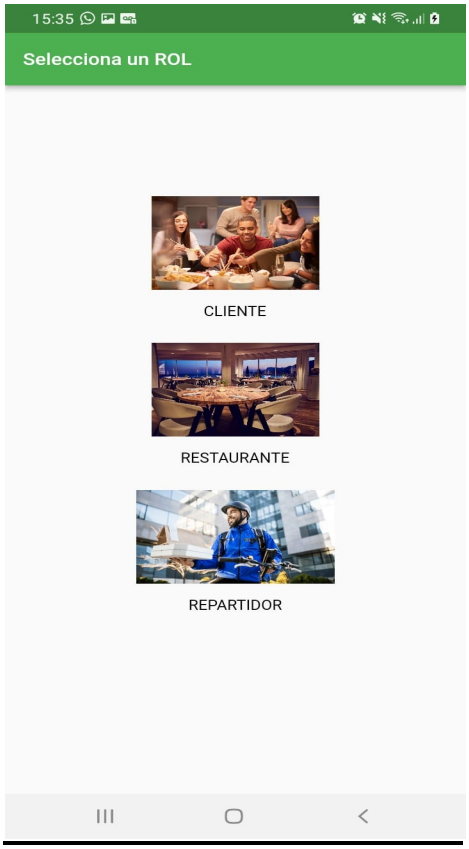


Figura 5.9.1: Rol de Usuario

5.9.1 Creación de los 3 tipos de roles

- Restaurante
- Repartidor
- Cliente

Las tablas creadas para los diferentes roles permitirán almacenar la información dentro de la aplicación. Por ejemplo, la correlación que existe entre la tabla Usuarios y la tabla Roles permite que el usuario pueda tener uno o varios roles dentro de la aplicación.

Es decir, si se tiene a un usuario con el nombre de “Juan”, este puede ser cliente, pero a la vez también puede ser repartidor y administrador de la aplicación.

Dentro de la aplicación se utilizan diferentes consultas realizadas a la base de datos, es decir, que permitirá hacer un CRUD de las tablas con la información que existe dentro de la base. Todas las tablas van a contener con un identificador único.

Cada vez que se almacena un nuevo usuario, en este caso un nuevo rol, dentro del DB existe un campo denominado “NOMBRE” el cual almacenará el nombre del rol que queremos guardar.

Se crea otra tabla de Usuario y Roles la que va a estar ligada a las dos tablas o tiene una relación tanto con la tabla Usuarios como con la tabla de Roles, esto con el fin de evitar una relación de “muchos a muchos”.

5.9.1.1 Rol por defecto.

Cada vez que un usuario se registre en la aplicación, la aplicación automáticamente le asigna un rol por defecto es decir el rol de Cliente.

5.10 Proceso para la autenticación de los usuarios.

Validar los datos como administrador, es decir si el usuario se registra y quiere ser repartidor debería de contactar con los desarrolladores quienes le asignan el rol de “repartidor”. Se procederá de similar manera con los demás roles.

Para que un usuario pueda acceder al rol “Restaurante” deberá contar con lo siguiente:

- Nombre del Restaurante
- Ubicación
- Productos
- Precios

Para que un usuario pueda acceder al rol “Repartidor” deberá contar con lo siguiente:

- Nombres completos del repartidor
- Cedula
- Papeles en línea de la Moto
- Celular

La aplicación contará con una ventana que visualiza los distintos restaurantes que están a disposición del usuario.

5.11 Menú de opciones

En el menú de opciones se va a crear una pantalla principal que dependerá del rol del usuario, es decir, la aplicación cuenta con un menú desplegable en la parte izquierda, en la imagen 5.8.1 se puede apreciar el despliegue del menú del cliente y cuenta con opciones como:

- Editar perfiles

- Mis pedidos
- Selección del rol
- Cerrar Sesión

En la opción de “Seleccionar Rol”, el usuario puede seleccionar un diferente rol en caso de tener más de uno. Esto se puede apreciar en la base de datos como también dentro del modelo Entidad-Relación.

Cada menú será distinto que el otro, con diferentes opciones, pero sin cambiar el diseño.

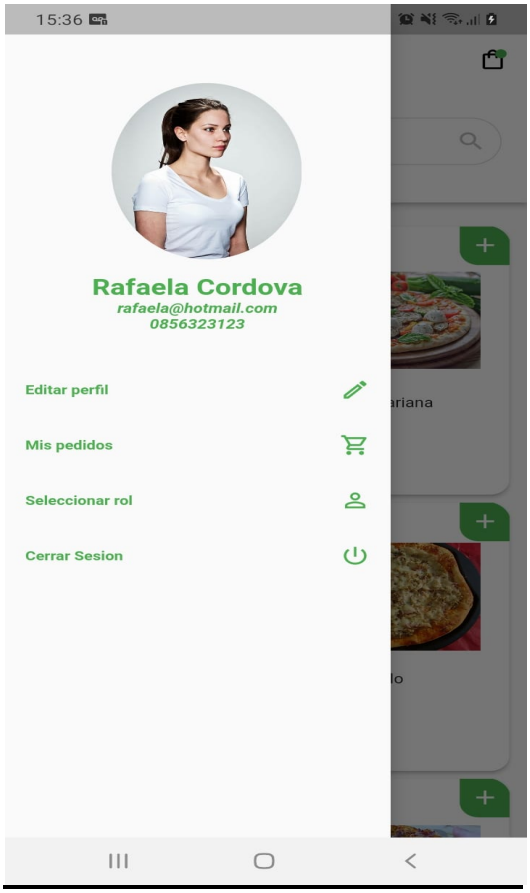


Figura 5.11.1: Menú Cliente.

Menú Delivery.

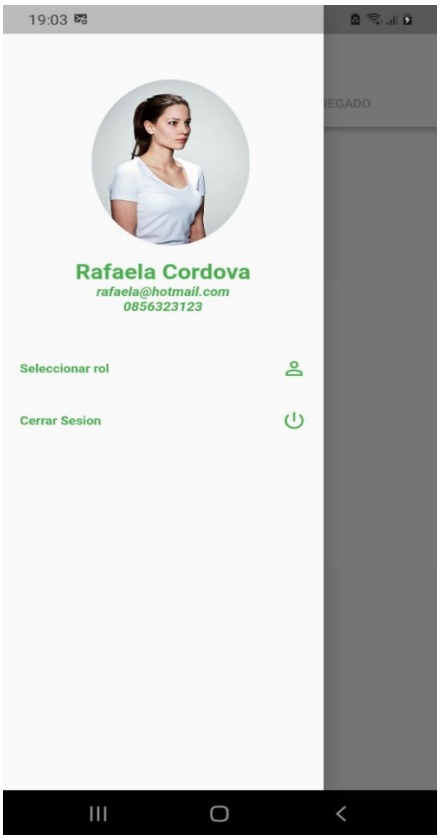


Figura 5.11.2: Menú Delivery.

Menú Restaurante

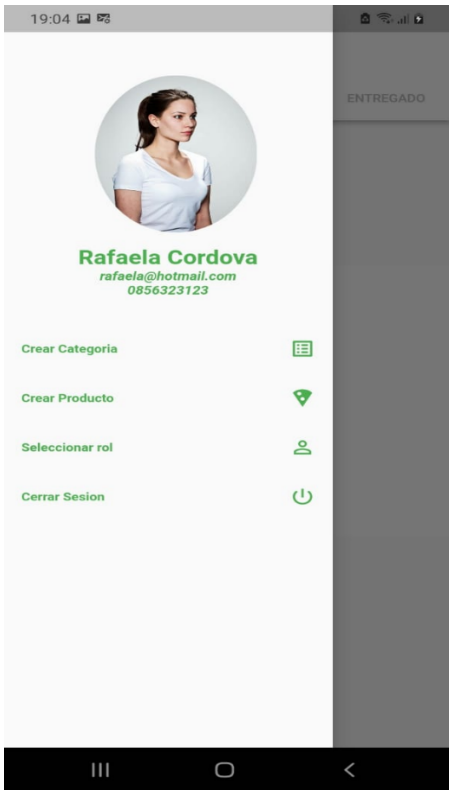


Figura 5.11.3: Menú Restaurante.

En la Figura 5.11.4, La opción “Crear categoría”, permite crear un nuevo grupo de productos que el restaurante ofrecerá, por ejemplo: “Hamburguesas”.

La opción “Crear productos” permitirá crear nuevos ítems dentro de las categorías, por ejemplo: “Hamburguesa de pollo”.

Finalmente, la última opción “Cerrar sesión” permite salir de la aplicación de manera segura.

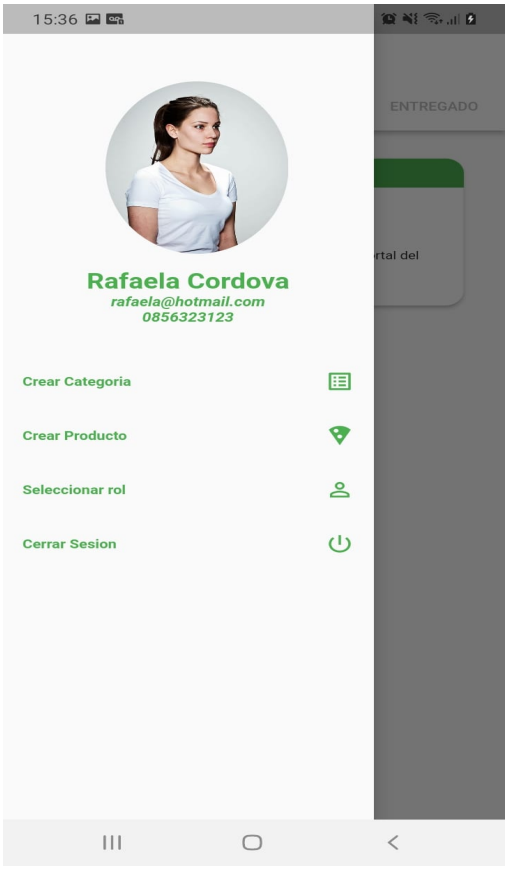


Figura 5.11.4: Opciones menú

5.12 Integración de Flutter con Firebase

5.12.1 Firebase

Firebase de Google, plataforma que se encuentra en la nube, para el desarrollo de aplicaciones web y móviles, disponible para plataformas (Android, IOS, y web) más rápido para el desarrollo de aplicaciones.

Figura 5.9.1.1: Logo de Firebase.

5.12.2 Realtime database

Consisten en bases de datos las cuales se están ejecutando en tiempo real almacenadas en la nube y se caracterizan por ser “No sql” (no relacionales). Los datos de la aplicación son actualizados en tiempo real programado para que el usuario no tenga que realizar ninguna acción.

Firebase envía los eventos de almacenamiento de manera automática cuando los datos son modificados siempre y cuando el usuario que se encuentre con la session iniciada se encuentre conectado a internet.

Google cuenta con Firebase, por lo cual es necesario acceder a la cuenta para utilizarlo.

5.12.3 Configuración Firebase con Android

Para crear un nuevo proyecto se debe establecer su nombre, posteriormente se agrega la aplicación (Android, IOS, Aplicación Web).

Para la configuración de Android se necesita:

- Paquete de Android que se está trabajando
- Sobrenombre de la App (Opcional)
- Certificado de firma SHA-1

5.12.3.1 Cómo obtener la firma SHA-1?

Dentro del computador:

1.- Paso

Disco local (C:) -> Archivos de programa -> Android -> Android Studio -> jre -> bin

2.- Paso

- Copiar la ruta directamente
- Abrir el programa **Git CMD**
- Nos ubicamos en la Ruta que copiamos con el comando **cd**
- Ingresar el siguiente comando:

```
keytool -list -v -alias androiddebugkey -keystore
%USERPROFILE%\.android\debug.keystore
```

Tener en cuenta que se debe de escribir correctamente, de lo contrario se producirá un error.

