

**UNIVERSIDAD TÉCNICA PARTICULAR DE LOJA**

*La Universidad Católica de Loja*

**FACULTAD DE DE INGENIERÍAS Y ARQUITECTURA**

**CARRERA DE INGENIERÍA EN INFORMÁTICA**

**Implementación de un prototipo de aplicativo móvil para facilitar la comercialización de cilindros de GLP (Gas licuado de petróleo) en la ciudad del Tena**

Trabajo de integración curricular previo a la obtención del título de:

**INGENIERIO EN INFORMÁTICA**

**Autor:** Tanguila Coquinche, Juan Carlos

**Director:** Elizalde, Rene Rolando

CENTRO UNIVERSITARIO TENA

2022

# Aprobación del director del Trabajo de Integración Curricular

Loja, 24 de Julio del 2022

Ingeniero

Daniel Alejandro Guamán

**Director de la carrera de Informática**

Quito. -

De mi consideración:

Me permito comunicar que, en calidad de director del presente Trabajo de Integración Curricular denominado: Implementación de un prototipo de aplicativo móvil con administrador web para facilitar la comercialización de cilindros de GLP (Gas licuado de petróleo) en la ciudad del Tena realizado por José Javier Llori Licuy y Juan Carlos Tanquila Conquinche ha sido orientado y revisado durante su ejecución, así mismo ha sido verificado a través de la herramienta de similitud académica institucional, y cuenta con un porcentaje de coincidencia aceptable. En virtud de ello, y por considerar que el mismo cumple con todos los parámetros establecidos por la Universidad, doy mi aprobación a fin de continuar con el proceso académico correspondiente.

Particular que comunico para los fines pertinentes.

Atentamente,

Director: Elizalde, Rene Rolando.

C.I.:

Correo electrónico: rrelizalde@utpl.edu.ec

# Declaración de **autoría** y cesión de derechos

Yo, Juan Carlos Tanquila Conquinche, declaran y aceptan en forma expresa lo siguiente:

Ser autor del Trabajo de Integración Curricular denominado: Implementación de un prototipo de aplicativo móvil con administrador web para facilitar la comercialización de cilindros de GLP (Gas licuado de petróleo) en la ciudad del Tena, de la carrera de Ingeniería Informática, específicamente de los contenidos comprendidos en: Capitulo I. Problemática, Objetivos y Justificación, Capitulo II. Antecedentes y marco teórico, Capitulo III. Arquitectura Hardware y Software del sistema y Capitulo IV. Diseño e implementación del prototipo de aplicativo móvil, siendo (Rene Rolando Elizalde), director del presente trabajo; también declaro que la presente investigación no vulnera derechos de terceros ni utiliza fraudulentamente obras preexistentes. Además, ratifico que las ideas, criterios, opiniones, procedimientos y resultados vertidos en el presente trabajo investigativo, son de mi exclusiva responsabilidad. Eximo expresamente a la Universidad Técnica Particular de Loja y a sus representantes legales de posibles reclamos o acciones judiciales o administrativas, en relación a la propiedad intelectual de este trabajo.

Que la presente obra, producto de mis actividades académicas y de investigación, forma parte del patrimonio de la Universidad Técnica Particular de Loja, de conformidad con el artículo 20, literal j), de la Ley Orgánica de Educación Superior; y, artículo 91 del Estatuto Orgánico de la UTPL, que establece: “Forman parte del patrimonio de la Universidad la propiedad intelectual de investigaciones, trabajos científicos o técnicos y tesis de grado que se realicen a través, o con el apoyo financiero, académico o institucional (operativo) de la Universidad”, en tal virtud, cedo a favor de la Universidad Técnica Particular de Loja la titularidad de los derechos patrimoniales que me corresponden en calidad de autor/a, de forma incondicional, completa, exclusiva y por todo el tiempo de su vigencia.

La Universidad Técnica Particular de Loja queda facultada para ingresar el presente trabajo al Sistema Nacional de Información de la Educación Superior del Ecuador para su difusión pública, en cumplimiento del artículo 144 de la Ley Orgánica de Educación Superior.

.................................................................

Autor: Juan Carlos Tanguila Conquinche

C.I.: 1500864010

Correo electrónico: jctanguila@utpl.edu.ec

# 

# Dedicatoria

El presente trabajo de grado va dedicado a Dios, quien como guía estuvo presente en el caminar de mi vida, bendiciéndome y dándome fuerzas para continuar con mis metas trazadas sin desfallecer. A mis padres que, con apoyo incondicional, amor y confiaron en mí y ahora puedo dedicarles el último esfuerzo de mi carrera académica. Finalmente, dedico este trabajo a todos mis amigos y amigas que me brindaron su apoyo y a quienes agradezco por medio de esta dedicatoria el creer en mí.

# Agradecimiento

El presente trabajo agradezco a Dios por ser mi guía y acompañarme en el transcurso de mi vida, brindándome paciencia y sabiduría para culminar con éxito mis metas propuestas.

A mis padres por ser mi pilar fundamental y haberme apoyado incondicionalmente, pese a las adversidades e inconvenientes que se presentaron.

Agradezco a mi director de tesis quien, con su experiencia, conocimiento y motivación me oriento en la investigación. A mis compañeros por sus consejos, enseñanzas, apoyo y sobre todo la amistad brindada en los momentos más difíciles de mi vida.

Agradezco a los todos docentes que, con su sabiduría, conocimiento y apoyo, motivaron a desarrollarme como persona y profesional en la Universidad Técnica Particular de Loja.

# Índice de contenido

[Aprobación del director del Trabajo de Integración Curricular II](#_Toc139633711)

[Declaración de autoría y cesión de derechos III](#_Toc139633712)

[Dedicatoria V](#_Toc139633713)

[Agradecimiento VI](#_Toc139633714)

[Índice de contenido VII](#_Toc139633715)

[Índice de tablas VIII](#_Toc139633716)

[Índice de figuras IX](#_Toc139633717)

[Resumen 1](#_Toc139633718)

[Abstract 2](#_Toc139633719)

[Capitulo uno 3](#_Toc139633720)

[Problemática, objetivos y justificación 3](#_Toc139633721)

[1.1 Problemática actual 3](#_Toc139633722)

[1.2 Objetivos 3](#_Toc139633723)

[1.3 Justificación 4](#_Toc139633724)

[Capítulo dos 5](#_Toc139633725)

[Antecedentes y marco teórico 5](#_Toc139633726)

[2.1 Generalidades del capitulo 5](#_Toc139633727)

[2.2 Fundamentación teórica 5](#_Toc139633728)

[2.2.1 Sistemas informáticos 5](#_Toc139633729)

[2.2.2 Lenguajes de programación web 6](#_Toc139633730)

[2.2.3 Node.js 11](#_Toc139633731)

[2.2.4 Express 12](#_Toc139633732)

[2.2.5 BackEnd 12](#_Toc139633733)

[2.2.6 FrontEnd 13](#_Toc139633734)

[2.2.7 CSS 14](#_Toc139633735)

[2.2.8 Arquitectura de software 15](#_Toc139633736)

[2.2.9 Metodología de desarrollo de software 16](#_Toc139633737)

[2.2.10 Aplicaciones móviles 20](#_Toc139633738)

[2.2.11 Programación para Android 21](#_Toc139633739)

[2.2.12 Bases de datos 24](#_Toc139633740)

[2.3 Antecedentes de investigación 30](#_Toc139633741)

[Capitulo tres 34](#_Toc139633742)

[Arquitectura hardware y software del sistema 34](#_Toc139633743)

[3.1 Generalidades del capitulo 34](#_Toc139633744)

[3.2 Arquitectura hardware 34](#_Toc139633745)

[3.2.1 Servidor 35](#_Toc139633746)

[3.2.2 Servidor web 35](#_Toc139633747)

[3.3 Arquitectura de software 37](#_Toc139633748)

[3.3.1 Librería HTTP 38](#_Toc139633749)

[3.3.2 API de Google 39](#_Toc139633750)

[3.3 MVC de la aplicación móvil 43](#_Toc139633751)

[Capítulo cuatro 44](#_Toc139633752)

[Diseño e implementación del prototipo de aplicativo móvil 44](#_Toc139633753)

[4.1. Requerimientos funcionales 44](#_Toc139633754)

[4.1.1. Requerimientos no funcionales 45](#_Toc139633755)

[4.2. Producto backlog 45](#_Toc139633756)

[4.2.1. Especificación de roles 51](#_Toc139633757)

[4.3. Sprint 0: Fase previa 51](#_Toc139633758)

[4.3.1. Declaración del producto 51](#_Toc139633759)

[4.3.2. Definición del terminado 51](#_Toc139633760)

[4.3.3. Diseño 52](#_Toc139633761)

[4.3.4. Prototipado 67](#_Toc139633762)

[4.4. Sprint 1: Módulo de seguridad, login, registro y recuperación de contraseña 70](#_Toc139633763)

[4.5. Sprint 2: Módulo de Usuarios 75](#_Toc139633764)

[4.7. Sprint 3: Módulo de productos 78](#_Toc139633765)

[4.8. Sprint 4: Módulo de pedidos, generar un nuevo pedido 81](#_Toc139633766)

[4.9. Sprint 5: Módulo de pedidos, lista de pedidos y asignación de repartidor 85](#_Toc139633767)

[4.10. Sprint 6: Módulo de pedidos, traza de rutas, notificaciones y fin de pedidos 89](#_Toc139633768)

[Bibliografía 93](#_Toc139633769)

# Índice de tablas

Tabla 1 32

Tabla 2 33

Tabla 3 46

Tabla 4 51

Tabla 5 52

Tabla 6 53

Tabla 7 57

Tabla 8 58

Tabla 9 59

Tabla 10 60

Tabla 11 61

Tabla 12 62

Tabla 13 63

Tabla 14 64

Tabla 15 65

Tabla 16 66

Tabla 17 70

Tabla 18 71

Tabla 19 75

Tabla 20 75

Tabla 21 78

Tabla 22 78

Tabla 23 81

Tabla 24 82

Tabla 25 85

Tabla 26 85

Tabla 27 89

Tabla 28 90

Tabla 29 92

# Índice de figuras

Figura 1 6

Figura 2 7

Figura 3 8

Figura 4 9

Figura 5 10

Figura 6 11

Figura 7 14

Figura 8 15

Figura 9 16

Figura 10 17

Figura 11 18

Figura 12 20

Figura 13 22

Figura 14 23

Figura 15 27

Figura 16 28

Figura 17 34

Figura 18 36

Figura 19 37

Figura 20 38

Figura 21 40

Figura 22 41

Figura 23 42

Figura 24 43

Figura 25 55

Figura 26 56

Figura 27 57

Figura 28 58

Figura 29 60

Figura 30 61

Figura 31 63

Figura 32 64

Figura 33 65

Figura 34 66

Figura 35 67

Figura 36 67

Figura 37 68

Figura 38 68

Figura 39 69

Figura 40 69

Figura 41 70

Figura 42 71

Figura 43 74

Figura 44 76

Figura 45 77

Figura 46 78

Figura 47 82

Figura 48 84

Figura 49 86

Figura 50 89

Figura 51 90

Figura 52 92

# Resumen

El objetivo del presente proyecto es el desarrollo de un prototipo de sistema informático que gestione la comercialización de cilindros de GLP (Gas licuado de petróleo) en la ciudad del Tena, por medio de una aplicación móvil para realizar los pedidos de los cilindros de gas. Para lograr la propuesta se procedió al análisis de requerimientos funcionales necesarios para obtener un correcto funcionamiento del prototipo. Posterior a ello se elaboró la tabla de sprint de desarrollo de todo el proyecto, lo cual fue la base para el diseño para la creación de la base de datos en MySQL, el diagrama de clases del sistema, los casos de uso necesarios para cada una de las actividades realizadas en la propuesta y finalmente, los prototipos de interfaz de usuarios o Mockups.