- Enter en introducir contraseña.
- Si se necesita contraseña, la contraseña es **android**
- Copiamos el certificado SHA-1.
- Pegamos el certificado en la firma SHA-1

Con esto se puede registrar la App dentro de Firebase, luego se descargará el archivo denominado **google-services. Json.**

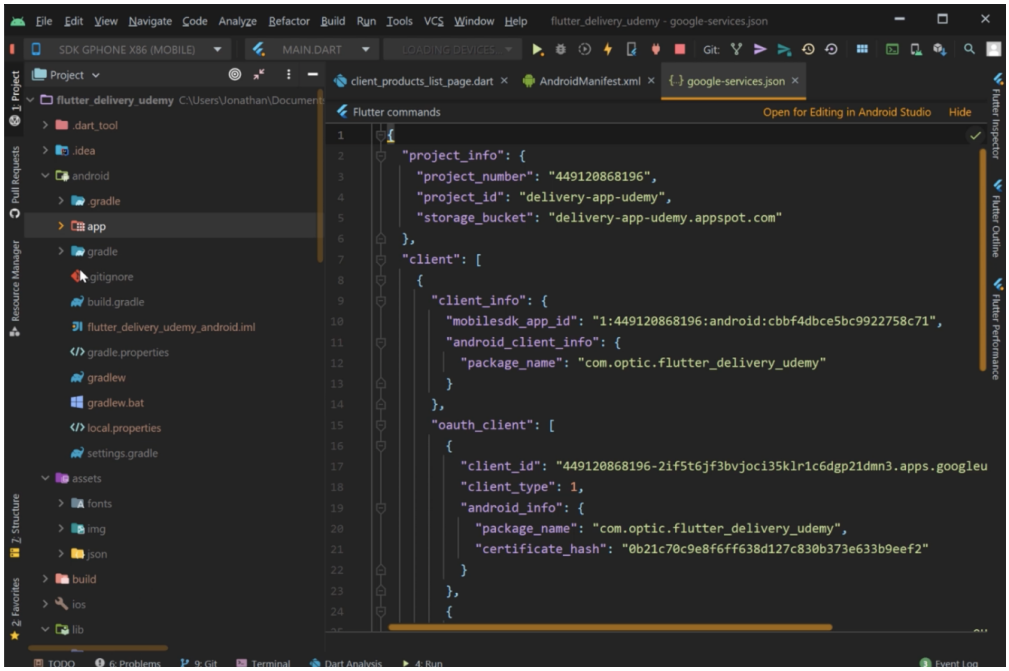


Figura 5.12.3.1.1: Archivo JSON

En la imagen# se muestra dónde debe estar ubicado el archivo Json, dentro de la carpeta Android (APP), se recomienda hacer un refactor una vez ubicado el archivo.

Terminado de hacer todas las configuraciones dentro de Firebase y descargado el archivo Json, lo siguiente es añadir la dependencia para Trabajar con Google Services en el proyecto. Para que no existan errores de compatibilidad, es necesario trabajar con la versión mínima del SDK (21) (agregar los plugin en los archivos que recomienda el propio Firebase).

Con las configuraciones realizadas, todo está listo para trabajar con el almacenamiento de imágenes (dejar la configuración por defecto).

5.13 Almacenamiento de imágenes en la Nube.

El almacenamiento de las imágenes se implementa mediante Backend. Para ello se necesitará de los archivos:

- **async_foreach.js.** Realiza un "for" asíncrono, sirve para esperar hasta que un proceso termine para volver a ejecutarse la siguiente tarea del ciclo, por ejemplo, si se desea almacenar en la base de datos cualquier usuario, el proceso puede tardar un cierto tiempo, porque ejecutaría el proceso realizado y continuará después con la siguiente tarea, Qué sería nuevamente volver a almacenar otro usuario. Eso lo realiza en caso de que existan varios usuarios a la vez.
- **cloud_storoge.js.** Este archivo contiene toda la configuración para subir los archivos a Firebase. Aquí está la lógica para subir los archivos de manera exitosa.
- **ServiceAccount.** Archivo de configuraciones de Firebase.

Estos archivos estarán ubicados dentro del proyecto, en una carpeta denominada "Utils"

Dentro de archivo cloud_storage.js, para subir las imágenes es necesario saber la identificación del proyecto del Firebase, el nombre de archivo clave, y del bucket que es la URL de donde está apuntado para almacenar las imágenes. La URL se encuentra dentro de Firebase.

5.14 Instalaciones en Node.js

Dentro del Node Js, se tendrá instalado:

- **Firebase Admin.** Conjunto bibliotecas de servidor para interactuar con Firebase con entornos de privilegios que permiten ejecutar diferentes acciones:
    - Leer y escribir datos del Realtime Database siempre y cuando tenga privilegios de administrador.
    - Realizar él envió de mensajes de Firebase hacia Cloud Messaging, esto se realizará con una orientación simple.
    - Generar y verificar los diferentes tokens los cuales servirán para la autenticación de Firebase.
    - Permite acceder a los recursos que no entrega Google Cloud (Buckets de Cloud Storage y base de datos de Cloud Firestore).
    - Propia consola de administrador simplificada que permite ejecutar diferentes tareas (Buscar, cambiar dirección de correo electrónico).

    El SDK de Admin para Node Js:

    > Node.js 12 o una versión más reciente. (Firebase Documentation, 2022)

5.15 Cloud Storage

Creado para desarrolladores de aplicaciones, con un servicio de almacenamientos de objetos potente, simple y a la misma vez rentable construido especialmente para el escalamiento de Google.

Los SDK dentro de Firebase, nos ayudaran a agregar seguridad dentro de Google de los archivos tanto de las operaciones de carga y descarga de las Aps de Firebase.

El SDK de Cloud Storage almacena imágenes, audio, video y otros archivos que son generados por el usuario.

5.15.1 Funciones

- *Operaciones Robustas*

  Realizan descargas o pueden cargar información sin importar la calidad de la red. Son cargas y descargas robustas, es decir se inician desde el punto de interrupción de esta manera se ahorrará tiempo y ancho de banda.

- *Seguridad Sólida*

  Los SDK de Firebase los cuales son utilizados para Cloud Storage serán integradas con Firebase Authentication con el único fin de brindar autenticación intuitiva y sencilla para los programadores.

- *Gran escalabilidad*

  Diseñado con el fin de escalar a exabytes, en caso de que la App se vuelva viral.

5.15.2 Ruta de Implementación

➔ Integra los SDK de Firebase para Cloud Storage.
➔ Crea una Referencia.
➔ Sube o descarga.
➔ Protege los Archivos.

5.15.3 Multer

El paquete Multer es el responsable de subir las imágenes para el usuario en nuestra base de datos, que en este caso corresponde al URL de la imagen.

5.16 Proceso de fotografiado o selección de imagen de galería.

Dentro de formulario de registro, se tiene que seleccionar una imagen, ya sea tomando una fotografía directamente o desde el dispositivo o seleccionando una imagen de galería
Para esto se necesita instalar un paquete, ya que se necesita que la dependencia de este paquete sea llamada desde la Aplicación para el uso de las imágenes.

5.16.1 Image picker flutter

Es un paquete complementario de Flutter utilizado tanto para iOS como para Android, el cual permite elegir imágenes desde una biblioteca de imágenes y además tomar nuevas fotos desde la cámara.

iOS (Plugin iOS 9.0 o superior.)

A partir de la versión 0.8.1, para la implementación de iOS usa PHPicker que permite elegir múltiples imágenes en iOS 14 o superiores. Como resultado de la implementación de PH Picker, se hace imposible elegir imágenes HEIC en el simulador de iOS en iOS 14+. Este es un problema conocido.

Android

A partir de la versión 0.8.1, la implementación de Android admite la selección (múltiples) imágenes en Android 4.3 o superior.

Tanto vídeos como imágenes que son seleccionadas son guardados en la caché local de la aplicación y, los mismo que deben esperar que estén disponibles temporalmente. Si necesita que las imágenes sean almacenadas temporalmente es responsabilidad del desarrollador moverle a una ubicación más permanente.

Con un método programado dentro de la Aplicación, que permitirá en primer lugar que muestre o indique qué acción quiere realizar el usuario, es decir, si seleccionar la imagen de elegir o tomar la fotografía.

El método *showAlertDialog* que permite que se puede elegir la imagen desde la galería o permitirá abrir la cámara:

La acción *galleryButton* sirve para abrir la galería de imágenes.

```
void showAlertDialog() {
Widget galleryButton = ElevateButton (
onPressed: () {
selectImage (ImageSource.gallery);
},
child: Text ('GALERIA')
);
```

Dentro del mismo método *showAlertDialog*, la acción de *cameraButtun*, permite abrir la cámara del dispositivo.

```
Widget cameraButton = ElevateButton (
onPressed: () {
selectImage (ImageSource.camera);
},
child: Text ('CAMERA')

};
```

Con la propiedad *AlertDialog*, se llamará a las dos acciones que se crearon de la siguiente manera.

```
AlertDialog alertDialog = alertDialog (
 title: Text('Seleccionar Imagen'),
 actions: [
galleryButton
cameraButton
],
);
```

Con el paquete Image picker que tiene la finalidad de abrir la galería o la cámara, simplemente se creará el método asíncrono *Future selectImage*.
Es necesario declarar las variables globales para hacer uso en el método.

```
PickedFile pickedFile;
File imageFile;
Function refresh;
```

Método

```
Future selectImage (ImageSource imageSource) async {
pickedFile = await ImagePicker().getImage(source: imageSource);
if (pickedfile != null){
      imageFile = File(pickedFile.path);
}
Navigator.pop(context);
refresh();
}
```

Estos métodos permiten realizar las acciones de cargar una imagen dentro del formulario del registro o a su vez abrir la cámara para poder cargar la imagen que se desea subir a la App.

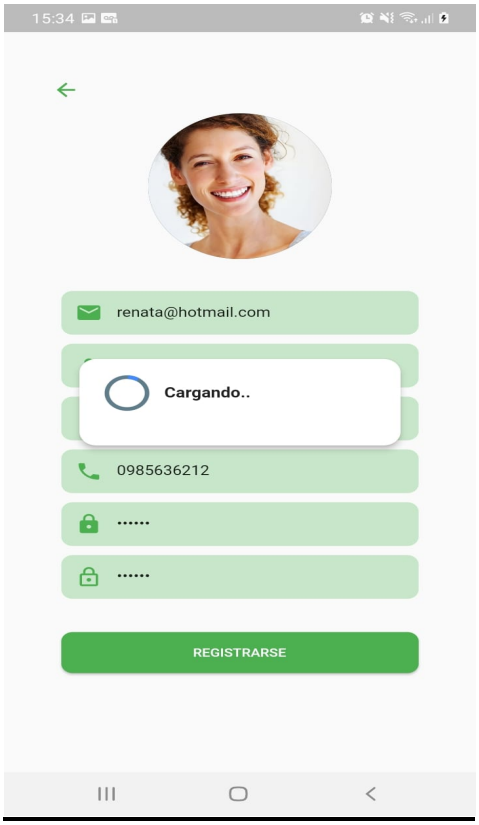
5.17 Proceso de Espera

Progress Dialog. Paquete ligero, que sirve para mostrar un diálogo en proceso. Tiene la misma funcionalidad de un Widget con estado, también permite cambiar el texto que se muestra en el diálogo dinámicamente.

Dependencia progress_dialog: ^1.2.4

5.18 Proceso para crear un usuario con imagen

En la Imagen # Se observa cómo se vería el usuario en el formulario de registro una vez que se haya, al momento que el usuario se registra, se muestra en cuadro de diálogo del paquete Progressdialog que nos indica que el usuario está siendo registrado correctamente en la aplicación.



Figuras 5.18.1: Proceso de registro

5.19 Proceso para editar un usuario

Cuando el usuario se haya registrado y este haya sido ingresado los datos mal, o se necesite hacer un cambio de rol, tendrá una opción que le permite editar sus datos dentro de la App, además se mostraron los datos del usuario en la pantalla de perfil de editar perfil, esto para hacer una actualización dentro de la base de datos de los datos del usuario, con eso se podrá guarda en la sesión los datos del usuario actualizado.

5.20 JWT (JSON Web Token)

JWT (JSON Web Token) Es un estándar de código abierto. Se pueden confiar y verificar contenido token cuando estos estén firmados digitalmente (JWS, RFC 7215). Estas firmas son generadas usando claves simétricas o a su vez claves asimétricas. Los JWT también pueden contener datos cifrados (JWE, RFC 7516) que protegen los datos sensibles.

5.20.1 Tipos de tokens y casos de uso.

Tipos de Tokens:

- Data token: forma serializada JWT, son fáciles y muy trabados a la hora de transmitir en peticiones HTTP y que son usadas principalmente para cuando este realice el intercambio de datos.
- ID token: Gestor de entidades que emite diferentes peticiones de una aplicación cliente, esto atreves de que el usuario sea autenticado. Con esto se logra a que la aplicación permita al cliente obtener los datos del usuario de manera segura y confiable, esto, sin que sus credenciales sean gestionadas.
- Access Token: Conocido como servidor de autorización a peticiones de una aplicación cliente, lo que autoriza el acceso a los recursos que son protegidos en nombre de un usuario. Acces token un método de autorización y autenticación por parte de la aplicación cliente, que permite al servidor alojar sus recursos.

JWT va a permitir que se realicen distintos intercambios de datos de manera segura, entre partes de manera más eficiente que otros estándares (SAML) esto debido a que tiene un tamaño menor. Lo que lo hace primordial para los diferentes casos de uso que se describen a continuación.

- Intercambio de datos de sesión entre servidor y cliente: Debido a su mecanismo de serialización que lo hace único, en ciertas ocasiones, entre el servidor y el cliente, son usados dentro de la sesión y estado para transmitir su información.
- Autenticación federada: Elimina la necesidad de aplicaciones para administrar los inicios de sesión de diferentes usuarios. El mismo delega el proceso de autenticación al administrador de una entidad de confianza. El controlador generado por el token puede ser verificado por la aplicación que contiene los datos de usuario requeridos.
- Autorización de accesos: La información que contiene el token es requerida para que un servidor de APIs, con eso se logra evaluar si la operación que es solicitada por el tenedor del token se puede permitir.

Cada caso de uso tiene distinto destinatario (aplicación cliente y servicio de API), pero en el caso de que se ejerza control simultáneo sobre las dos, un único token puede ser usado para ambos casos. (Hurtado, 2020)

5.21 Modulo de categorías y productos

En la imagen #. se puede visualizar el diseño de la pantalla para crear categoría, con los campos de: Nombre de la categoría, y la descripción de la categoría que tiene como máximo ingresar 255 caracteres, y por último el botón crear, que al momento que la categoría se crea, se pueda visualizar dentro de la Base de datos y dentro de la propia aplicación la creación de la categoría.

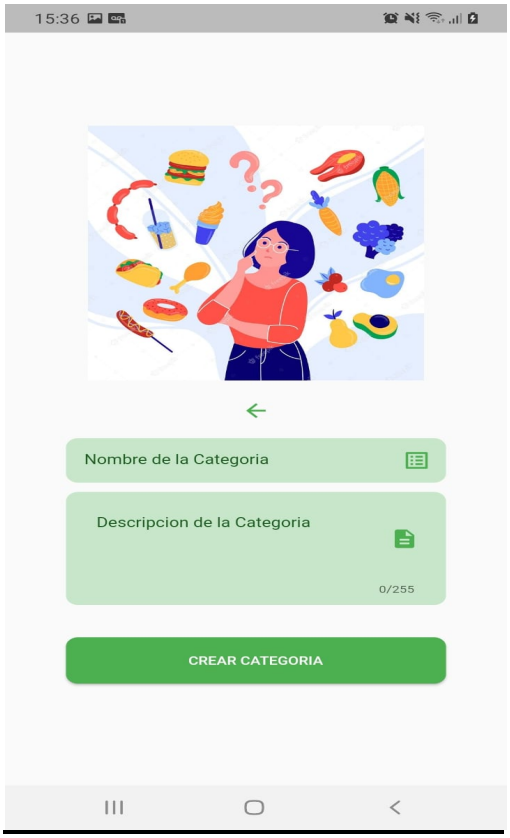


Figura 5.21.1: Pantalla crear categoría.

5. 21.1 Backend para la creación de nuevas categorías.

En el archivo "**db.sql"** que tiene la creación de las tablas que se necesitan para que la aplicación guarde, se tiene que realizar el mismo proceso con la Tabla de Categorías, es decir que, por el momento no existe una tabla categorías por lo que es necesario de crearlo para así crear y almacenar en la base de datos.

Si la tabla ya está creada, ya sea el motivo de que se lo hizo por realizar pruebas anteriores, es recomendable eliminarlo de manera que no exista errores que puedan marcar al momento de crear la tabla con el mismo nombre.

Eliminar la Tabla categoría (En caso de haber sido creada anteriormente)

```
DROP TABLE IF EXISTS categorías CASCADE;
```

Crear la tabla Categoría

```
CREATE TABLE categorias (
id BIGSERIAL PRIMARY KEY,
nombre VARCHAR(180) NOT NULL UNIQUE,
descripcion VARCHAR(255) NOT NULL,
crear TIMESTAMP(0) NOT NULL,
actualizar TIMESTAMP(0) NOT NULL);
```

Para el id se trabaja con el tipo de dato **BIGSERIAL,** ya que permite que al momento que se vayan almacenado nuevas categorías esto permite que en número vaya incrementando de acuerdo a cuantas categorías sean creadas.

Una vez que se ha creado la Tabla dentro del archivo "db.sql" se procede a realizar el mismo proceso dentro del PgAdmin para así se pueda verificar si la secuencia sql está realizada de manera correcta.

Ahora dentro de la carpeta de modelos, vamos a crear un nuevo archivo "category.js" que es el que nos permite trabajar con las consultas en la base de datos, por lo tanto, se requerirá del paquete "config" el archivo "config.js"

Se agregan los métodos que se quieren utilizar en este caso, para la aplicación se agrega los métodos de "create" que recibe la función de una categoría. Dentro de este método se creará una nueva tupla SQL que en la base de datos esto representaría la tabla de categorías que integran la inserción de los datos de categoría.

Al momento que los datos son enviados se retornan como resultado una o ninguna respuesta.

5.21.2 Controlador de Categorías

El controlador nos va a permitir recibir los datos que son enviados de Flutter o Postman que van a ser creados dentro de la base de datos o a su vez representar los errores que se pueden presentar al momento de crear las categorías.

Postman Categorías. Es necesario que se realicen pruebas con una petición a Postman que en este caso sería el cliente que nos permite hacer el ingreso de los datos.

5.21.3 Crear una nueva categoría desde flutter

Para crear una nueva categoría desde flutter, se utiliza la herramienta app.quicktype.

App.quicktype. Nos ayuda a generar modelos y serializadores a partir de JSON schema y GraphQL que nos van a permitir trabajar de forma segura y rápida en cualquier lenguaje de programación. (Quicktype, 2018).

En Flutter también se va a disponer de los mismos campos que anteriormente se trabajó para la creación de nuevas categorías es decir se tendrán los campos:

- id,
- nombre y,
- descripción.

Es necesario que las conexiones que se realizan tanto en Node.JS y Flutter estén configurados con los puertos correspondientes en cada uno de los puertos de los servidores para que los datos que se ingresen al momento de crear una nueva categoría sean almacenados tanto en la base de datos, o cuando el cliente (Postman) mande nuevas solicitudes con nuevos datos.

Para que los datos estén almacenados dentro de la base de datos es necesario tener una autorización, sin dicha autorización del Token, los datos no podrán ser cargados y enviados con la autorización http que es la dirección IP del ordenador.

Los datos son almacenados y validados.

5.21.4 Backend para almacenar productos

5.22 Direcciones y Google Maps

Para el desarrollo de la aplicación y su funcionamiento de direcciones, se dispondrá con:

- **Dirección:** como tal de la residencia del cliente, el barrio o vecindario,
- **Latitud, Longitud:** punto de residencia para el repartidor**.**

5.22.1 Registrar o crear una nueva dirección

En la figura 5.19.1.1 Se visualiza que el usuario que está registrado en la aplicación necesita registrar una nueva dirección el mismo que será presentado al motorizado para la entrega.

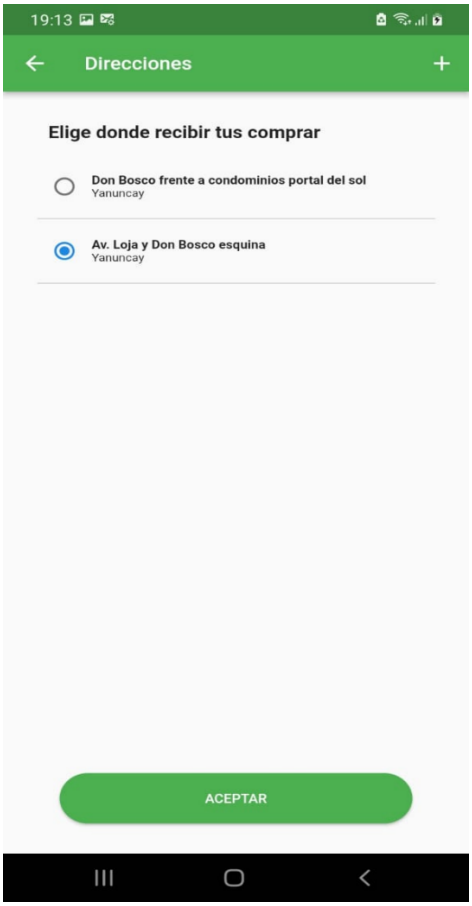


Figura 5.22.1.1: Pantalla de direcciones

Si el usuario tiene ya registrada una o más direcciones dentro de la aplicación, estas direcciones quedarán guardadas, para así hacerle más fácil especificar dónde desea recibir la entrega de la compra realizada.

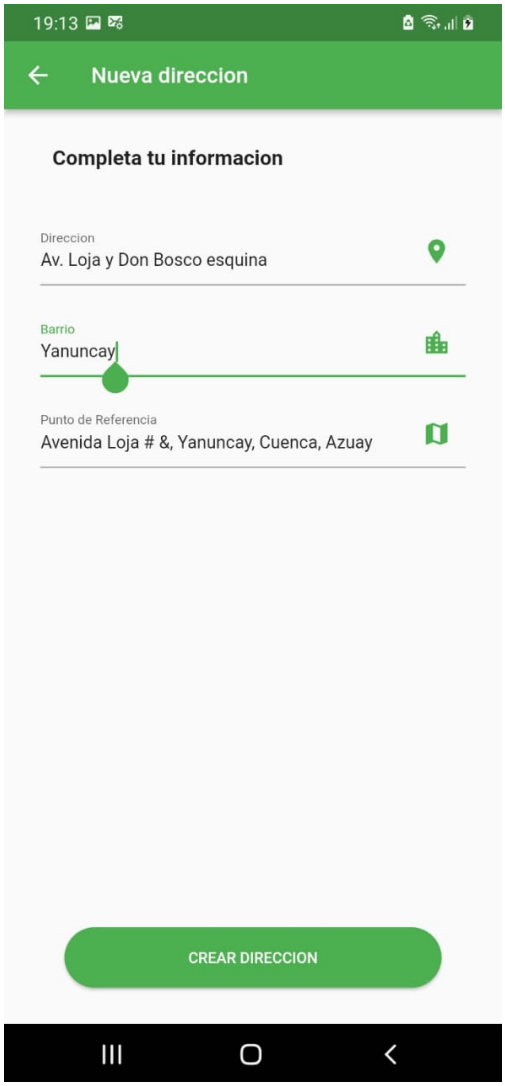


Figura 5.22.1.2: Registro de nueva dirección.

Al momento que el usuario registre una nueva dirección, se le presentará la siguiente pantalla el mismo que deberá ser ingresar la dirección, el barrio y el punto de referencia, que facilitará al motorizado tener el lugar exacto donde realizar la entrega.

5.22.2 Integración de Google Maps en Flutter

Google Cloud Platform (GCP). "Tiene una similitud con Amazon Web Services y Microsoft Azure ya que estas son nubes públicas. Se basa en una estructura masiva y vanguardista de Google, encargada de manejar el tráfico y la carga de trabajo de los usuarios de Google" (Netec, 2019)

Herramientas que se utilizan al momento de usar Google Console para la integración en flutter.

Maps SDK for Android. "Con el SDK de Maps para android, podemos agregar mapas a las diferentes Aplicaciones especialmente para android, incluidas las Apps para Wear OS, que utilizan mapas, datos y respuestas gestuales de Google Maps, permite también ofrecer información adicional sobres las ubicaciones del mapa y facilita la interacción con los diferentes usuarios si se le agrega marcadores, polígono y superposiciones a un mapa. El SDK es compatible con los lenguajes de programación Kotlin y Java" (Google Maps Platform, 2022)

Maps SDK for iOS. "El SDK gestiona de manera automática los diferentes accesos a los servidores de Google Maps, la visualización del mapa y la respuesta a los gestos del usuario. Se puede además añadir marcadores, polilíneas, superposiciones de terreno y ventas de información adicional para las ubicaciones del mapa y permiten la interacción del usuario con el mapa" (Google Maps Platform, 2022)

Para el uso de estas dos herramientas que ofrece Google Cloud Platform, es necesario siempre crear la clave de API, el mismo que sirve para conectar la implementación de con flutter. Con eso podemos usar el mapa de Google.

En la integración de la app con Google Maps se debe de instalar la dependencia.

```
dependencies:

google_maps_flutter: ^2.1.2
```

Con la dependencia instalada en flutter se tendría listo para su uso.

5.22.3 Implementación en Flutter

- Obtener la clave API (Maps SDK for Android y iOS clave generada)
- Habilitar Los SDK
- Seleccionar el proyecto de la App
- Seleccionar el menú de navegación y luego seleccionar "Google Maps"
- Seleccionar API de Google Maps
- Habilitar Google Maps para Android, seleccionar "Maps SDK para Android" en la sección "API adicionales" y, a continuación, seleccionar "Habilitar".
- Habilitar Google Maps para iOS, seleccionar "Maps SDK para iOS" en la sección "API adicionales" y luego seleccionar "HABILITAR".
- Asegurarse que las API que se habilitaron, este "API habilitadas"

Android. La aplicación solo estará disponible para los usuarios que ejecuten Android SDK 2.0 o superior.