Conforme a los diseños elaborados se realizó el desarrollo de la aplicación móvil en Android Studio lenguaje JAVA, el cual permitió que los usuarios con rol de Cliente y Repartidor puedan acceder a sus funciones por medio del ingreso de credenciales de seguridad. En cuanto al usuario Administrador se implementó una interfaz en la aplicación móvil para la gestión de usuarios así mismo que le brinda al cliente la posibilidad de generar pedidos mismos que serán despachados por los repartidores.

*Palabras clave****:*** Prototipo, aplicación, requerimientos, JAVA, Android y MySQL.

# Abstract

The objective of this project is the development of a prototype of a computer system that manages the commercialization of LPG cylinders (Liquefied Petroleum Gas) in the city of Tena, through a mobile application to place orders for gas cylinders. To achieve the proposal, we proceeded to analyze the functional requirements necessary to obtain a correct functioning of the prototype. After that, the development sprint table for the entire project was prepared, which was the basis for the design for the creation of the MySQL database, the class diagram of the system, the use cases necessary for each of the activities carried out in the proposal and finally, the user interface prototypes or Mockups.

According to those elaborated, the development of the mobile application was carried out in Android Studio JAVA language, which demonstrated that users with the role of Client and Distributor can access their functions by entering security credentials. As for the Administrator user, an interface was implemented in the mobile application for user management, which also gives the client the possibility of generating orders that will be dispatched by the distributors.

*Keywords*: Prototype, application, requirements, JAVA, Android and MySQL

# Capitulo uno

# Problemática, objetivos y justificación

## 1.1 Problemática actual

En la actualidad la demanda de cilindros de gas es una problemática de todos los días, ya que en algunos sectores los repartidores de este servicio no acuden de forma rápida o simplemente su ruta no completa la totalidad de calles, por lo que las personas prefieren dirigirse ellos mismo a los depósitos para poder adquirir un cilindro de gas. Esto sin duda causa molestia e inconformidad en la población, ya que para adquirir un cilindro de gas se debe esperar un tiempo demasiado largo. Para ello es necesaria una solución que permita a los ciudadanos solicitar un cilindro de gas y que los depósitos de gas envíen un repartidor a suplir esta necesidad.

Para lo cual se propone la implementación de un sistema informático que permita a los usuarios solicitar un cilindro de gas por medio de una aplicación móvil, la misma que envía esta solicitud al administrador del depósito de gas. Después por medio de la aplicación web se podrá asignar el pedido al repartidor más cercano el mismo que recibirá una notificación por medio de la aplicación móvil y que a su vez procederá a llevar el cilindro de gas al usuario a la dirección desde donde solicito el servicio. De esta forma se brindará un servicio rápido, efectivo y asegurando que los repartidores realicen entregas fijas.

## 1.2 Objetivos

*Objetivo general*

Desarrollar un prototipo de sistema informático que gestione la comercialización de cilindros de GLP (Gas licuado de petróleo) en la ciudad del Tena, por medio de una aplicación móvil para realizar los pedidos de los cilindros de gas.

*Objetivos específicos*

1. Elaborar un marco teórico que permita determinar las herramientas más adecuadas para la generación de una aplicación móvil.
2. Diseñar el prototipo conforme a los requerimientos funcionales que deben cumplir la aplicación móvil. Además de construir una base de datos que permita gestionar la información de forma segura y confiable.
3. Construir la aplicación móvil con todos los componentes técnicos necesarios para mejorar el proceso de compra y venta de cilindros de GLP.
4. Realizar las pruebas necesarias al sistema a fin de evaluar su rendimiento, verificar su funcionamiento y corregir los posibles errores a fin de tener un prototipo funcional.

## 1.3 Justificación

La propuesta presentada que consiste en la implementación de un prototipo de sistema informático para la venta de cilindros de gas está dirigida a las empresas que se encargan de distribuirlos, los mismos que tendrán acceso a una herramienta informática que busca innovar la forma actual de comercialización de este producto. Por un lado, los clientes podrán solicitar un cilindro de gas desde una aplicación, mientras que las empresas que se encargan de su distribución podrán atender a este pedido y enviarlo de forma inmediata brindando un servicio rápido y eficiente a la comunidad.

La finalidad de la presente propuesta es mejorar el servicio actual de distribución de gas en la ciudad del Tena con el fin de centralizar así los clientes y que estos sean atendidos de forma rápida, disminuyendo los tiempos de espera y brindando un servicio más eficiente.

# Capítulo dos

# Antecedentes y marco teórico

## 2.1 Generalidades del capitulo

En el presente capítulo se trata de analizar antecedentes de propuestas similares que ayuden a identificar funcionalidades que pueden ser parte de la propuesta actual. Además, se busca extraer los conceptos más importantes que fundamentan el proyecto para el desarrollo del prototipo de aplicación móvil.

## 2.2 Fundamentación teórica

### 2.2.1 Sistemas informáticos

En la actualidad se ha convertido en una especia que prolonga varios factores de computación, programación, software y electrónica, la omnipresencia de la informática en la sociedad es paradójicamente, la invención del ordenador representa uno de los hilos fundamentales de la humanidad quizás tan importante como la aparición de la rueda o la escritura, en el campo de la informática muestra que un ordenador es más puro su esencia. (Chavez, 2020)

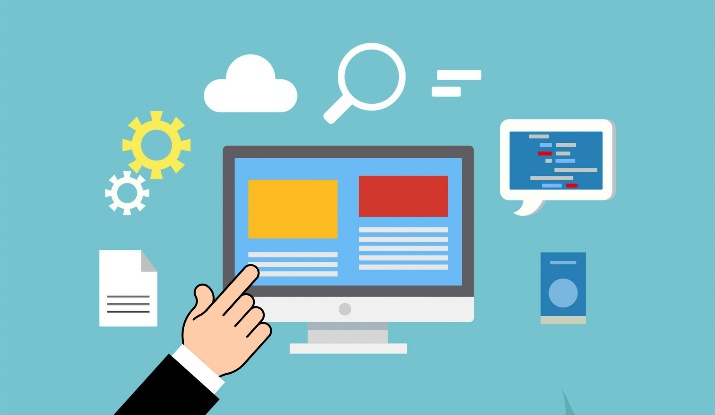
***Ventajas y beneficios de los sistemas web***

Según (San Juan, 2016), una aplicación web es una aplicación de software que se puede utilizar accediendo a un servidor web a través de Internet o de una intranet mediante un navegador. Las aplicaciones web son cada vez más populares y cada vez las usa más gente.

* *Ahorro de dinero:* Los sistemas web permiten aumentar la eficiencia ya que se utilizan menos recursos de software ya que no requiere la instalación de programas extra. Otra ventaja es que se recurre únicamente en el gasto del servidor, mas no en ningún otro equipo.
* *Facilidad de usabilidad:* El manejo de estos es muy sencillo ya que requiere de una experiencia mínima en ofimática.
* *Facilitan el trabajo colaborativo y a distancia:* Se puede usar el sistema por varios usuarios desde puntos distintos, equipos distintos y de forma simultánea. Todo esto gracias a que los datos se encuentran centralizados en un solo sistema. Otra ventaja importante es que los usuarios pueden trabajar encima de un mismo documento al mismo documento lo cual aumenta el trabajo colaborativo en las empresas.
* *Son escalables y actualizables:* El actualizar o aumentar módulos al sistema es más fácil ya que no requiere que el usuario instale o descarguen instaladores ni ningún otro software, sino simplemente se suben las actualizaciones al servidor y están disponibles a la brevedad.
* *Provocan menos errores y problemas:* Los sistemas o aplicaciones web siempre están activas y es difícil que no se encuentren disponibles, ya que los servidores en donde se encuentran alojados difícilmente presentan errores.
* *Los datos son más seguros:* No hay que preocuparse de posibles rupturas del disco duro ni de los virus que pueden hacer perder toda la información.

Figura 1

*Sistemas informáticos en la sociedad*



*Fuente:* https://bit.ly/3STOIgw

Dentro del desarrollo web existen ciertos lenguajes de programación que son más utilizados actualmente tales como: JavaScript, Python, Java y PHP, mismos que permiten crear sistemas web adaptados a las necesidades de una empresa o negocio. Para ello se dará una pequeña introducción de cada uno de ellos en función de determinar el más adecuado para la propuesta.

### 2.2.2 Lenguajes de programación web

**JavaScript**

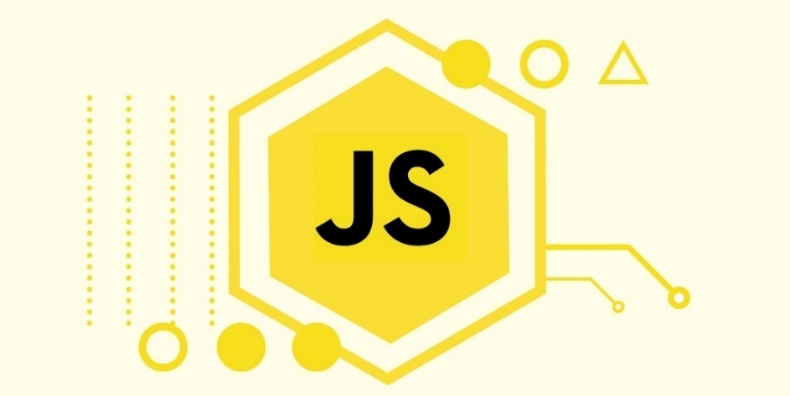
JavaScript, originariamente llamado LiveScript, es un lenguaje de programación creado por Netscape y nació en 1995 con un propósito muy claro: Mejorar la navegación del usuario directamente desde el navegador. Cuando se dice que un lenguaje es del lado del cliente, nos referimos a que se ejecuta en la máquina del propio cliente a través de un navegador. Algunos de estos lenguajes son el propio JavaScript, HTML, CSS o Java. (Mdn Web Docs, 2022)

Según (Miteris, 2018), La categoría de lenguajes se diferencia de la otra gran categoría: los lenguajes del lado del servidor. Estos lenguajes se ejecutan e interpretan por el propio servidor y necesitan un tratamiento antes de mostrarlos al usuario final. Algunos de los lenguajes de programación del lado del servidor más conocidos son PHP, ASP o PERL. JavaScript por sí solo es bastante compacto, aunque muy flexible, y los desarrolladores han escrito gran cantidad de herramientas encima del núcleo del lenguaje JavaScript, desbloqueando una gran cantidad de funcionalidad adicional con un mínimo esfuerzo. Esto incluye:

* Interfaces de Programación de Aplicaciones del Navegador (API’s) — API’s construidas dentro de los navegadores que ofrecen funcionalidades como crear dinámicamente contenido HTML y establecer estilos CSS, hasta capturar y manipular un vídeo desde la cámara web del usuario, o generar gráficos 3D y muestras de sonido.
* API’s de terceros, que permiten a los desarrolladores incorporar funcionalidades en sus sitios de otros proveedores de contenidos como Twitter o Facebook.
* Marcos de trabajo y librerías de terceros que puedes aplicar a tu HTML para que se pueda construir y publicar rápidamente sitios y aplicaciones.

Figura 2

*JavaScript como lenguaje de programación actual*



*Fuente:* https://bit.ly/3QLGfdj

**Python**

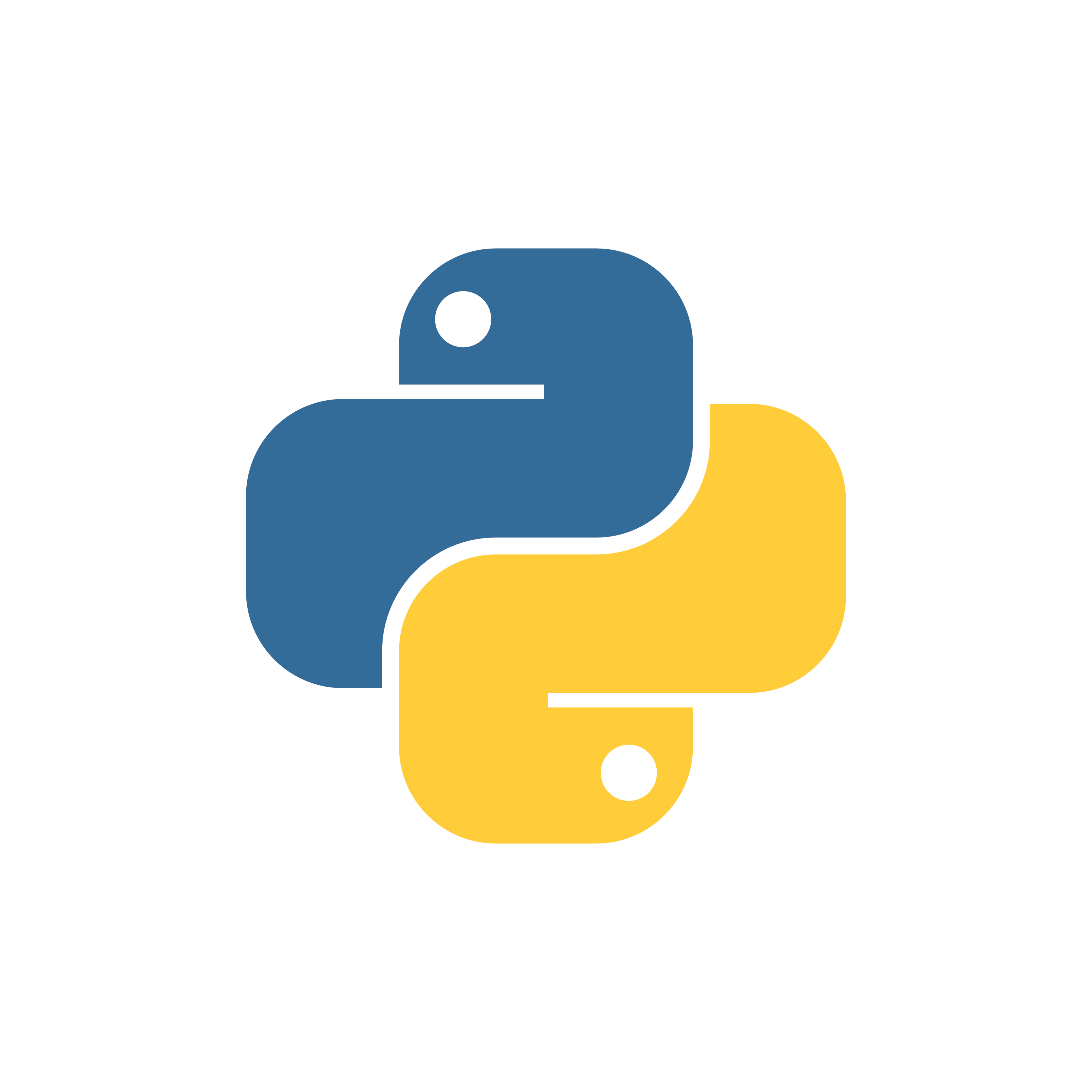
Según (García Monsálvez, 2017), Python es un lenguaje de programación de alto nivel, además de ser orientado a objetos tiene la ventaja de tener una semántica dinámica, lo que facilita el desarrollo web o de sistemas informáticos. En el campo del Desarrollo Rápido de Aplicaciones (RAD) resulta atractivas las características de tipificación y encuadernado dinámico. Python es realmente un lenguaje de propósito general, por lo que se ha vuelto común en el desarrollo de aplicaciones web, ciencia de datos, computación científica, inteligencia artificial e internet de las cosas.

Según (Angel, 2019), las principales características de Python son:

1. *Simplificado rápido:* Este lenguaje simplifica mucho la programación, es un gran lenguaje para scripting.
2. *Elegante y flexible:* El lenguaje ofrece muchas facilidades al programador al ser fácilmente legible e interpretable.
3. *Programación sana y productiva:* Es sencillo de aprender, con una curva de aprendizaje moderada. Es muy fácil comenzar a programar y fomenta la productividad.
4. *Ordenado y limpio:* Es muy legible y sus módulos están bien organizados.
5. *Portable:* Es un lenguaje muy portable. Podemos usarlo en prácticamente cualquier sistema de la actualidad.

Figura 3

*Python como lenguaje de programación web*



*Fuente:* https://bit.ly/3pFJLum

**Java**

Según (Digital Guide, 2019), Java es un lenguaje de programación open source y multiplataforma que, gracias a su versatilidad, es adecuado para, prácticamente, cualquier proyecto. Como la mayoría de los lenguajes web más conocidos, está orientado a objetos, es decir, depende de su campo de aplicación concreto. Java permite diseñar softwares que podrán ser ejecutados y distribuidos en las diferentes plataformas (MAC, Linux, Windows, etc.), sin la necesidad de modificarlos e incluso sin pensar en la arquitectura de la máquina. Gracias al Java Virtual Machine, una máquina virtual que crea un puente entre la aplicación y el hardware del dispositivo.

Las principales características de Java son:

1. *Simple:* Java ofrece la funcionalidad de un lenguaje potente, derivado de C y C++, pero sin las características menos usadas y más confusas de estos, haciéndolo más sencillo.
2. *Orientado a objetos:* El enfoque orientado a objetos (OO) es uno de los estilos de programación más populares. Permite diseñar el software de forma que los distintos tipos de datos que se usen estén unidos a sus operaciones.
3. *Distribuido:* Java proporciona una gran biblioteca estándar y herramientas para que los programas puedan ser distribuidos.
4. *Seguro y sólido:* Proporcionando una plataforma segura para desarrollar y ejecutar aplicaciones que, administra automáticamente la memoria, provee canales de comunicación segura protegiendo la privacidad de los datos y, al tener una sintaxis rigurosa evita que se quiebre el código, es decir, no permite la corrupción del mismo.

Figura 4

*Java como lenguaje de programación web*



Fuente: https://bit.ly/3Cw6BMH

**PHP**

PHP es un lenguaje de código abierto usado principalmente para desarrollar sistemas web y que permite ser insertado en HTML. Las páginas que son desarrolladas en PHP contienen etiquetas normales y código propio del lenguaje de programación, lo que permite interactuar entre la parte gráfica y la lógica de programación. Cuando llega la solicitud de la página web, el servidor preprocesa los datos y ejecuta las instrucciones PHP. Una vez resuelto todo el proceso el servidor envía al cliente una página solo con etiquetas HTML. (Henriquez, 2018)

Este lenguaje de programación ha crecido por lo que actualmente tiene una interfaz de línea de comandos que se utiliza en independientes aplicaciones gráficas. PHP puede ser empleado en casi todos los servidores web y también en casi todas las plataformas y sistemas operativos de manera gratuita.

Según (INFO-IUTEPI, 2017), las principales características de PHP son:

1. Es un lenguaje sencillo lo que facilita su estudio y aprendizaje.
2. Es un lenguaje muy rápido.
3. Puede mezclarse con HTML con facilidad.
4. Tiene características de orientación a objetos como el uso de clases y herencias.
5. Puede manejar ficheros y conectarse a distintas bases de datos (MySQL, Oracle, SQL Server, Informix, PostgreSQL, etcétera).
6. El software para que funcione PHP en los servidores es libre y gratuito.
7. No requiere definición de tipos de variables, aunque sus variables se pueden evaluar también por el tipo que estén manejando en tiempo de ejecución.

Figura 5

*PHP como lenguaje de programación web*



Fuente: https://bit.ly/2OLxMrz

Una vez que se analizaron las características de cada lenguaje de programación se eligió JavaScript como lenguaje para toda la parte web de la propuesta gracias a que tiene las siguientes ventajas según (Gutierrez, 2009):

* Los Script permite acceder a otras funcionalidades o servicios por medio de la página HTML, lo que produce páginas web más profesionales.
* Por medio de AJAX se pueden realizar solicitudes en segundo plano, lo que disminuye los tiempos de espera al enviar y recibir respuestas del servidor.
* JavaScript tiene la capacidad de funcionar en todos los navegadores modernos, por lo que su resultado no varía.
* Tiene compatibilidad con una gran variedad de lenguajes de programación, por lo que puede ser utilizado para aplicaciones web, móviles o de escritorio.
* Existe una gran cantidad de documentación en la web en general, por lo que se facilita su aprendizaje y aplicación de todas las funcionalidades que tiene este lenguaje.
* Permite la creación de interfaces graficas profesionales tanto en componentes gráficos, como en comportamientos, eventos y animaciones.
* Hay algunas formas de usar JavaScript a través de servidores Node.js. Es posible desarrollar una aplicación JavaScript completa de principio a fin utilizando solo JavaScript.