Dentro del archivo de configuración "AndroidManifest.xml" del proyecto de la Aplicación se debe ingresar la API o la clave que se generó de los SDK.

```
<meta-data
  android:name="io.flutter.embedding.android.SplashScreenDrawable"
  android:resource="@drawable/launch_background"
  />

<meta-data android:name="com.google.android.geo.API_KEY"
    android:value="YOUR KEY HERE"/>

<intent-filter>
    <action android:name="android.intent.action.MAIN"/>
    <category android:name="android.intent.category.LAUNCHER"/>
</intent-filter>
</activity>
```

Figura 5.23.3.1: Clave API Generada

```
<meta-data android:name="com.google.android.geo.API_KEY"
android:value="AIzaSyBc4KDfLy7szeKtT9T6CbvZjChCXTnJeq0"/>
```

iOS. Para la integración se requiere de un plugin iOS 9.0 o superior. Se debe importar Google Maps.

```
@UIApplicationMain
@objc class AppDelegate: FlutterAppDelegate {
  override func application(
    _ application: UIApplication,
    didFinishLaunchingWithOptions launchOptions: [UIApplication.LaunchOptionsKey:
  ) -> Bool {
    GMSServices.provideAPIKey("YOUR KEY HERE")
    GeneratedPluginRegistrant.register(with: self)
    return super.application(application, didFinishLaunchingWithOptions: launchOpti
  }
}
```

Figura 5.22.3.2: Clave API iOS

Con la clave de la API de los SDK tanto como de Android y iOS, son las configuraciones nativas que necesita para el uso de Google Maps

5.22.3.1 Pasos para implementar los Mapas en Flutter

Importación de los paquetes de Google Maps

- GoogleMap
- CameraPosition

Para este caso necesario de un punto Inicial de referencia, es decir una Latitud y longitud.

```
CameraPosition initialPosition =
CameraPosition(target: LatLng(-2.912720, -
79.021440),zoom: 14 );
```

Controlador

```
Completer<GoogleMapController>
_mapController = Completer();
```

Método

```
void
onMapCreated(GoogleMapController
controller){
_mapController.complete(controller
);
}
```

Widget para Google Maps en la aplicación

```
Widget _GoogleMap () {
return: GoogleMap(
mapType: MapType.normal,
initialCameraPosition:
_con.initialPosition,
onMapCreated: _con.onMapCreated);
}
```

Una vez realizada todas las configuraciones dentro de flutter, se podrá visualizar el mapa dentro de la aplicación.

Obtener Ubicación actual

Para que la aplicación desarrollada en flutter es recomendado usar Geolocator flutter, que permite obtener las ubicaciones en tiempo real, tanto como coordenadas

exactas del dispositivo siempre y cuando el GPS este habilitado; a continuación, se detalla cómo funciona:

Geolocator flutter

Proporciona un fácil acceso a los servicios de localización específicos de la plataforma

Características

- Obtener la última actualización conocida.
- Ubicación actual del dispositivo,
- Actualizaciones continuas de ubicación,
- Comprobar que todos los servicios de localización están habilitados en el dispositivo,
- Calcular la distancia (en metros) entre dos geo-coordenadas.
- Calcular el rodamiento entre dos coordenadas.

Geolocalizador en la aplicación flutter.

Android. Es necesario actualizar los proyectos anteriores a 1.12 para Android.

Desde la versión 5.0.0, este plugin simplemente utilizando la API de Plugin flutter 1.12 para Android. Desafortunadamente esto significa que los desarrolladores de aplicaciones también necesitan migrar sus aplicaciones para admitir una nueva infraestructura de Android. En caso de no realizarlo, se puede procurar comportamientos inesperados.

iOS y macOS. Para iOS y macOS, es necesario agregar nuevas entradas en el archivo info.plist, de esta manera se puede acceder a la ubicación del dispositivo.

Web. En Web se necesita el plugin del Geolocalizador sea Flutter 1.20 o superior.

Windows. Para utilizar el plugin del Geolocalizador en Windows es necesario utilizar Flutter 2.10 o superior.

La dependencia para Geolocator Flutter a usar en la Aplicación es la siguiente.

dependencies:

geolocator: ^7.0.3

Ejemplo de cómo Obtener la posición actual del dispositivo, comprobando si los servicios de localización están habilitados (GPS), comprobando y solicitando permiso a la Dispositivo para acceder a su posición.

```
import 'package:geolocator/geolocator.dart';
Future<Position> _determinePosition() async {
bool serviceEnabled;
LocationPermission permission;
serviceEnabled = await Geolocator.isLocationServiceEnabled();
if (!serviceEnabled) {
return Future.error('Location services are disabled.');
}
permission = await Geolocator.checkPermission();
if (permission == LocationPermission.denied) {
permission = await Geolocator.requestPermission();
if (permission == LocationPermission.denied) {
return Future.error('Location permissions are denied');
}
}
if (permission == LocationPermission.deniedForever) {
return Future.error(
'Location permissions are permanently denied, we cannot request
permissions.');
}
return await Geolocator.getCurrentPosition();
}
```

Plugin Location Flutter. Este plugin para Flutter se encarga de obtener una ubicación en Android e iOS. También proporciona devoluciones de llamada cuando se cambia la ubicación. (Pub dev, 2021)

Dependencia recomendada utilizar

dependencies:

location: ^4.1.1

Con el GPS habilitado y realizadas las configuraciones, se logra visualizar la actualización actual de dispositivo o de donde se desea realizar el pedido.

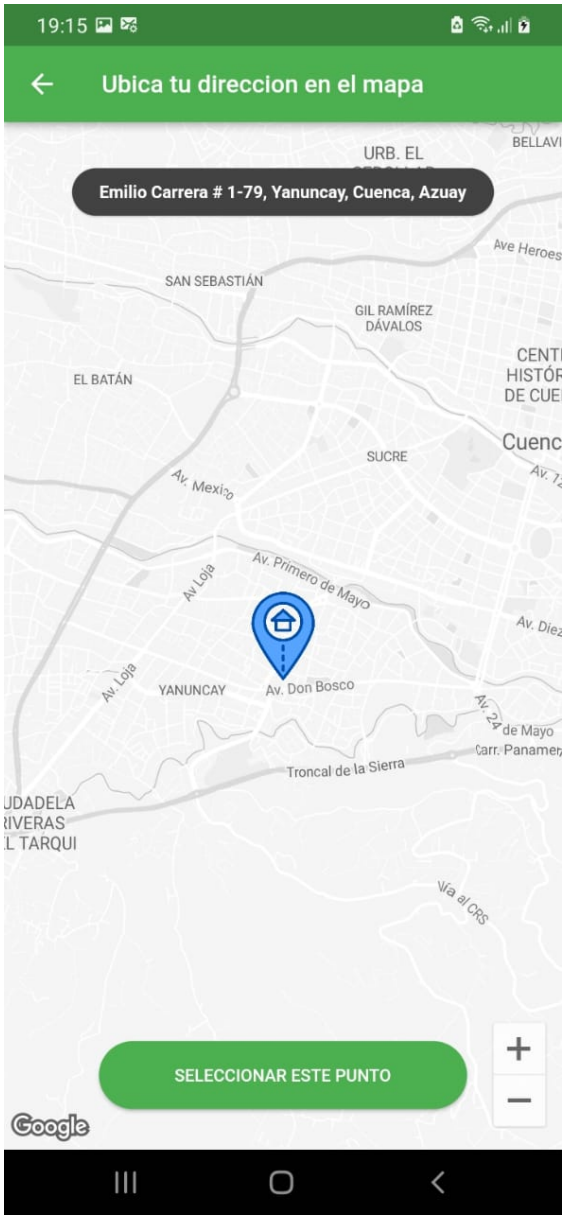


Figura 5.22.3.1.1: Ubicación actual

En la imagen #. se observa la ubicación actual de donde se encuentra el dispositivo, marcando así una latitud y longitud del mismo y lo que le ayudar al motorizado saber dónde y a qué distancia se encuentra para hacer el pedido.

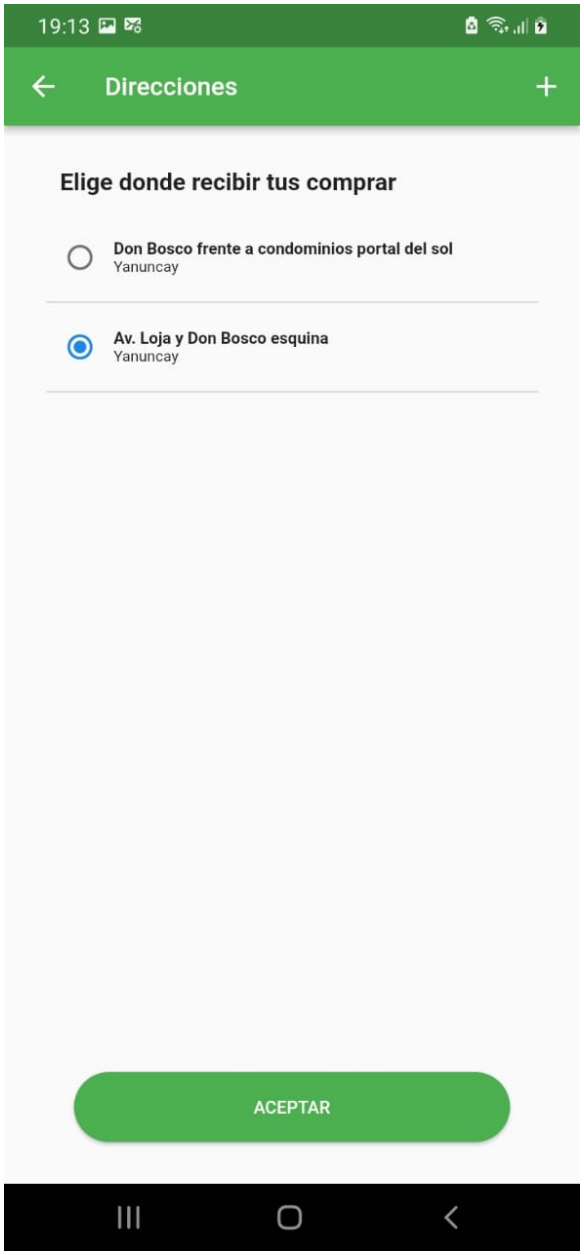


Figura 5.22.3.1.2: Direcciones del Usuario

Cuando se hayan registrado algunas direcciones, estas serán almacenadas en la base de datos, los mismo que el usuario tenga la opción de marcar la misma dirección donde requiera realizar el pedido, o en caso contrario se procederá a crear una nueva dirección.

5.22.4 Ruta trazada

En la Imagen 2.7 se puede visualizar la ruta por donde se va a dirigir el motorizado marcando así el tiempo, la distancia y el recorrido que va a realizar, si existen cambios o el motorizado decida irse por diferente camino la ruta permanecerá

trazada pero no marcará en el mapa, pero si es posible verificar la ubicación exacta del motorizado, así existe la posibilidad de verificar que este dirigiéndose al destino que originalmente se estableció.

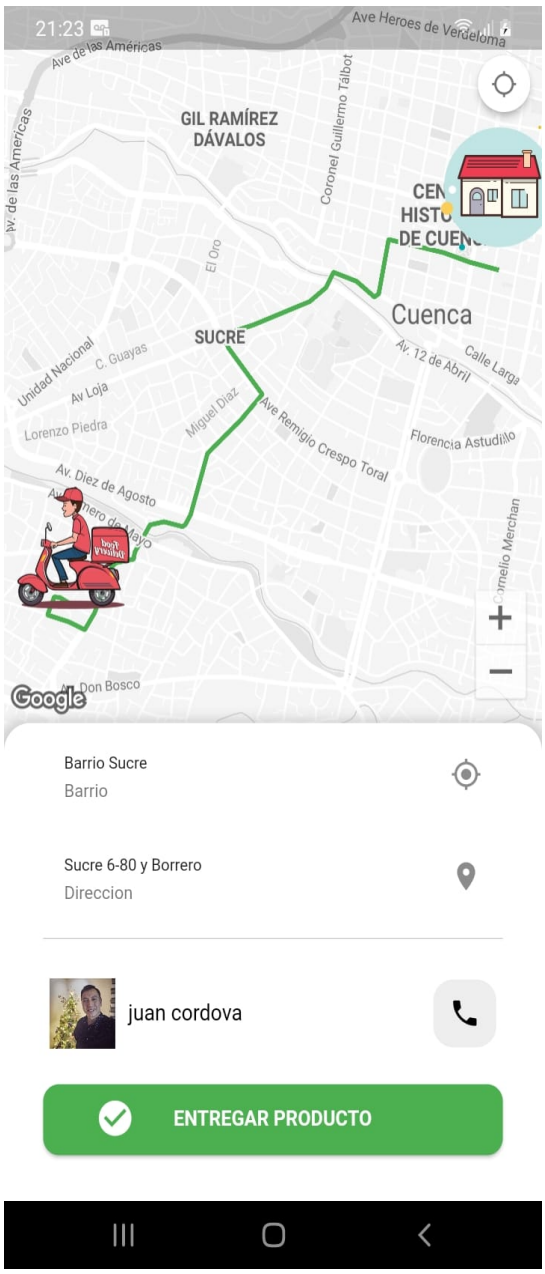


Figura 5.22.4.1: Ruta del Motorizado

Para que se muestre la ruta trazada se utilizaran herramientas que se detalla a continuación.

Directions API. API de direcciones, es un servicio web que utiliza una solicitud HTTP, que permiten devolver direcciones JSON o con formato XML entre

ubicaciones. Permiten recibir indicaciones para diferentes modos de transporte, como el tránsito, la conducción, el senderismo o ciclismo. (Google Maps Platform, 2022)

Flutter Polyline points. Plugin de aleteo, codifica la cadena o polilínea de Google, permite la codificación de una lista de coordenadas geográficas adecuadas para mostrar ruta/polilínea en mapas. Además, que es un paquete que contiene funciones que permitieran la decodificación de una cadena polilínea codificada en Google, el mismo que va a devolver una lista de coordenadas que indica la ruta entre dos posiciones geográficas. (Pub dev, 2021)

La dependencia que se usa en el desarrollo de la App, es la siguiente.

```
dependencies:

flutter_polyline_points: ^1.0.0
```

Para realizar llamadas por parte del repartidor al Cliente es necesario instalar el paquete:

url_launcher 6.0.20. Permite a las aplicaciones, inicializar navegadores web, aplicaciones de mapas, aplicaciones de marcadores y aplicaciones de correo. Funciona creando intenciones para abrir aplicaciones utilizando diferentes esquemas de URL. (Ahmed, 2022)

5.22.5 Obtener la posición del repartidor en tiempo real.

En la imagen #. se muestra cómo obtener la ubicación actual del repartidor, es decir que al momento que el repartidor cambie la ruta o de demora, se actualiza la posición en la que se encuentra.

```
_positionStream = Geolocator.getPositionStream(
    desiredAccuracy: LocationAccuracy.best,
    distanceFilter: 1
).listen((Position position) {

  _position = position;

  addMarker(
      'delivery',
      _position.latitude,
      _position.longitude,
      'Tu posicion',
      '',
      deliveryMarker
  );

  animateCameraToPosition(_position.latitude, _position.longitude);

  refresh();

});

} catch(e) {
  print('Error: $e');
}
```

Figura 5.22.5.1: Código posición del repartidor.

5.22.6 Implementación Google Maps.

Con la implementación de Google Maps el usuario podrá obtener una mejor navegación.

Método que permite abrir las aplicaciones de Google Maps, este método va a requerir de una Latitud y longitud.

5.22.7 SOCKET IO Tiempo real de la Aplicación.

```
void launchGoogleMaps() async {
  var url =
'google.navigation:q=${order.addres.lat.toString()},${order.add
res.lng.toString()}';
  var fallbackUrl =

'https://www.google.com/maps/search/?api=1&query=${order.addres
.lat.toString()},${order.addres.lng.toString()}';
  try {
   bool launched =
   await launch(url, forceSafariVC: false, forceWebView:
false);
   if (!launched) {
    await launch(fallbackUrl, forceSafariVC: false,
forceWebView: false);
   }
  } catch (e) {
   await launch(fallbackUrl, forceSafariVC: false,
forceWebView: false);
  }
 }
```

SOCKET IO con esta biblioteca permite la comunicación bidireccional, baja latencia y se basa siempre en eventos entre el cliente y servidor.

Protocolo WebSocket, que proporciona garantías como recurrir a la encuesta prolongada HTTP o la reconexión automática. (Socket.IO, 2022)

Figura 5.22.7.1: Protocolo HTTP cliente servidor

Fuente: https://socket.io/docs/v4/

Socket IO, es usado en la Aplicación para saber el tiempo real de la ubicación del repartidor, es decir permite que el cliente sepa en qué lugar se encuentra el repartidor con el pedido.

Si el repartidor o el Cliente hacen cambios, por ejemplo: Si el repartidor elige ir por un lugar por donde no exista mucho tráfico entonces lo que permite saber la posición actual o si el cliente cambió su lugar de entrega el repartidor podrá saber dónde se encuentra.

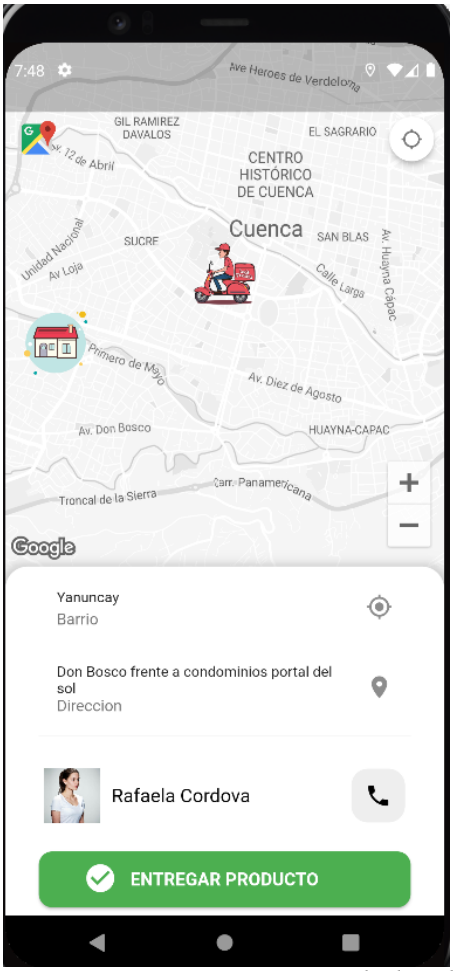


Figura 5.22.7.2: Emulador de Repartidor

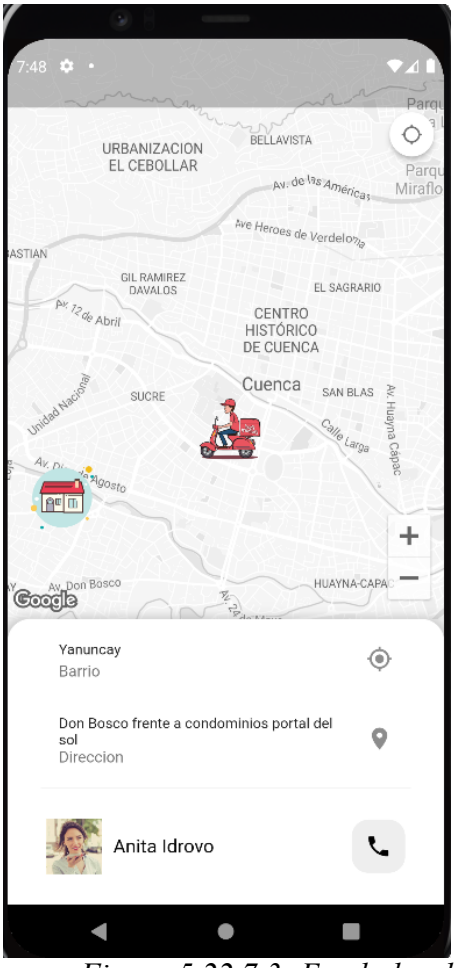


Figura 5.22.7.3: Emulador de Cliente

Archivo JavaScript

Permite al repartidor saber la ubicación actual del cliente cuando se haya traslado de un lugar a otro.

```
Order.updateLatLng = (order) => {
const sql = `UPDATE orders
SET lat = $2,  lng = $3
WHERE
ID = $1
`;
return db.none (sql, [
order.id, order.lat, order.lng
]);
}
```

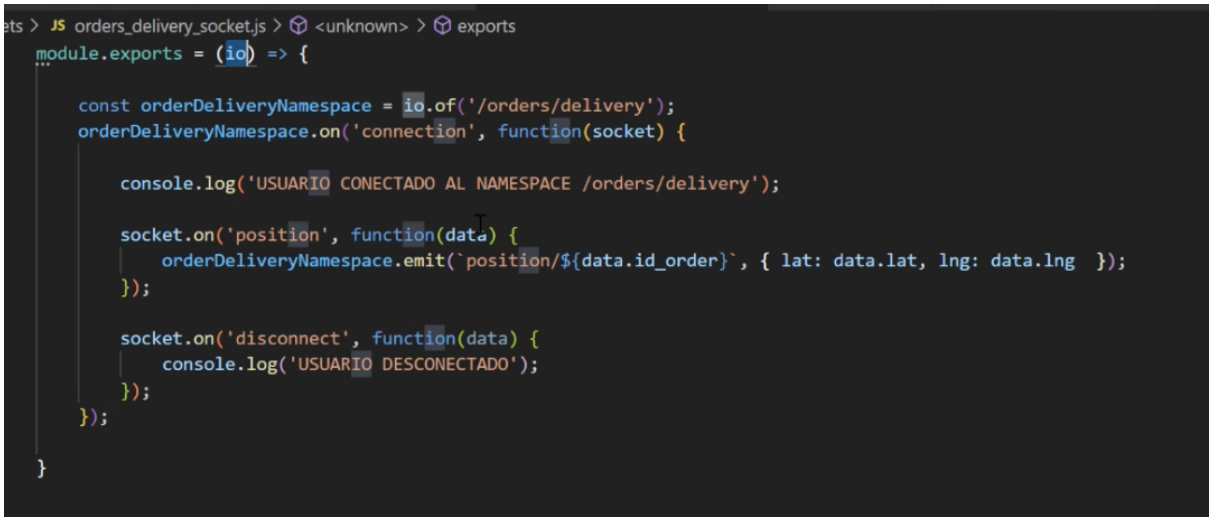


Figura 5.22.7.4: Código ejemplo, ubicación del repartidor

Para el cliente que sepa también la ubicación en tiempo real se utilizara el siguiente paquete.

Flutter socket IO

5.22.8 Guardar la posición actual en la base de datos.

Para realizar la acción de guardar los datos se necesita de una actualización en nuestra base de datos, ya que, al momento de guardar la posición, de lo hará con la latitud y longitud.

Con esto se puede guardar la posición actual del repartidor y cliente en la base de datos.

Una vez se hayan hecho todas las configuraciones se procederá hacer las pruebas correspondientes en la aplicación.

5.22.9 Restricciones

- *Establecer la ruta del motorizado al restaurante.*

  Al momento que exista una petición por parte de cliente, este automáticamente generará una ruta que permita al motorizado visualizar el restaurante que está solicitando sus servicios.

- *Establecer la ruta del restaurante al cliente.*

  Cuando exista la petición por parte del cliente, se procede a realizar el respectivo pedido del cliente al restaurante, por lo cual el motorizado, al

momento que va a realizar el pedido, el mismo será capaz de verificar la ruta que ha marcado la aplicación identificando el destino del producto.

- *Estados de entregado (Pagado, Despachado, En camino, Entregado)*

En la imagen #. Se puede evidenciar que existe ya un pedido que se ha realizado, pero sin repartidor, ya sea porque esté muy lejos o no exista repartidores cerca, al momento que existe un repartidor cerca este será asignado y se marcará la ruta específica que deberá seguir.

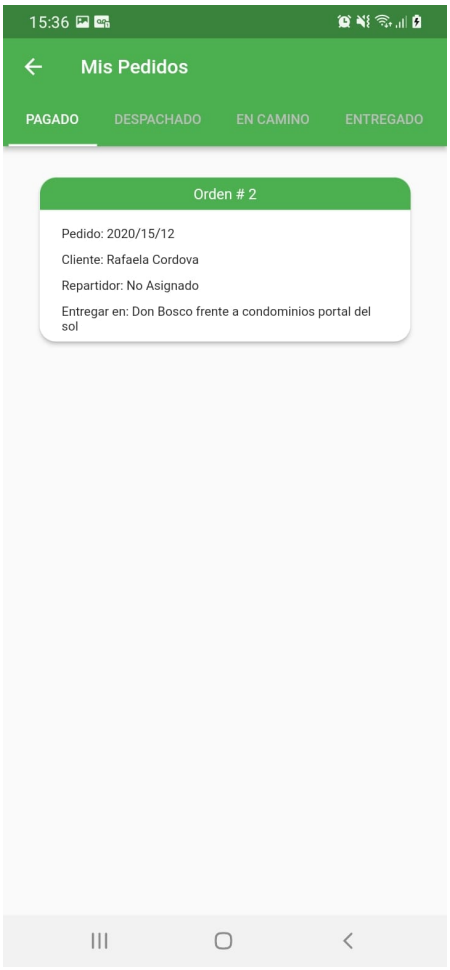


Figura 5.22.9.1: Estado de entregas.