Para programar con JavaScript en la actualidad es posible hacerlo por medio de Node.js, ya que por medio de este se puede ejecutar código no solo en la web sino para la elaboración de un BackEnd que sirva como capa de acceso a los datos del sistema para lo cual es importante mencionar los conceptos necesarios para una mejor comprensión.

### 2.2.3 Node.js

Según (Lucas, 2019) nos indica que fue creado por los desarrolladores originales de JavaScript, transformando algo que solo se podía ejecutar en el navegador a algo que se puede ejecutar en los ordenadores como aplicaciones independientes, siendo un entorno de tiempo de ejecución de JavaScript, es decir, ejecución en tiempo real, aportando a beneficios y soluciones a problemas, se tiene la capacidad de hacer cosas que otros lenguajes de secuencia de comandos como en Python se puede crear. Node siendo lanzado inicialmente, solo para Linux, en 2009. El gestor de paquetes NPM fue lanzado en 2010, y el soporte nativo para Windows fue agregado en 2012, hasta la actualidad la versión LTS (Long Term Support) es Node v12.18.0 mientras que la última versión es Nodo 14.4.0.

*Ventajas de Node.js*

Según (Arias Melo, 2017), usar Node.js tiene las siguientes características:

* Una de las características más importantes de Node.js es la programación asincrónica, la cual permite que las aplicaciones no bloqueen las líneas de código, de modo que los ciclos de procesamiento siempre estén disponibles.
* Cuando se realizan cantidades grandes de iteraciones el lenguaje responde de forma mucho más favorable que por ejemplo PHP lo que indica que será un lenguaje muy utilizado para el desarrollo web.
* Mediante paquetes de node.js se puede enviar archivos de gran peso, lo cual es importante cuando de sistemas de información se trata.
* Con el pasar de los años se ha formado una gran comunidad que día a día genera documentación y demás paquetes que contribuyen al crecimiento de node.js. El crecimiento es exponencial.

Figura 6

*Node.js como entorno de desarrollo de JavaScript*



*Fuente:* https://bit.ly/3SMXgWx

### 2.2.4 Express

Según (Kinsta, 2022), Express es un marco de desarrollo para Node.js que permite La estructuración de una aplicación de una forma rápida, además de proporcionar funcionalidades como en enrutamiento, sesiones, cookies, entre otras. Es el framework más utilizado ya que cuenta con las siguientes características:

* Proporciona un conjunto de herramientas para aplicativos webs, solicitudes y respuestas HTTP.
* Proporciona herramientas de interfaz de línea de comandos CLI que se llama NPM por medio del cual se puede acceder a paquetes desarrollados por una comunidad grande de desarrolladores.
* Permite la integración con motores de renderización de "vistas" para generar respuestas mediante la introducción de datos en plantillas.
* Establecer ajustes de aplicaciones web como qué puerto usar para conectar, y la localización de las plantillas que se utilizan para renderizar la respuesta.
* Añadir procesamiento de peticiones "middleware" adicional en cualquier punto dentro de la tubería de manejo de la petición.

Además, express es muy utilizado para la creación de SPA (Single-Page Application) o aplicaciones de una sola página, de esta forma se puede tener una aplicación web en un solo índice de enrutamiento. El objetivo de este tipo de aplicaciones es el de tener una experiencia fluida y escalable.

Hay librerías para trabajar con cookies, sesiones, inicios de sesión de usuario, parámetros URL, datos POST y cabeceras de seguridad mismo que es muy necesario para la creación de un BackEnd robusto que permita que los datos tengan integridad, se encuentren seguros y el acceso a ellos sea controlado por medio de las sesiones.

### 2.2.5 BackEnd

Según (Maldeadora, 2018), en desarrollo de software el termino BackEnd ha ido tomando mucha relevancia ya que al momento de desarrollar una aplicación web, local o móvil es importante separar la parte lógica de la parte gráfica, esta necesidad ha llevado a los programadores a llamar BackEnd a la capa de acceso a los datos a lo cual los usuarios no tienen acceso y en donde se desarrolla la lógica de un negocio, mientras que el FrontEnd es la parte con la que interactúa el usuario para poder crear, editar, ver y eliminar los datos que se manejan en los procesos de una aplicación.

Según (Bautista García, 2021), el BackEnd contiene toda la lógica de programación, permitiendo así que cada uno de los módulos o funciones del sistema funcionen correctamente. También es la parte en donde se realiza el manejo de la información es decir la conexión a las bases de datos, la creación de modelos, rutas y controladores que darán funcionamiento a todo el sistema, es una parte invisible pero fundamental en los diferentes proyectos web que realice un programador.

Dentro de las principales funcionalidades de un BackEnd están las siguientes:

* Acceder a la información, a través de una aplicación o FrontEnd, es decir el BackEnd pueda recibir solicitudes buscar la información y devolverla como respuesta al cliente.
* Combinar o transformar la información encontrada para que sea vista, leída o entendida de mejor manera por la aplicación que la solicitó.
* Permite la autentificación de usuarios, ya que en la actualidad la seguridad en las aplicaciones o sistemas cumple un rol muy importante.
* Permite que la aplicación sea escalable y pueda añadir comportamientos, funciones o servicios en el futuro.

Desde cualquier punto de vista un BackEnd brinda un mejor manejo de la información en un sistema informático y también permite el acceso al mismo desde cualquier otra aplicación que requiera de la información siempre y cuando cumpla con las seguridades implementadas en el BackEnd.

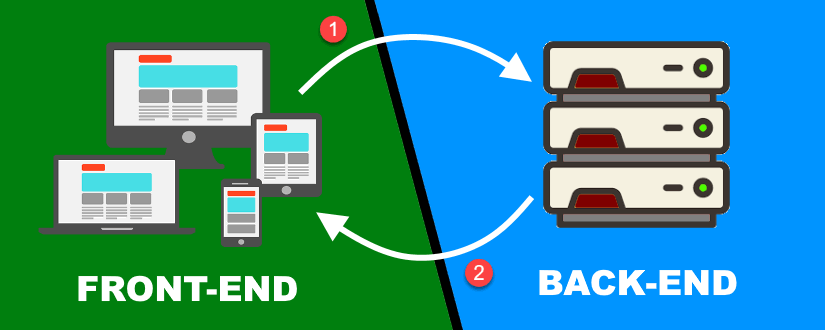
### 2.2.6 FrontEnd

El FrontEnd trabaja en la interfaz visual de un proyecto, por medio de este se le permite al usuario interactuar con el sistema. Es la parte que se encarga de estilizar de forma que la información que se muestra al usuario final sea agradable. El programador que desarrollara el FrontEnd debe tener conocimientos de diseño de Interacción para colocar los elementos en el sistema de forma que para el usuario final sea intuitiva la usabilidad del sistema. (Pérez Ibarra, Quispe, Mullicundo, & Lamas, 2021)

Aunque el FrontEnd es principalmente el diseño de una aplicación, no quiere decir que no exista código, al contrario, en el FrontEnd se debe realizar las solicitudes necesarias al BackEnd, recibir la información y mostrarla al usuario por lo que es muy importante que se tenga en cuenta el manejo de HTML, CSS, JavaScript entre otras librerías que permiten la implementación de una interfaz gráfica funcional.

Figura 7

*BackEnd y FrontEnd en el desarrollo de sistemas*



*Fuente:* https://bit.ly/3dC4Mn2

Para un mejor diseño del FrontEnd es necesario el uso de Bootstrap ya que proporciona un CSS con variedad de estilos que logran formar una interfaz gráfica muy agradable para los usuarios y totalmente estética.

### 2.2.7 CSS

Cascading Style Sheets o Hojas de estilo en cascada es un lenguaje que permite colocar estilos en los elementos de un lenguaje de marcado como HTML, es decir trabaja en conjunto con HTML. En palabras más simples es un lenguaje que se encarga del manejo del diseño y presentación de las páginas web. Cuando se utiliza CSS, el tamaño y posicionamiento de los elementos que conforman las webs será exacto. (Bayas, 2022)

CSS permite tener múltiples estilos en una página HTML, y esto hace que las posibilidades de personalización sean casi infinitas. Hoy en día, esto se está volviendo una necesidad más que algo básico. Para lo cual en el proyecto se hará uso del framework Bootstrap por medio del cual se podrá aplicar estilos en la parte del FrontEnd.

**Bootstrap**

Bootstrap es un framework CSS es decir de estilos el cual permite desarrollar páginas web de una manera más fácil y sencilla. Por medio de Bootstrap se puede modificar los estilos de tipografías, formularios, botones, tablas, navegaciones, menús desplegables, etc. Una de las características principales de Bootstrap es permitir que la adaptación de la página se realice según el tipo de dispositivo utilizado, es decir para garantizar la responsividad. (Deyimar, 2022)

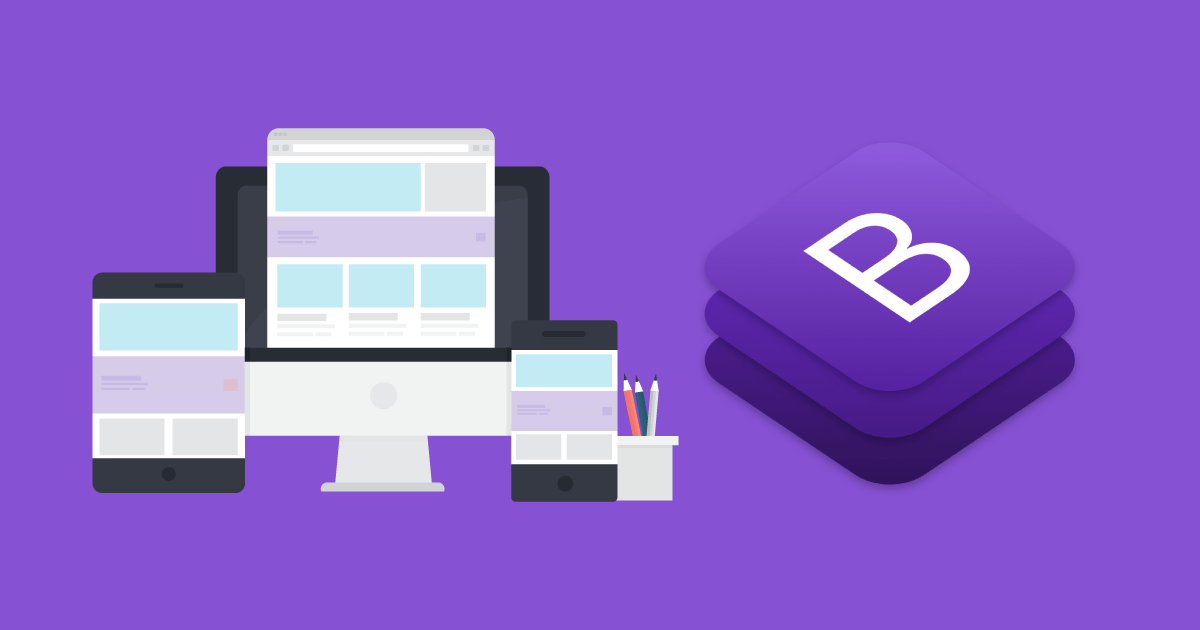
*Ventajas de usar Bootstrap*

Según (Sarzosa Bombón, 2018) el uso de Bootstrap trae las siguientes ventajas:

* Se puede acceder a través de GitHub a toda la documentación y repositorios CSS.
* El sistema de parrilla es de 12 columnas.
* Compatible con Chrome, Firefox, Safari, IE 7 y Opera con tabletas y teléfonos inteligentes.
* Usa HTML5 y CSS3
* Integra algunas librerías de JavaScript y Less 3.
* Es de código libre u open source por lo que puede ser modificable dependiendo la necesidad o gusto del programador.
* El diseño y el estilo que se le puede dar a una página web dando como resultado una interfaz elegante e interactiva con el usuario.

Figura 8

*Bootstrap para todos los dispositivos*



*Fuente:* https://bit.ly/3piR3Uz

### 2.2.8 Arquitectura de software

Según (Martinez, 2021), la arquitectura de software son patrones y lineamientos con los que la aplicación o sistema será desarrollado, además que define la manera en la que se irán creando los módulos, dejando intuir el tipo de aplicación que el desarrollador pretende. Para definir la estructura del sistema se debe tomar en cuenta 4 etapas fundamentales en base a la arquitectura de software y son:

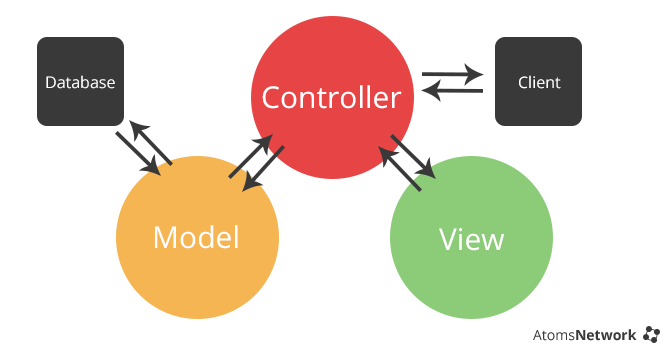
* **Requerimientos:** En esta etapa se recolectan todos los requerimientos y funcionalidades que tendrá el sistema.
* **Diseño:** Es la etapa más importante en donde se define las tecnologías a usarse, así como también los patrones de diseño como por ejemplo MVC.
* **Documentación:** Es importante crear documentación que sirva como referencia a todos y cada uno de los usuarios que intervienen en el sistema.
* **Evaluación:** La evaluación se debería hacer posterior teniendo métricas por ejemplo del rendimiento de la aplicación y saber si un cambio mejora o no dicho rendimiento.

*Patrón MVC de arquitectura de software*

Según (Alvarez, 2020), MVC es la propuesta de arquitectura de software que es usada para separar el código por su funcionalidad formando así capaz que se encargan de realizar tareas concretas por lo que ofrece mayores beneficios de un código limpio y escalable. Para este proyecto se hará uso del Modelo Vista Controlador ya que permite organizar el código de forma ordenada y simple. En la arquitectura MVC existe un controlador principal que es el encargado de gestionar las entradas y salidas de información, posteriormente el modelo se encarga de buscar los datos solicitados para que finalmente la interfaz muestra al usuario la información requerida, tal y como se puede observar en la figura 9.

Figura 9

*Patrón MVC para el diseño de sistemas*



*Fuente:* https://bit.ly/3Qtgfnp

### 2.2.9 Metodología de desarrollo de software

Según (Maida & Pacienzia, 2015), son aquellos conjuntos de métodos y técnicas organizados que permiten el diseño de soluciones informáticas que tienen como objetivo el organizar de forma clara y optimizada a los equipos de trabajo a fin que puedan desarrollar cada una de las funciones del sistema de la manera más ordenada.

Las metodologías de desarrollo tradicionales tienen como características el ser inflexibles al inicio del proyecto y que durante el proceso permiten realizar muy pocas variantes. Mientras que las metodologías agiles han incentivado el cambio progresivo conforme sea la necesidad de cada uno de los procesos en el desarrollo de un software. Al momento las metodologías agiles son usadas a gran escala mientras que las tradicionales han perdido mucha fuerza en los últimos tiempos. En cuanto a las metodologías agiles más importantes se tiene las siguientes:

*Metodología Extreme Programming o XP*

La metodología XP, es la metodología ágil más conocida, orientada a guiar equipos de trabajo pequeños o medianos en ambientes en donde las condiciones pueden variar. La principal particularidad de este tipo de metodología son las historias de usuario, mismas que corresponden a una técnica para especificar los requerimientos que tendrá un sistema. El Planning game es un proceso que permite fijar una fecha de entrega y el alcance del entregable. (Molina Montero, Vite Cevallos, & Cuesta, 2018)

Se realizan entregas pequeñas que son el uso de ciclos cortos de desarrollo, llamado iteraciones, que muestra al cliente una funcionalidad del software terminado y se obtiene una retroalimentación de él. La metodología XP trabaja en 5 fases las cuales están representadas en la figura 10.

Figura 10

*Diagrama de fases de la programación extrema o XP*



*Fuente:* https://bit.ly/3QpJZS8

*SCRUM*

Según (Drumond, 2021), Scrum es una metodología ágil que permite una gestión flexible del desarrollo de un sistema informático, con el principal objetivo de la optimización de recursos y asegurar el retorno de la inversión realizada por la empresa. El pilar fundamental es que el sistema tenga una gran funcionalidad que agrade al cliente, de modo que resulte un software innovador.

Esta metodología permite abordar proyectos complejos que exigen una flexibilidad y una rapidez esencial a la hora de ejecutar los resultados. La estrategia irá orientada a gestionar y normalizar los errores que se puedan producir en desarrollos demasiado largos, a través de, reuniones frecuentes para asegurar el cumplimiento de los objetivos establecidos.

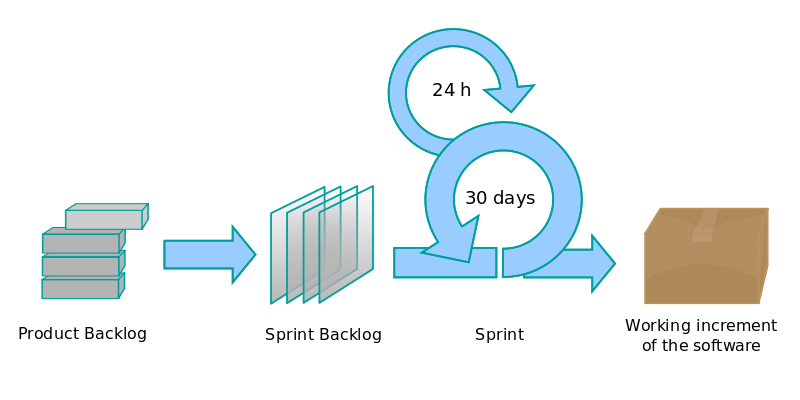
*Ventajas de Scrum*

Para (Drew, 2019), trabajar con Scrum tiene las siguientes ventajas:

* **Gestión de las expectativas del usuario:** Los usuarios participan de cada una de las etapas del proceso, además de proponer soluciones. Las evaluaciones se realizan de forma conjunta ya que en cada una de las etapas del proyecto los usuarios aportan ideas que mejoren al proyecto.
* **Resultados anticipados:** En Scrum cada una de las etapas del proceso arrojan resultados parciales, por lo que no se necesita el proyecto finalizado para realizar observaciones. Al finalizar cada Sprint pueden irse evidenciando cada uno de los logros, por lo que los tiempos de pruebas se acortan permitiendo así que el proyecto se finalice más rápido.
* **Flexibilidad y adaptación a los contextos:** Scrum se adapta a cualquier área de una gestión de trabajo, es decir no es una técnica disciplinaria.
* **Gestión sistemática de riesgos:** Al trabajar con Scrum se pueden contrarrestar los problemas en cada uno de los sprint o módulos, de forma que se pueden ir solucionando conforme existe un progreso del proyecto. La intervención de los equipos de trabajo se realiza al momento de que surjan problemas o dificultades en el curso del proyecto.
* **Sistema jerárquico de actividades:** Plantear este sistema de actividades a ejecutar durante el ciclo de vida del proyecto, permite que los colaboradores puedan darle prioridad a aquello que se requiera con mayor urgencia frente a aquellas tareas que no son las más demandantes.
* **Fechas realistas de entregas del proyecto:** En Scrum, al trabajar con iteraciones, se segmenta el objetivo a entregar lo que hace que los márgenes de error sean menores como así también, que las entregas finales se ajusten a lo que fue planificado. Así, los Sprint o iteraciones hacen más fácil de gestionar las tareas y un mejor manejo de los tiempos.

Figura 11

*Proceso de desarrollo con la metodología Scrum*



*Fuente:* https://bit.ly/3AodYEf

En la figura 11 se puede observar claramente la secuencia con la que trabaja SCRUM para la planificación de un proyecto. Se arranca con el Product Backlog, se procede a identificar los Sprint, se trabaja en cada uno de los Sprint y se va entregando el software poco a poco.

Para la presente propuesta se hará uso de la metodología SCRUM ya que permite realizar la entrega de cada uno de los módulos de forma progresiva, lo que es conveniente para la propuesta ya que se ira realizando el sistema poco a poco y se puede ir mejorando las funcionalidades del sistema conforme lo requiera el usuario final.

*Product Backlog en SCRUM*

Según (Molina, 2021), el product backlog es un listado de todas las tareas que se pretenden hacer durante el desarrollo de un proyecto. Cada una de las tareas debe constar en el product backlog, para que puedan estar a la vista de cada integrante del equipo de trabajo a fin de que se pueda visualizar cual será el producto final.

Para tener éxito en la elaboración de un product backlog es necesario que el mismo cumpla con los siguientes requisitos:

* Debe ser redactado de forma clara
* El product backlog debe dividirse en varias entradas que agreguen lo requerido por el cliente, en lo cual debe indicarse la importancia que tiene en el desarrollo. La importancia puede ser ALTA, MEDIA o BAJA.
* Incluye el trabajo técnico y el conocimiento necesario, es decir el equipo deba llevar a cabo un trabajo técnico con el objetivo de sacar adelante el proyecto.
* Definición de requisitos del producto a su priorización por categorías, pasando por su evolución constante y por la estimación del valor aproximado de su esfuerzo.