El motorizado contará con los datos siempre y cuando tenga acceso a internet o con el celular prendido.

5.23 Módulo de pedidos para clientes

Creación de una Nueva orden

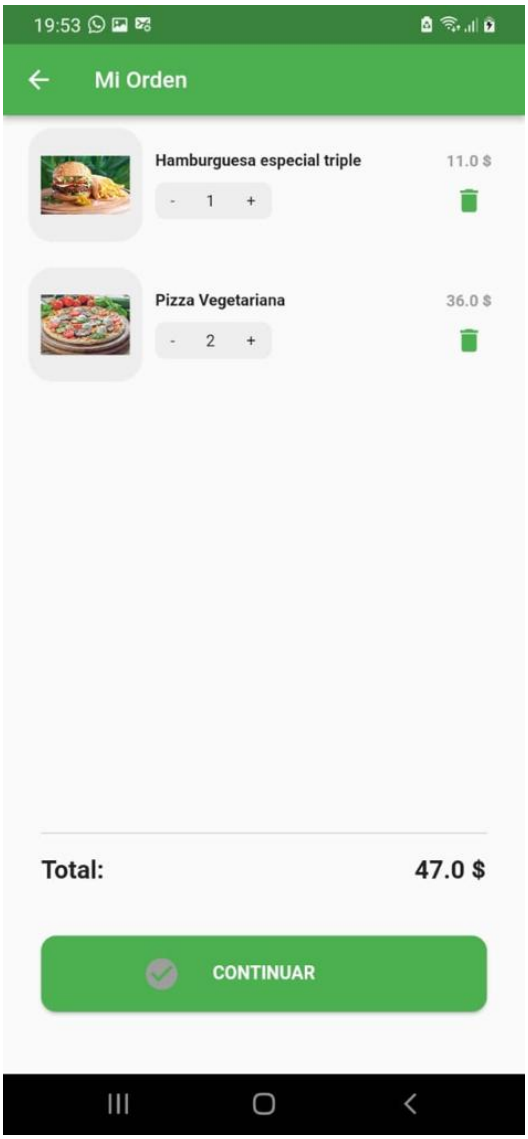


Figura 5.23.1: Pantalla de Órdenes

En la imagen 5.23.1 cuando el cliente realiza una compra o varas compras estos son registrados en nuestra base de datos, y a su vez calculados los precios (total) que se debe de pagar, una vez se ha realizado el pedido, el cliente procederá a elegir el lugar donde se va a realizar la orden, especificando una dirección, y lugar de referencia para que el motorizado entregue la orden.

5.24 Módulo de pedidos para restaurante

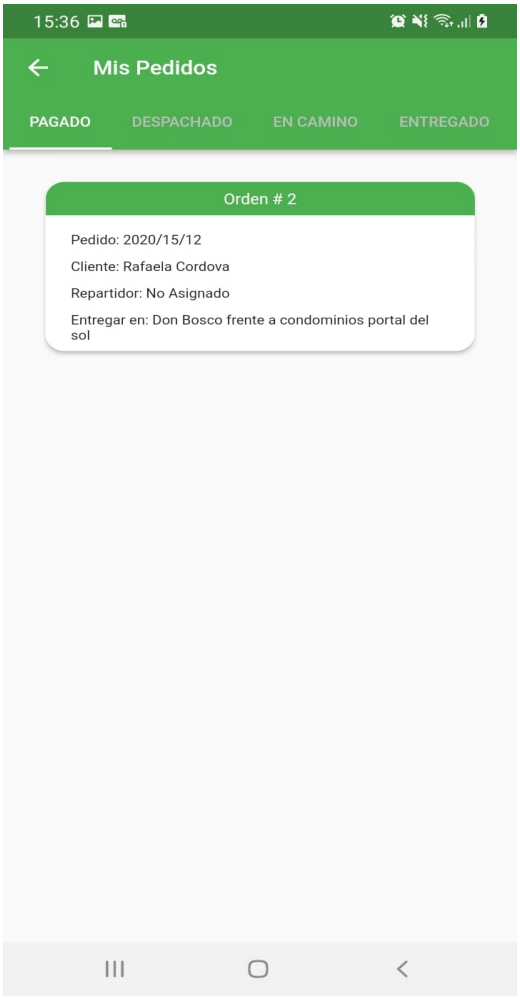


Figura 5.24.1: Pantalla de Pedidos

En la pantalla de pedidos se cuenta con 4 estados:

- *Pagado*

Una vez el cliente haya realizado la compra y este a su vez sea cancelado, aparecerá que la orden ya fue hecha, pero que el repartidor aún no ha sido designado, si el repartidor estuviera en un lugar cerca, esté a su vez asumirá el cargo y procederá a realizar la orden y entregarla.

- *Despachado*

La finalidad de este proceso consiste en que el producto salga del restaurante y sea entregado en su destino final, a tiempo y en perfectas condiciones.

Existen labores como

a. Asegurar que el pedido esté en orden antes del despacho;
b. Planificar todas las entregas que se van a realizar durante el día;
c. Comprobar que la los pedidos y direcciones esté en perfectas condiciones;
d. Destinar al motorizado para la entrega;
e. Especificar la ruta que debe seguir el motorizado.

- *En Camino*

Cuando el pedido se haya realizado, el cliente podrá visualizar mediante la aplicación el lugar exacto donde se encuentra el motorizado realizando el pedido y el tiempo que este se debe de tardar en realizar el mismo.

- *Entregado*

Una vez se haya realizado la orden, en la pantalla de la lista de estados, pasará a un estado de entrega, que se entendería que el motorizado ya realizó la entrega en el tiempo exacto.

Los cuatro estados que se muestran en la aplicación serán consultas realizadas a la base de datos, con la finalidad de saber en qué estado se encuentra la orden, si es entregado.

Esquema de la consulta de la BD en flutter, para listar los estados de los pedidos.

```
Order.findByStatus = (status) =>{
  const sql = `
  SELECT
    O.id,
    O.id_cliente,
    O.id_direccion,
    O.id_delivery,
    O.estado,
    O.timestamp,
    JSON_AGG(
      JSON_BUILD_OBJECT(
        'id', P.id,
        'nombre', P.nombre,
        'descripcion', P.descripcion,
        'precio', P.precio,
        'imagen1', P.imagen1,
        'imagen2', P.imagen2,
        'imagen3', P.imagen3,
        'cantidad', PE.cantidad
      )
    )AS products,
    JSON_BUILD_OBJECT(
      'id', U.id,
      'nombre', U.nombre,
      'apellido', U.apellido,
      'telefono', U.telefono,
      'imagen', U.imagen
    ) AS cliente,
    JSON_BUILD_OBJECT(
      'id', U2.id,
      'nombre', U2.nombre,
```

```
      'apellido', U2.apellido,
      'imagen', U2.imagen
    ) AS delivery,
    JSON_BUILD_OBJECT(
      'id', D.id,
      'direccion', D.direccion,
      'barrio', D.barrio,
      'latitud', D.latitud,
      'longitud', D.longitud
    ) AS direccion
  FROM
    ordenes AS O
  INNER JOIN
    usuarios AS U
  ON
    O.id_cliente = U.id
  LEFT JOIN
    usuarios AS U2
  ON
    O.id_delivery = U2.id
  INNER JOIN
    direccion D
  ON
    D.id = O.id_direccion
  INNER JOIN
    pedidos AS PE
  ON
    PE.id_orden= O.id
  INNER JOIN
    productos P
  ON
    P.id = PE.id_producto
  WHERE
    estado = $1
  GROUP BY
    O.id, U.id, D.id, U2.id
  `;
  return db.manyOrNone(sql, status);
}
```

Este esquema servirá para tener una lista de los estados de los pedidos, permitiendo así obtener la orden, la dirección, el nombre y apellido del cliente, el motorizado que realizará la orden.

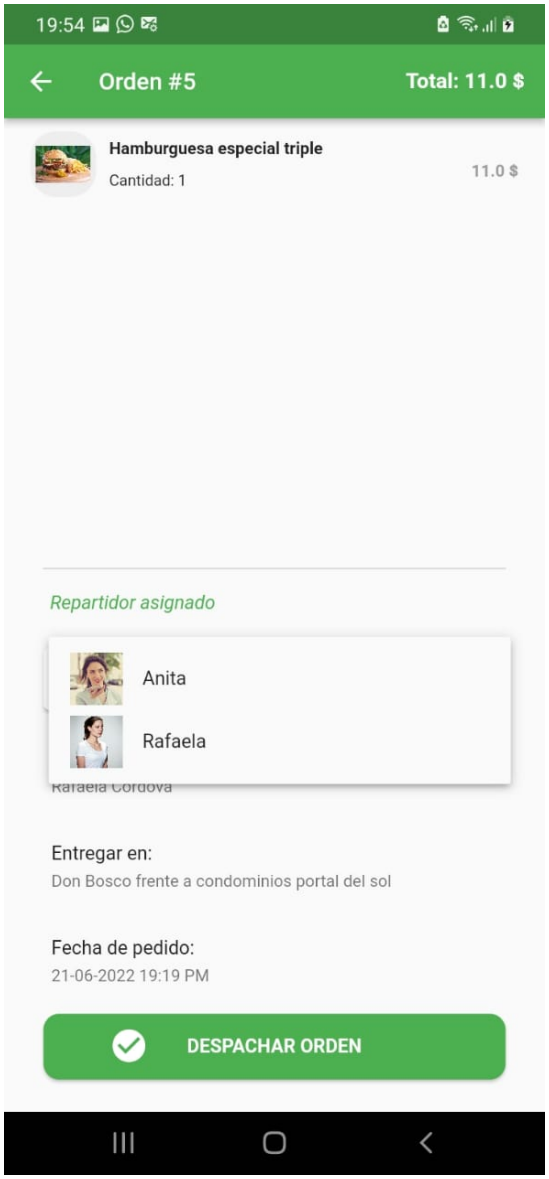


Figura 5.24.2: Lista de Órdenes

Todas las órdenes que se realicen por parte del restaurante se mostrarán detallados como se muestra en la imagen 5.24.2.

De acuerdo al pedido que realice el cliente, el restaurante podrá visualizar la orden con los pedidos que haya hecho el cliente, así como la cantidad de pedidos que se ha realizado.

El restaurante asignará a un repartidor siempre y cuando este esté disponible y se encuentre cerca del establecimiento para que pueda realizar el pedido del cliente, además que el restaurante dentro del detalle de la orden, puede ver cuál es el cliente que ha realizado el pedido, así como el lugar de ubicación, la fecha y hora del pedido, y el total que el cliente deberá de cancelar para que el restaurante puede despachar la orden, una vez detallado el orden, con el motorizado se podrá despachar la orden.

5.25 Listado de repartidores.

Con una consulta a nuestra base de datos, se obtendrá los repartidores que se encuentran registrados ya en la base de datos.

El siguiente esquema que se muestra, permitirá obtener la lista de todos los repartidores.

```
User.findDelivery = () => {
  const sql = `
SELECT U.id, email, U.nombre, U.apellido, U.telefono,
U.contrasena, U.imagen, U.session_token
    FROM usuarios AS U
    INNER JOIN usuario_roles AS UR
    ON UR.id_usuarios = U.id
    INNER JOIN roles AS R
    ON R.id = UR.id_rol
    WHERE R.id = 3
  `;
  return db.manyOrNone(sql);
}
```

Una vez hecha la consulta se mostrarán todos los repartidores que se encuentran ingresados en nuestra base de datos, los repartidores se listan para saber a qué repartidor se le será asignado para que este realice las órdenes del restaurante.