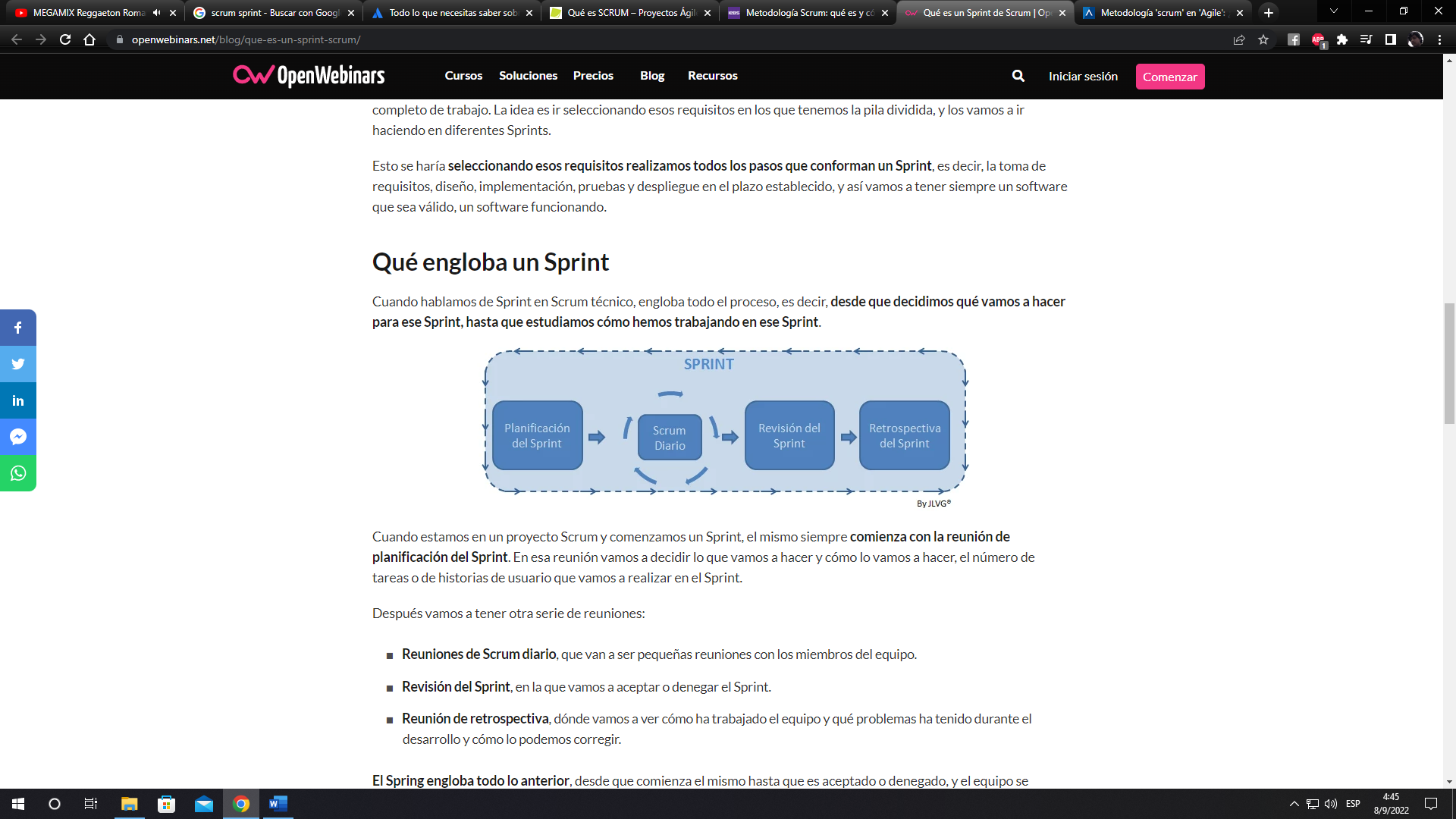
*Sprint en SCRUM*

Según (Sáez Hurtado, 2021), Sprint es el nombre asignado a cada una de las tareas o ciclos que van a realizarse dentro de un proyecto Scrum. Los Sprint permiten tener un ritmo de trabajo con un tiempo prefijado, siendo la duración habitual de un Sprint unas cuatro semanas, aunque lo que la metodología dice es que debería estar entre dos semanas y un máximo de dos meses.

En cada Sprint los resultados a conseguir se denominan entregables o incremento del producto lo cual será aportado al software del cliente. En cada uno de los sprint se consigue un producto, aunque se debe considerar que el mismo debe estar funcionando correctamente para ser considerado como terminado. En la figura 12 se puede observar el ciclo de un sprint.

Figura 12

*Ciclo de un sprint en SCRUM*



*Nota.* Fases que debe respetar cada uno de los sprint

*Fuente:* https://bit.ly/3AO607l

### 2.2.10 Aplicaciones móviles

Una Aplicación móvil o App es un programa que se instala en el Smartphone o Tablet del cliente. De esta manera la App pasa a formar parte del celular. Las aplicaciones permiten que un usuario pueda realizar tareas específicas del tipo profesional, educativas o de cualquier índole, lo cual permita facilitar la gestión de actividades o procesos. (Ríos Marín, 2019)

* **Aplicaciones Nativas:** Este tipo de aplicaciones pueden ser instaladas en el dispositivo, con lo que pueden utilizar los canales estándares de distribución de aplicaciones nativas, o bien incorporarse como accesos directos. (Ramirez Vique, 2019)
* **Aplicaciones Web Progresivas (PWA):** Este tipo de aplicaciones son una combinación de un sistema web y una aplicación nativa lo cual le permite al usuario final una mayor accesibilidad de navegación. Principalmente usan tecnología de programación web es decir HTML, CSS y JavaScript. (Caihuara Sossa, 2019)
* **Aplicaciones hibridas:** La aplicación hibrida se basa en el desarrollo de una página móvil, la cual tiene la capacidad de manejar los elementos del dispositivo móvil (GPS, sensores, entre otros). Las aplicaciones hibridas son capaces de funcionar en cualquier sistema operativo sin distinción de marca o fabricante. (Angulo, 2013)

En base a los conocimientos adquiridos en la carrera y la facilidad del lenguaje se ha decidido realizar la parte móvil de la propuesta en aplicaciones nativas, más concretamente en Android Studio con el lenguaje JAVA.

### 2.2.11 Programación para Android

Según (KeepCoding, 2022), para programar una aplicación nativa para Android, existen dos lenguajes programación, Java y Kotlin. Java es un lenguaje que cada día crece más gracias a su rapidez, fiabilidad y seguridad, mientras que Kotlin va tomando fuerza con el pasar del tiempo a pesar de que desde su inicio utiliza algunas características de Java.

Para el desarrollo de la aplicación nativa en Android, se usará el entorno de desarrollo de Android Studio ya que al hacerlo de forma nativa se tienen las siguientes ventajas que menciona (Imbaquingo Esparza, 2018):

* **Complementos:** Utiliza una fácil interface de instalación de Plug-in (Complementos), se puede verificar que complementos ya están instalados, cuales aún no están en el sistema y cuales se pueden actualizar.
* **Soporte:** La posibilidad de programar para Android Wear (dispositivos corporales), TV y otros dispositivos.
* **Mayor rendimiento:** Este tipo de aplicaciones aprovechan todas las funcionalidades del dispositivo móvil, como sensores, GPS, entre otras. Por lo que se puede hacer uso de las mismas sin ningún problema.
* **Funcionan sin internet:** Se puede desarrollar aplicaciones que no requieran de un servicio de internet para su funcionamiento, siempre y cuando no se vayan a utilizar herramientas o utilidades que requieran de internet.
* **Alertas:** Alertas en tiempo real de errores sintácticos, compatibilidad o rendimiento antes de compilar la aplicación.
* **Google Cloud Plataform:** Cabe destacar la integración con Google Cloud Plataform, esto beneficia al usuario que desarrolla el Proyecto en este IDE, ya que aprovecha los servicios de Google.
* **Widgets:** Una aplicación nativa puede hacer uso de widgets, es decir una vista miniatura de la app en la página principal del dispositivo móvil.

A continuación, se dará una breve descripción de Android Studio y el lenguaje JAVA el cual va a permitir el desarrollo de la aplicación para dispositivos móviles.

**Android Studio**

Las herramientas y aplicaciones desarrolladas para Android poseen un entorno de trabajo propio y este se denomina Android Studio, mismo que brinda una variada gama de herramientas y características que permiten el desarrollo del mismo. El uso de esta herramienta permite tener aplicaciones eficientes y autosuficientes ya que no requieren de un software extra para su funcionamiento. En algunos casos pueden ser compatibles con otros sistemas o plataformas. (Santaella, 2022)

*Características de Android Studio*

Según (Nielfa S, 2020), Android Studio tiene las siguientes características:

* Permite ejecutar compilaciones de forma rápida lo que permite la detección de fallos y por ende mejorar el código y funcionamiento de una aplicación.
* Realizar renderizados en tiempo real de las vistas de cada una de las escenas de una aplicación, además que pueden ser visualizados en un dispositivo real gracias al potente emulador que tiene la herramienta.
* Multidispositivos emulados en el entorno de desarrollo, lo que facilita la visualización de resultados en diversos dispositivos de forma simultánea.
* Creación de elementos gráficos en la interfaz, sin necesidad de hacerlo por código, sino con una paleta de elementos que el software proporciona para el diseño de pantallas profesionales.
* Es el IDE oficial de Android. Esto asegura el correcto funcionamiento del software, ya que es el mismo que utilizan los propios desarrolladores del sistema operativo oficial de Android para crear todas sus aplicaciones.

Figura 13

*Desarrollo de aplicaciones en Android Studio*



*Fuente:* https://bit.ly/3K3w5mk

**JAVA en Android Studio**

Java es una plataforma informática de lenguaje de programación creada por Sun Microsystems en 1995. Ha evolucionado desde sus humildes comienzos hasta impulsar una gran parte del mundo digital actual, ya que es una plataforma fiable en la que se crean muchos servicios y aplicaciones. Los nuevos e innovadores productos y servicios digitales diseñados para el futuro también siguen basándose en Java. (JAVA, 2022)

*Ventajas de JAVA*

Según (Carranza, 2022), el uso de Java en Android tiene las siguientes ventajas:

* **Tiene una curva de aprendizaje corta:** Una de las primeras ventajas de Java es que este lenguaje de programación es relativamente simple de entender.
* **Está orientado a los objetos:** El concepto de OOP (orientado a objetos) es lo que permite a los desarrolladores del lenguaje Java reutilizar el código en otros programas. Esto diferencia de la programación por procedimientos donde se requiere una secuencia de código para el uso de variables y funciones, mientras que en POO a los objetos se les puede asignar variables y funciones propias.
* **Multiplataforma:** Puedes utilizar el mismo programa en una computadora con Windows, Linux, servidor SUN, Solaris e incluso en un teléfono móvil.
* **Cuenta con liberación de memoria:** Los programadores de Java decidieron romper con el sistema tradicional acerca de la liberación de memoria, quitando esa responsabilidad al programa. La ventaja de usar Java es que se solicita memoria al sistema solo si es necesario.