Para listar los repartidores se utilizará un DropDown

En flutter, es un botton de diseño de material. Es un widget que se puede usar para seleccionar un valor único de un conjunto de valores. Por lo que, le permite al usuario seleccionar un valor de varios elementos, el valor predeterminado va a mostrar el valor seleccionado actualmente. (Gayatribkar, 2021)

5.26 Módulos de pedidos para el Delivery.

5.26.1 Mostrar órdenes para un repartidor asignado.

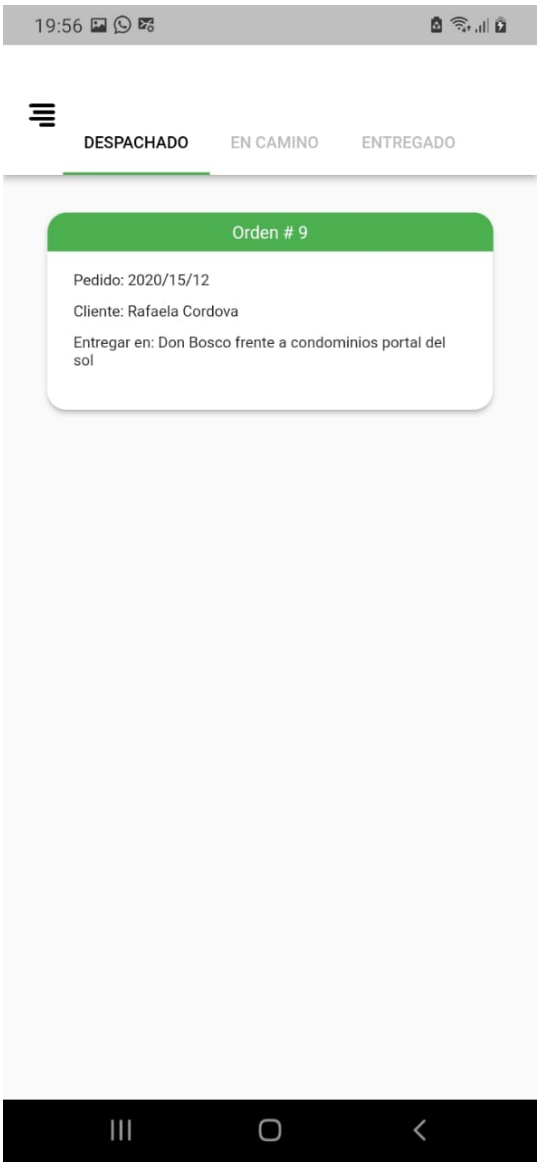


Figura 5.26.1.1: Órdenes del repartidor.

En el caso del repartidor, únicamente cuenta con tres estados a diferencia del Restaurante que trabajaba con los cuatro estados.

Los estados con los que trabaja el repartidor son:

- Despachado.
- En camino.
- Entregado.

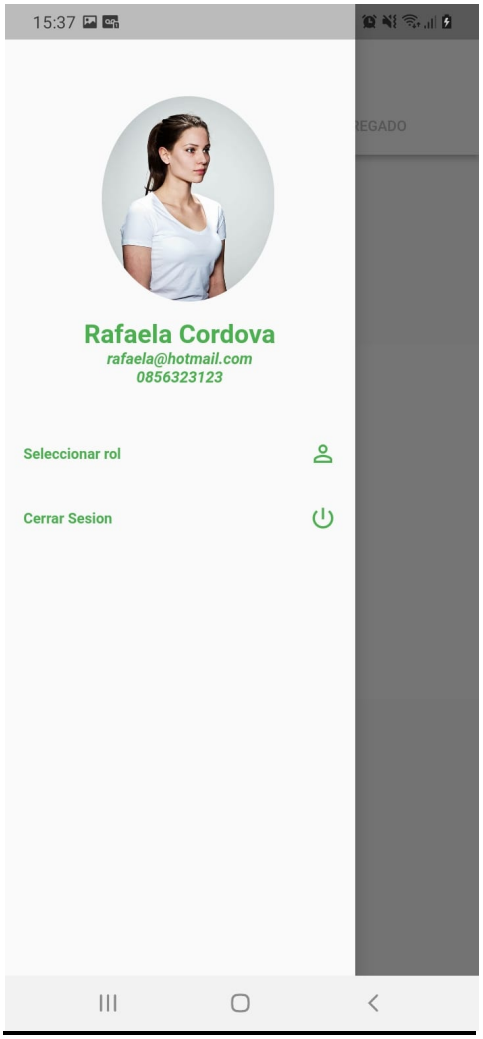


Figura 5.26.1.2: Menú del repartidor.

Para el repartidor contará con el siguiente menú, el mismo que solo cuenta con la selección del Rol, que en este caso al ingresar lo hará de manera de Repartidor, dirigiéndose así a la pantalla de ordenes con los tres estados con los que cuenta.

5.26.2 Actualizar estado “EN CAMINO”

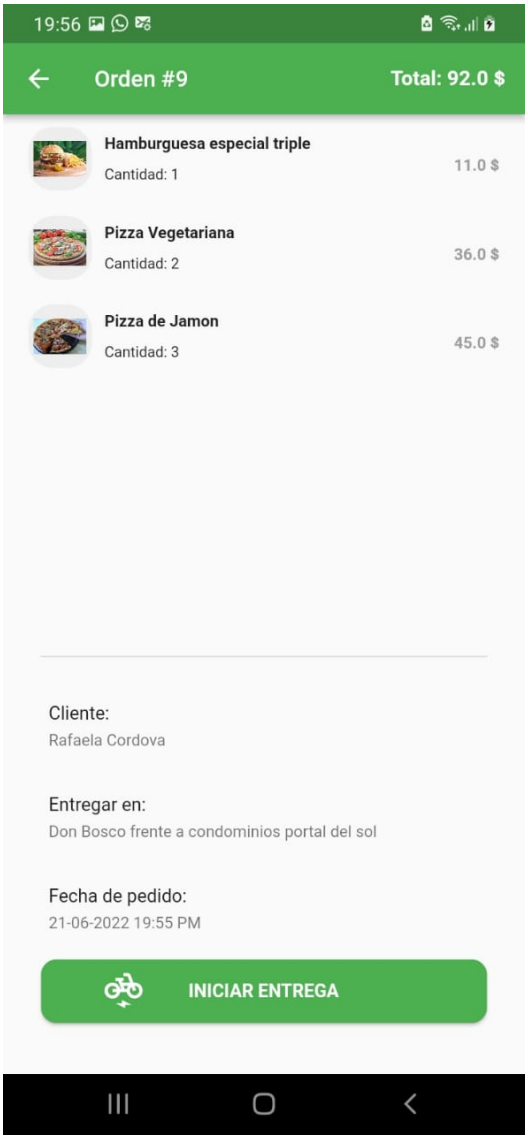


Figura 5.26.2.1: Detalle de la orden.

Al iniciar la entrega este pasará automáticamente a un estado de “EN CAMINO” lo que directamente, pasará a mostrar la ruta que debe de seguir el repartidor desde el restaurante a la ubicación del cliente, mostrando el barrio y la dirección y además el repartidor que realizará la orden.

RESULTADOS

Flutter al ser una herramienta SDK desarrollada por Google, en el 2022 posee unas avances significativos, llegando al punto de tener un rendimiento nativo para el desarrollo de multiplataformas y una notable optimización al momento de realizar aplicaciones, por lo cual hemos podido cumplir con todos los objetivos propuestos desde un principio, llegando a satisfacer nuestras metas propuestas que desde un principio fue el seguimiento del repartidor en tiempo real, además de llegar a categorizar productos para su respectiva búsqueda, crear direcciones en donde utilizamos paquetes muy importantes de Google Map para una mejor ubicación dentro del mapa y sobre todo una ruta más rápida que el repartidor pueda llegar a utilizar.

Con relación a la implementación de la aplicación dentro de cualquier plataforma de ventas tanto de android como de iOS, no lo vamos a realizar aún, el motivo es que la aplicación aún necesita grandes avances que pondrán a la aplicación a competir dentro del mercado de las apps.

En lo referente a los diferentes roles que tenemos en nuestra aplicación cada una tiene sus ventajas, en este caso cuando el repartidor comience su entrega, el cliente como el dueño del negocio podrán realizar simultáneamente el seguimiento del repartidor, así el repartidor no siga la ruta indicada por la aplicación, no se dejara de seguir al repartidor, siempre estará presente en el mapa la trayectoria que utilice.

La implementación de un botón de llamada tanto en el repartidor como en el cliente simplificará la tarea de realizar la llamada al momento de realizar la entrega, esto gracias a que este botón llamará internamente a la clase en donde está almacenado el teléfono del cliente que realizó el pedido y lo podrá llamar de una manera más fácil.

Las diferentes restricciones que hemos implementado en la aplicación ayudarán a la seguridad de la misma, como por ejemplo la contraseña está encriptada, el repartidor no podrá pulsar la opción “Entregar” si no está a menos de 20 metros del lugar de entrega, todas las sesiones están protegidas por un session token, la sesión se cerrará después de 3 minutos de inactividad, todas estas restricciones y muchas otras ayudan a tener una aplicación más optimizada y segura.

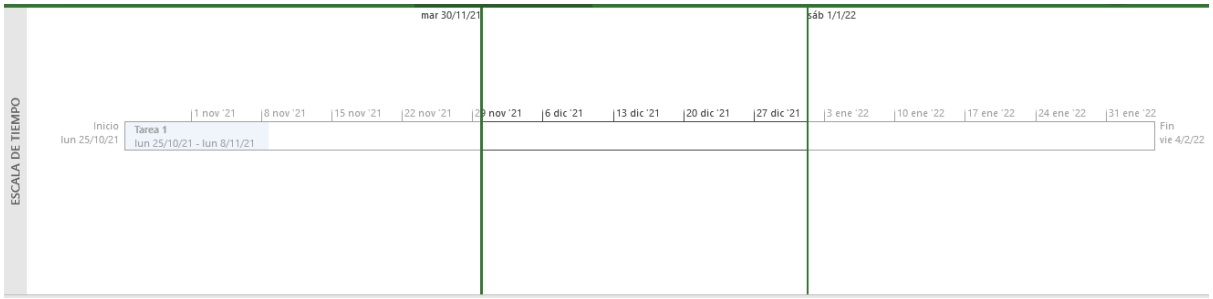
Los diferentes problemas que hemos tenido al momento del desarrollo, se suscitaron principalmente por las diferentes versiones de los paquetes implementados, esto ha causado retrasos al momento del desarrollo, la solución que pudimos darle a los diferentes problemas fue revisar la compatibilidad de los diferentes paquetes con la versión de Flutter.

En cuanto al almacenamiento estamos utilizando software libre en este caso realizamos el almacenamiento de todos los usuarios en PostgreSQL, en cuanto al almacenamiento de imágenes tanto de perfiles como de productos de las diferentes categorías se guardan

directamente en la base de datos de Google Firebase, la ventaja de realizar este proceso se da al momento de llamar a la imagen dentro nuestra aplicación, directamente en la base de datos se guarda únicamente la URL de la imagen, de esta manera se optimiza la app, para una carga de imágenes más rápida.

CRONOGRAMA

	Modo de tarea	Nombre de tarea	Duración	Comienzo	Fin
1		**Proyecto**	**600 horas?**	**lun 25/10/21**	**vie 4/2/22**
2		**Tarea 1**	**86 horas?**	**lun 25/10/21**	**lun 8/11/21**
3		**Inicio Revision 1**	**80 horas**	**lun 25/10/21**	**vie 5/11/21**
4		**OE. 1**	**20 horas**	**lun 25/10/21**	**mié 27/10/21**
5		AC. 1 Estudio de los fundamentos básicos de utilización de sockets para poder implementar la geolocalización.	6 horas	lun 25/10/21	lun 25/10/21
6		AC. 2 Estudio de la implementación de sockets en nuestro framework flutter	6 horas	mar 26/10/21	mar 26/10/21
7		AC. 3 Estudio de la estructura de posicionamiento de Flutter al momento de trabajar con el lenguaje Dart	8 horas	mar 26/10/21	mar 26/10/21
8		**OE. 2**	**60 horas**	**mié 27/10/21**	**vie 5/11/21**
9		AC. 1 Construcción de un método en donde el dueño del restaurante pueda elegir al motorizado que se encuentre más cerca para realizar la entrega.	30 horas	mié 27/10/21	lun 1/11/21
10		AC. 2 Construir un método en donde el usuario pueda escoger al repartidor más cercano para que pueda realizar cualquier envio.	30 horas	lun 1/11/21	jue 4/11/21
11		**Tarea 2**	**185 horas?**	**vie 5/11/21**	**mié 8/12/21**
12		**Inicio Revision 2**	**180 horas**	**vie 5/11/21**	**mar 7/12/21**
13		**OE. 1**	**30 horas**	**vie 5/11/21**	**mié 10/11/21**
14		AC. 4 Establecer tiempos de espera para los diferentes repartidores y tiempos de entrega para los que realizan los envios.	30 horas	vie 5/11/21	mié 10/11/21
15		**OE. 4**	**145 horas**	**jue 11/11/21**	**mar 7/12/21**
16		AC. 1 Estudio acerca de los fundamentos sobre Node JS para poder crear un crud de cliente, repartidor y dueño del negocio.	40 horas	jue 11/11/21	mié 17/11/21
17		AC. 2 Estudio de las diferentes consultas en la base de datos en este caso SQL, para los diferentes procesos de respuestas.	40 horas	mié 17/11/21	mar 23/11/21
18		AC. 3 Estructurar y distinguir los diferentes roles para poder realizar un login y registro en la base de datos utilizando JSON WEB TOKENS	60 horas	jue 25/11/21	lun 6/12/21
19		**Tarea 3**	**200 horas**	**mar 7/12/21**	**lun 10/1/22**
20		**Inicio Revision 3**	**195 horas**	**mar 7/12/21**	**lun 10/1/22**
21		**OE. 3**	**35 horas**	**mar 7/12/21**	**lun 13/12/21**
22		AC. 1 Construir un método que ayude a buscar al repartidor previamente registrado en la aplicación para que realice los pedidos del cliente.	35 horas	mar 7/12/21	lun 13/12/21
23		**OE. 4**	**160 horas**	**lun 13/12/21**	**vie 7/1/22**
24		AC. 3 Estructurar y distinguir los diferentes roles para poder realizar un login y registro en la base de datos utilizando JSON WEB TOKENS	40 horas	lun 13/12/21	vie 17/12/21
25		AC. 4 Estudio de las diferentes peticiones HTTP las cuales se podrán realizar ya sea desde Android o desde IOS	40 horas	vie 17/12/21	jue 23/12/21
26		AC. 5 Estudio de la integración y consulta de direcciones dentro de Google Maps API.	40 horas	lun 27/12/21	vie 31/12/21
27		AC. 6 Construcción de los diferentes modelos de trazado de rutas utilizando API de Google.	40 horas	lun 3/1/22	vie 7/1/22
28		**Tarea 4**	**130 horas**	mar 11/1/22	**mié 2/2/22**
29		**Inicio Revision 4**	**125 horas**	**mar 11/1/22**	**mar 1/2/22**
30		**OE. 3**	**70 horas**	**mar 11/1/22**	**vie 21/1/22**
31		AC. 2 Aplicar normativas de bioseguridad para los repartidores que hagan las entregas a domicilio o envios por parte de los clientes.	35 horas	mar 11/1/22	lun 17/1/22
32		AC. 3 Realizar politicas de entrega registradas en la app en donde el dueño del restaurante o el cliente que realiza el pedido pueda visualizar y calificar el protocolo propuesto.	35 horas	lun 17/1/22	vie 21/1/22
33		**OE . 2**	**50 horas**	lun 24/1/22	**mar 1/2/22**
34		AC. 3 Dar seguimiento en tiempo real a los repartidores por medio de socket los cuales estarán implementados por medio de APIs en Flutter	30 horas	lun 24/1/22	jue 27/1/22
35		AC. 4. Establecer tiempos de espera para los diferentes repartidores y tiempos de entrega para los que realizan los envios.	20 horas	jue 27/1/22	lun 31/1/22



Total, de horas: 600
Horas Juan Fernando Córdova Arévalo 300 H
Horas Telmo Sebastián Rocano Ortega 300 H
Fecha de Inicio: 25/10/2021
Fecha de Finalización: 01/02/2022

PRESUPUESTO

DENOMINACIÓN	CANT.	COSTO UNITARIO	COSTO TOTAL
Unidades	**Dólares**	**Dólares**	**Dólares**
1. Bienes			
Pasajes	6	2.10	12.60
2. Tecnológico			
Laptop (Intel Core i7 8va Gen - 32 GB 2667 MHz DDR4- AMD Radeon Pro 5500M 2 GB Intel UHD Graphics 630 1GB)	2	800	1600
Celulares android	3	350	1050
2. Servicios			
Servicios de Internet (Utilizar Google para la geolocalización)	5	15	90
Servicio de Internet	6	40	240
4. Personal			
Estudiantes/Desarrollador	600 horas (2 Estudiantes)	20 (por hora)	12,000
3. Otros			
Imprevistos	1	30.00	30.00
TOTAL			**15,022.60**

CONCLUSIONES

Este proyecto tuvo como objetivo principal desarrollar una aplicación que permita al usuario realizar compras en línea mientras el negocio al cual fue realizado el pedido asigne un repartidor dándole seguimiento en tiempo real tanto por el usuario que realizó la compra como por el dueño del negocio que envió el producto utilizando herramientas de geolocalización lo que nos permitirá, por una parte trazar la ruta desde la ubicación del repartidor a la ubicación del domicilio en donde se entregará el producto, cabe recalcar que la ruta que se trazara será la de menor distancia, la que esté libre de tráfico y la más segura con ayuda de paquetes de Google Maps enlazadas directamente con el mapa que trae nuestros teléfonos inteligentes, así el usuario que realiza el pedido podrá realizar el seguimiento del motorizado y en caso de que el motorizado no siga la ruta trazada el usuario no perderá de vista al motorizado, los diferentes paquetes que utilizamos en nuestro proyecto nos ha ayudado a que el seguimiento en tiempo real se pueda cumplir como lo determinamos desde un principio, nuestra aplicación está diseñada para ser escalable, al momento de terminar la aplicación hemos cumplido con los objetivos propuestos desde un principio, pero esto no quiere decir que nuestra aplicación está lista para salir al mercado y formar parte de una tienda de aplicaciones, es necesario realizar estudios de ganancias y ventas, realizar un diseño más amigable con el usuario, realizar mejoras de ventas y categorizar negocios que sería de mucha ayuda para poder distribuir la aplicación no solo a un cliente dueño de negocios, sino tener varias alternativas de consumos, en cuanto a almacenamiento y base de datos realizar un estudio más detallado de software que podamos utilizar para que nuestra aplicación se extienda a gran escala, para la presentación se utiliza software libre pero este software se puede mejorar con estudios más avanzados en cuanto a almacenamiento de información que con el avance tecnológico todo pudiera darse directamente en la nube.

Hemos realizado el comienzo de una aplicación que con ayuda de profesionales en varias ramas podemos llegar a obtener a futuro una aplicación estable y sólida para el consumo de miles de clientes, de esta manera poder llegar a competir con grandes aplicaciones que realizan las mismas funciones, pero con la diferencia que esta aplicación tendría una marca ecuatoriana.

RECOMENDACIONES

- Utilizar siempre paquetes actualizados, y verificar que exista compatibilidad con las diferentes versiones que ofrece el software, de esta manera no existirá problemas de compatibilidad.
- Realizar siempre un seguimiento de las nuevas versiones de los paquetes que se actualizan día a día de esta manera la app no tendrá problemas de funcionamiento.
- Tener mucho cuidado con los paquetes obsoletos, existen muchos casos en donde los paquetes que se utilizan para una aplicación quedan obsoletos dañando la imagen de la aplicación.
- Los seguimientos en tiempo real están basados en paquetes propios de flutter por lo que se recomienda realizar un análisis de las nuevas versiones antes de realizar cualquier actualización del software.
- Se recomienda siempre tener cuidado con las duplicaciones en la base de datos relacionado con Firebase, esto puede ocasionar problemas al momento de realizar un llamado desde la aplicación.
- Realizar un correcto análisis en cuanto a las tablas de la base de datos, refiriéndonos principalmente a las relaciones que existen entre cada una de las tablas, de esta manera no existirá problemas al momento de guardar, buscar, modificar o eliminar un elemento.
- Se recomienda investigar a fondo las diferentes formas de pago que pueden ser admitidas en la aplicación dentro de Ecuador como por ejemplo Paymentez que se podría utilizar puesto que tiene convenios con el Banco del Pacifico.

REFERENCIAS BIBLIOGRÁFICAS

Ahmed, Y. (2022, January 20). *Launching URLs in Flutter with url_launcher.* Retrieved March 10, 2022 from LogRocket Blog: https://blog.logrocket.com/launching-urls-flutter-url_launcher/

Asfo Senior Application Developer. (2018, Agosto 29). *Desarrollando una sencilla API REST con NodeJS y Express.* Retrieved January 20, 2022 from Asfo – Medium: https://asfo.medium.com/desarrollando-una-sencilla-api-rest-con-nodejs-y-express-cab0813f7e4b

Ceballos, F. J. (2004). *Lenguajes de Programación.* Retrieved December 21, 2021 from UNAM: https://programas.cuaed.unam.mx/repositorio/moodle/pluginfile.php/1023/mod_resource/content/1/contenido/index.html

Developer, A. (2021). *Xcode 13 Overview.* Retrieved December 20, 2021 from Apple Developer: https://developer.apple.com/xcode/

Enriquez, J. G., & Casas, S. I. (2014). *Usabilidad en aplicaciones móviles. Informes Científicos Técnicos - UNPA, 5(2), 25–47.* From https://doi.org/10.22305/ict-unpa.v5i2.71

Firebase Documentation. (2022, February 01). *Add the Firebase Admin SDK to your server | Firebase Documentation.* Retrieved February 17, 2022 from Firebase: https://firebase.google.com/docs/admin/setup/

Flutter. (2022). *Introducción a los Widgets.* Retrieved January 17, 2022 from Flutter: https://esflutter.dev/docs/development/ui/widgets-intro

Flutter. (2022). *Tutorial de animaciones.* Retrieved January 17, 2022 from Flutter: https://esflutter.dev/docs/development/ui/animations/tutorial

Gayatribkar. (2021, October 16). *Flutter - DropDownButton Widget.* Retrieved March 9, 2022 from GeeksforGeeks: https://www.geeksforgeeks.org/flutter-dropdownbutton-widget/

Google Maps Platform. (2022, Febrary 15). *Maps SDK for Android overview.* Retrieved March 7, 2022 from Google Developers: https://developers.google.com/maps/documentation/android-sdk/overview

Google Maps Platform. (2022, Febrero 15). *Overview | Maps SDK for iOS.* Retrieved March 7, 2022 from Google Developers: https://developers.google.com/maps/documentation/ios-sdk/overview

Google Maps Platform. (2022, Marzo 8). *The Directions API overview.* Retrieved March 10, 2022 from Google Developers: https://developers.google.com/maps/documentation/directions/overview

Hurtado, H. (2020, June 17). *JSON Web tokens (JWT): claves para usarlos de manera segura.* Retrieved February 23, 2022 from BBVA: https://www.bbva.com/es/json-web-tokens-jwt-claves-para-usarlos-de-manera-segura/

Introducción a Android Studio | Desarrolladores de Android. (2021, May 17). Retrieved December 20, 2021 from Android Developers: https://developer.android.com/studio/intro?hl=es-419

López, J. G. (2019, February 22). *Mercado de apps, en crecimiento y con desafíos.* Retrieved January 4, 2022 from Consumotic: https://www.consumotic.mx/tecnologia/mercado-de-apps-en-crecimiento-y-con-desafios/

MDN contributors. (2022, Junuary 3). *Introducción a Express/Node - Aprende sobre desarrollo web | MDN.* Retrieved January 4, 2022 from MDN Web Docs: https://developer.mozilla.org/es/docs/Learn/Server-side/Express_Nodejs/Introduction

Medina, R. (2020, May 1). *Google Search Console: ¿Qué es, para qué sirve y cómo funciona?* Retrieved March 7, 2022 from Branch: https://branch.com.co/marketing-digital/que-es-google-search-console/

Microsoft. (2021, December 17). *Así es como se crea software.* Retrieved December 21, 2021 from Visual Studio: IDE y Editor de código para desarrolladores de software y Teams: https://visualstudio.microsoft.com/es/

Microsoft. (2021, January 29). *CREATE SEQUENCE (Transact-SQL) - SQL Server.* Retrieved January 24, 2022 from Microsoft Docs: https://docs.microsoft.com/en-us/sql/t-sql/statements/create-sequence-transact-sql?view=sql-server-ver15

Moreno, G. (2021, June 29). *Cómo realizar pruebas automatizadas con Postman.* Retrieved December 21, 2021 from Encora: https://www.encora.com/es/blog/como-realizar-pruebas-automatizadas-con-postman

Netec. (2019, April 25). *Qué es Google Cloud Platform, caracteristicas y ventajas.* Retrieved March 7, 2022 from Netec: https://www.netec.com/post/todo-lo-que-necesita-saber-de-google-cloud-platform

Pub dev. (2021, Abril 16). *flutter_polyline_points | Flutter Package.* Retrieved March 10, 2022 from Pub.dev: https://pub.dev/packages/flutter_polyline_points

Pub dev. (2021, Junio 14). *location | Flutter Package.* Retrieved March 8, 2022 from Pub.dev: https://pub.dev/packages/location

Quicktype. (2018, August). *App Quicktype.* Retrieved March 3, 2022 from Convert JSON to Swift, C#, TypeScript, Objective-C, Go, Java, C++ and more • quicktype: https://quicktype.io

Rank MyApp. (2018, November 10). *Tendencias del mercado de aplicaciones.* Retrieved January 4, 2022 from RankMyAPP: https://www.rankmyapp.com/es/mercado-es/tendencias-del-mercado-de-aplicaciones/

Socket.IO. (2022, March 7). *Introduction.* Retrieved March 10, 2022 from Socket.IO: https://socket.io/docs/v4/

Aguilar, L. U. M., López, M. D. L. Á. S., & Peña, J. M. F. Herramienta para Administrar Información Básica de Requerimientos Funcionales y No Funcionales en el Desarrollo de Sistemas Informáticos.