Figura 14

*Programación en JAVA por medio del programa Android studio*



*Fuente:* https://bit.ly/3AigtXJ

*Componentes de JAVA*

Según (Muradas, 2021), los componentes de Java son los siguientes:

* **JDK (Java Development Kit):** Es un software a través del cual tendremos acceso a las herramientas de desarrollo Java, este puede ser instalado de forma local o en red, permitiendo a los programadores llevar a cabo sus desarrollos de código los cuales posteriormente serán ejecutados por la JRE que describiremos en breve.
* **JRE (Java Runtime Enviroment):** Es la maquina encargada de ejecutar y/o llevar la aplicación desarrollada al plano “físico” del resultado de la JVM; básicamente es la intermediaria entre Java y el sistema operativo.
* **JVM (Java Virtual Machine):** Se encuentra encargada de ejecutar el código Java previamente compilado y transformado a bytecode.

*Estándar de codificación en JAVA*

Según (Cantabria, 2020), los principales estándares para programar con Java son:

* **Paquetes:** Por defecto todos los paquetes se escribirán en minúsculas y sin utilizar caracteres especiales. El paquete base queda definido como *es.ecuador.org*, en este paquete no se definirá ninguna clase.
* **Nombre de interfaces:** Los nombres de interfaces utilizarán el sufijo Interface y estarán compuestos por palabras con la primera letra en mayúscula (Camel Case). Se debe evitar el uso de abreviaciones que dificulten la comprensión del código.
* **Nombres de clases:** Los nombres de clases deben ser mezclas de mayúsculas y minúsculas, con la primera letra de cada palabra interna en mayúsculas (Camel Case).
* **Métodos:** Los métodos deberán ser verbos (en infinitivo), en mayúsculas y minúsculas con la primera letra del nombre en minúsculas, y con la primera letra de cada palabra interna en mayúsculas (lowerCamelCase).
* **Variables:** Los nombres de las variables tanto de instancia como estáticas reciben el mismo tratamiento que para los métodos.
* **Comentarios:** Los comentarios serán utilizados para dar información adicional al desarrollador sobre la implementación del diseño de la clase. Se tiene, por tanto, que evitar referencias al diseño funcional de la misma.
* **Declaraciones:** Para la declaración de las variables se utiliza una declaración de cada vez y no se permiten dejar variables locales sin inicializar salvo en el caso de que sean propiedades de un objeto bean.
* **Sentencias:** Una sentencia por línea de código y Todo bloque de sentencias entre llaves, aunque sea una sola sentencia después de un IF.

### 2.2.12 Bases de datos

La base de datos consiste en una variedad de información interrelacionada que es almacenada sin daños ni redundancias de forma que un sistema informático tenga los datos almacenados en un espacio definido. La información almacenada puede ser gestionada por motores que permiten la interacción con los datos. En la actualidad existen motores específicos que pueden trabajar con datos complejos a fin de proporcionar informes. (Vasquez Lozano, 2019)

*Base de datos relacional*

Según (Ayudaley, 2020), la base de datos relacional es un conjunto de información que organiza los datos en relaciones definidas por el usuario, por medio de las cuales se puede almacenar una o más tablas con lo que se facilita la visualización y comprensión de cómo está hecha la relación entre los datos de aquellas tablas. Las relaciones son conexiones lógicas entre tablas y se establecen a partir del uso de ellas.

Los datos se encuentran distribuidos en tablas, mismas que se encuentran relacionadas entre sí. El modelo de datos relacional es una forma estándar de representar la información de un sistema informático. En un inicio los desarrolladores encontraron formidable el uso de tablas, ya que era una forma intuitiva, eficiente y flexible de almacenar información.

*Base de datos distribuida*

La base de datos distribuida consiste en almacenar partes de una base de datos en distintas ubicaciones físicas, por lo que el procesamiento se encuentra replicado en diversos puntos de una red de trabajo. Además, este tipo de bases de datos se caracterizan por almacenar la información en varios computadores facilitando así el acceso de cualquier usuario que tenga acceso a la red de la empresa. (Vasquez Lozano, 2019)

*Base de datos NoSQL*

El nombre NoSQL proviene de No solo SQL, es decir este tipo de bases de datos utilizan SQL como un apoyo, mientras que para las consultas utilizan una variedad de modelos de datos que permiten acceder y gestionar la información. Cuando se requiere manejar un gran volumen de para aplicaciones grandes, es normal que se use este tipo de base de datos gracias a su baja latencia y modelos de datos flexibles lo que se logra mediante la flexibilización de restricción de coherencia de datos en otras bases de datos. (Vasquez Lozano, 2019)

*Ventajas de usar una base de datos*

Según (Beynon Davies, 2018), las ventajas de trabajar con una base de datos son las siguientes:

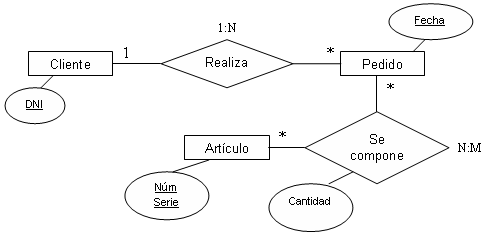
* **Control sobre redundancia de datos:** Los sistemas de ficheros almacenan varias copias de los mismos datos en ficheros distintos. Esto hace que se desperdicie espacio de almacenamiento, además de provocar la falta de consistencia de datos.
* **Consistencia de datos:** Eliminando o controlando las redundancias de datos se reduce en gran medida el riesgo de que haya inconsistencias. Si un dato está almacenado una sola vez, cualquier actualización se debe realizar sólo una vez, y está disponible para todos los usuarios inmediatamente.
* **Compartir datos:** En los sistemas de ficheros, los ficheros pertenecen a las personas o a los departamentos que los utilizan. Pero en los sistemas de bases de datos, la base de datos pertenece a la empresa y puede ser compartida por todos los usuarios que estén autorizados.
* **Mantenimiento de estándares:** Gracias a la integración es más fácil respetar los estándares necesarios, tanto los establecidos a nivel de la empresa como los nacionales e internacionales. Estos estándares pueden establecerse sobre el formato de los datos para facilitar su intercambio, pueden ser estándares de documentación, procedimientos de actualización y también reglas de acceso.
* **Mejora en la integridad de datos:** La integridad de una base de datos tiene relación con la validez y consistencia de la información almacenada. Por lo general se generan restricciones o reglas que no se puedan violar. Las restricciones pueden aplicarse directamente a los datos o a las relaciones.
* **Mejora en la seguridad:** La seguridad en la base de datos se realiza mediante autentificación de usuarios, por lo cual los usuarios no autorizados no pueden acceder a los datos. Sin medidas de seguridad la integridad de la información queda expuesta a cualquiera.

*Modelo entidad – relación*

Los diagramas o modelos entidad-relación son una herramienta para modelar los datos de un sistema de información, los cuales expresan a cada una de las entidades que intervienen en los procesos de un sistema, además de la relación que tienen entre sí y sus propiedades únicas. (Mendoza A. & López, 2018)

Figura 15

*Ejemplo de modelo entidad – relación*



*Nota.* Ejemplo de un modelo entidad – relación en una base de datos

*Fuente:* https://bit.ly/3VjtmKc

*Cardinalidad de las relaciones*

Según (Mendoza A. & López, 2018), el diseño de relaciones entre las tablas de una base de datos puede ser la siguiente:

* **Relaciones de uno a uno:** una instancia de la entidad A se relaciona con una y solamente una de la entidad B.
* **Relaciones de uno a muchos:** cada instancia de la entidad A se relaciona con varias instancias de la entidad B.
* **Relaciones de muchos a muchos:** cualquier instancia de la entidad A se relaciona con cualquier instancia de la entidad B.

En la actualidad existen algunos métodos para la gestión de una base de datos, dentro de lo más común es tenerla en una red local o en un entorno cloud, es decir almacenada en algún servidor web. Para el sistema es necesario una base de datos en la nube para lo cual se define un concepto de lo que significa una base de datos en la nube.

*Nube*

El método DBaaS (Database as a Service) se basa en la contratación de proveedores de servicios para el almacenamiento de los datos. En este caso, la información se guarda en los servidores del proveedor, pudiendo acceder a ella a través de internet. Para la presente propuesta se hará uso de una base de datos relacional para lo cual es importante conocer los principales gestores que pueden servir de forma adecuada en la propuesta.

Dentro de los gestores de bases de datos más utilizados se tiene a MySQL y PostgreSQL los cuales se describen a continuación para determinar la mejor opción para el proyecto.

**MySQL**

Según (Robledano, 2019), MySQL es un sistema de gestión de bases de datos que cuenta con una doble licencia. Por una parte, es de código abierto, pero por otra, cuenta con una versión comercial gestionada por la compañía Oracle. MySQL es el sistema de gestión de bases de datos relacional más extendido en la actualidad al estar basada en código abierto. Desarrollado originalmente por MySQL AB, fue adquirida por Sun MicroSystems en 2008 y esta su vez comprada por Oracle Corporation en 2010, la cual ya era dueña de un motor propio InnoDB para MySQL.

*Características de MySQL*

MySQL presenta algunas ventajas que lo hacen muy interesante para los desarrolladores. La más evidente es que trabaja con bases de datos relacionales, es decir, utiliza tablas múltiples que se interconectan entre sí para almacenar la información y organizarla correctamente.

1. *Flexible y fácil de usar:* El proceso de instalación es relativamente simple y no debería durar más de 30 minutos. Además, se puede modificar el código fuente a necesidad del programador.
2. *Alto rendimiento:* Un amplio compendio de servidores de clúster respalda a MySQL. Ya sea que se esté almacenando enormes cantidades de datos de e-Commerce grandes o realizando actividades intensas de inteligencia de negocios, MySQL puede trabajar sin problemas con una velocidad óptima.
3. *Seguro:* Losdatos deberían ser la principal preocupación al elegir el software RDBMS correcto. Con su sistema de privilegios de acceso y la administración de cuentas de usuario, MySQL establece un alto estándar de seguridad.

Figura 16

*MySQL como base de datos relacional*



Fuente: https://bit.ly/3R5Aist

**PostgreSQL**

PostgreSQL en un sistema que gestiona bases de datos relacionales y que permite la extensibilidad y cumplimiento de SQL. Es un software de código abierto y que tiene como característica importante el MVCC, mismo que permite agregar una imagen del estado de la base de datos por cada transacción. Lo que mejora en rendimiento y consistencia de la información. (Cesar, 2015)

*Características de PostgreSQL*

Según (Vazquez Ortíz, Mier Pierre, & Sotolongo León, 2016), PostgreSQL presenta las siguientes características:

1. El almacenamiento de datos de forma ágil.
2. El almacenamiento de datos libre de esquemas.
3. La lectura de datos relacionales de una tabla y su retorno como JSON y viceversa, haciendo uso de sus operadores y funciones.
4. La integración fácil de sentencias convencionales SQL con tipos de datos JSON y HSTORE, ejecutadas en el mismo entorno transaccional ACID y basados en el mismo planificador de consultas, optimizador y tecnologías de indexado.
5. Permite crear esquemas, tablas heredadas y triggers orientados a eventos que no poseen otros motores.

Una vez descritos los conceptos necesarios, se determinó que MySQL es el gestor adecuado para el proyecto gracias a las siguientes ventajas mencionadas por (HostingPedia, 2019):

* MySQL es de uso libre y gratuito.
* Bajo costo en requerimientos para la elaboración y ejecución del programa.
* No se necesita disponer de Hardware o Software de alto rendimiento para la ejecución del programa.
* Velocidad al realizar las operaciones y buen rendimiento.
* Facilidad de instalación y configuración.
* Soporte en casi el 100% de los sistemas operativos actuales.
* Baja probabilidad de corrupción de datos.
* Entorno con seguridad y encriptación.

## 2.3 Antecedentes de investigación

Para determinar las herramientas adecuadas para el diseño de la propuesta se realizará una comparativa entre proyectos de similares características, tomando en cuenta la metodología de desarrollo, el lenguaje de programación, la base de datos y las funcionalidades de las propuestas.

El proyecto de investigación de Aedo, Edwin denominado “Sistema web para la optimización de la gestión de ventas de una empresa comercializadora”, buscó la forma de reducir los tiempos de atención a los clientes y los costos operativos en la entrega de cilindros de gas. En la investigación se utilizó la metodología de desarrollo RUP, la cual permitió la entrega del proyecto en fases para garantizar así el cumplimiento de cada de los requisitos en función de la solución de cada uno de los problemas de la empresa de gas. En cuanto al lenguaje de programación del sistema web es de PHP por medio del framework LARAVEL y una base de datos MySQL. En la empresa se pudieron encontrar algunos problemas que fueron solucionados con el sistema, entre ellos la cartera de clientes, el despacho de los productos de una manera más sencilla y la venta de los mismos por medio de una facturación automatizada. (Humanñahui Aedo, 2021)

En cuanto a la propuesta presentada por Carlosama Jingo y Fárez Rodríguez denominada “Desarrollo de un sistema prototipo para la automatización de los procesos de venta e inventario de un distribuidor de gas licuado de petróleo”, el prototipo fue diseñado bajo la metodología SCRUM que por medio de sketches basados en los requerimientos de usuario permitió la entrega tanto de la aplicación web, así como también de la aplicación móvil. En cuanto a la programación, se utilizó JavaScript, Node.js para la creación de un API REST, la base de datos fue almacenada en el gestor MySQL y finalmente el API REST fue almacenado en un servidor AWS para un mejor envió y recepción de la información que almacena el prototipo de sistema web y móvil. En cuanto a funcionalidades el prototipo permite la gestión de clientes, solicitud de GLP, seguimiento a la compra, asignación de repartidor, visualización de pedidos, trazado de rutas y reportes. (Carlosama Jingo & Fárez Rodriguez, 2022)

También se logró encontrar un proyecto similar el cual fue presentado por Acurio Christian denominado “Desarrollo de una aplicación móvil que envíe una solicitud de requerimiento de gas doméstico a los distribuidores o a los vehículos de distribución más cercanos a una ubicación en la ciudad de Quito mediante Georreferenciación”, el cual por medio de un aplicativo móvil permite a los clientes solicitar un producto de manera rápida y sin complicaciones. Los distribuidores pueden registrarse por medio de la aplicación móvil para poder estar disponibles para que los clientes soliciten el servicio de reposición del cilindro de gas. La propuesta fue desarrollada por medio de la metodología XP, la cual permitió la identificación de requerimientos funcionales para la aplicación móvil, también se pudo determinar que el lenguaje de programación utilizado fue JAVA en el programa Android Studio. La base de datos remota fue realizada en MySQL y Windows Azure. La app permite el registro de clientes y distribuidores, la solicitud de un cilindro de gas, visualizar un mapa con la ubicación e información de los distribuidores, escoger un distribuidor por medio de georreferenciación, envío de mensajes de texto y envío de coordenadas del cliente. (Acurio Acurio, 2017)

Finalmente, el proyecto de investigación de Eras Frank e Hidalgo Pablo denominado “Aplicación móvil en Android para la adquisición de Gas Licuado de Petróleo (GLP) doméstico”, mismo que buscó la forma de solucionar la ineficaz manera de comercializar gas de uso doméstico, por lo cual el desarrollo de un aplicativo móvil busca mejorar dicho problema y disminuir la congestión vehicular, contaminación sonora y contaminación ambiental. La propuesta recopilo los requerimientos funcionales en base a las necesidades presentadas por los clientes, mientras que para el desarrollo de la aplicación se utilizó JAVA en Android Studio, para el almacenamiento de información FIREBASE y las funcionalidades de geolocalización y georreferenciación las API de Google. La aplicación móvil permite al cliente realizar un pedido, notificaciones al distribuidor, aceptación o rechazo de una solicitud de gas, cancelación del pedido, registro de clientes y distribuidores y un historial de pedidos tanto para distribuidores como para clientes. (Eras Camacho & Hidalgo Lascano, 2021)

Una vez analizados los antecedes de propuestas similares, se procedió a realizar la tabla comparativa que permita identificar ciertas funcionalidades que deben ser parte de la propuesta, la mejor herramienta en cuanto a lenguajes de programación, la base de datos y la metodología de desarrollo más adecuada que permita el diseño e implementación del prototipo que facilite la comercialización de cilindros de GLP en la ciudad del Tena. Para lo cual denominaremos a las propuestas con un número para facilitar la comparación entre ellas:

1. Propuesta Aedo Edwin
2. Propuesta Carlosama – Fárez
3. Propuesta Acurio Christian
4. Propuesta Eras Hidalgo

Tabla 1

*Comparativa entre propuestas similares*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Ítem | Propuesta 1 | Propuesta 2 | Propuesta 3 | Propuesta 4 |
| Metodología de desarrollo | | | | |
| RUP | X |  |  |  |
| SCRUM |  | X |  |  |
| XP |  |  | X |  |
| Ninguna |  |  |  | X |
| Metodología ágil de desarrollo |  | X | X |  |
| Lenguaje de Programación | | | | |
| Utilizó un framework de desarrollo | X |  |  |  |
| Utilizó BackEnd y FrontEnd | X | X |  |  |
| Plataforma web para administración | X | X |  |  |
| Aplicativo móvil |  | X | X | X |
| Utilizó un lenguaje de programación actual (JAVA, Kotlin, JavaScript, PHP, Python) | X | X | X | X |
| Autentificación con API externas |  |  |  | X |
| Base de datos | | | | |
| Base de datos relacional | X | X | X |  |
| Base de datos no relacional |  |  |  | X |
| Base de datos externa (Nube) | X | X | X | X |
| Base de datos libre o gratuita | X | X | X |  |
| Funcionalidades | | | | |
| Autentificación para ingresar | X | X | X | X |
| Gestión de Clientes | X | X | X | X |
| Gestión de Usuarios | X | X | X | X |
| Gestión de Distribuidores | X | X | X | X |
| Trazado de rutas |  | X | X | X |
| Notificaciones o alertas |  | X | X | X |
| Geolocalización |  | X | X | X |
| Georreferenciación |  | X | X | X |
| Gestión de pedidos | X | X | X | X |
| Facturación | X |  |  |  |
| Reportes | X | X |  |  |

*Nota.* Comparativa de metodologías, lenguaje de programación, base de datos y funcionalidades

*Fuente:* Elaborado por el tesista

Una vez analizados cada uno de los puntos de las propuestas similares, se procedió a escoger las herramientas con las que se hará el desarrollo del presente proyecto y son las siguientes:

Tabla 2

*Herramientas informáticas escogidas para la propuesta*

|  |  |
| --- | --- |
| Ítem | Herramienta |
| Lenguaje de programación para el BackEnd | JavaScript, Node.js |
| Herramientas para el FrontEnd | Java y XML |
| Lenguaje de programación para la aplicación móvil | JAVA |
| Base de datos | MySQL |
| Metodología de desarrollo | SCRUM |

*Nota.* Detalle de las herramientas que serán utilizadas en el desarrollo de la propuesta

*Fuente:* Elaborado por el tesista

# Capitulo tres

# Arquitectura hardware y software del sistema

## 3.1 Generalidades del capitulo

En este capítulo se trata de manera general los equipos que servirán para el funcionamiento de la aplicación móvil. A nivel de software se mencionan algunas herramientas y servicios tanto para el BackEnd, FrontEnd y las aplicaciones lo cual se detalla a continuación. Para el BackEnd se hará uso de Node.js, mientras que como servidor se usará express lo cual proporciona un conjunto sólido de características para las aplicaciones móviles. En cuanto a la conexión a la base de datos se utilizará un paquete de código (npm) de MySQL para configurar y permitir el acceso a los datos almacenados en la misma.

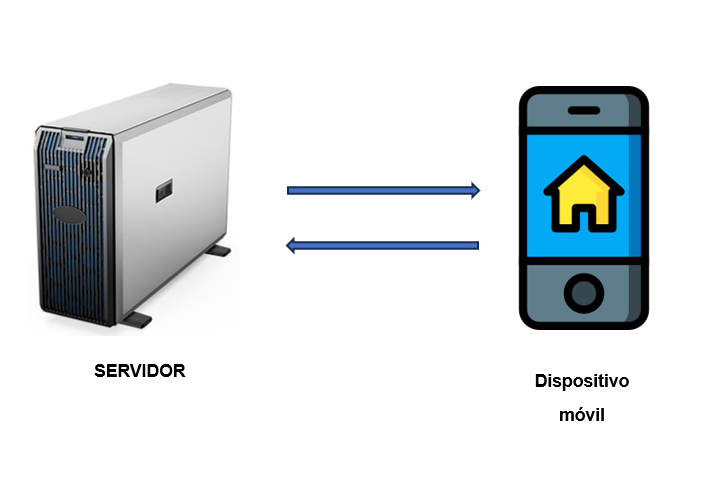
Para la aplicación móvil se usará el programa Android Studio, el lenguaje de programación escogido es JAVA y se conectaran a la base de datos por medio de solicitudes HTTP al BackEnd que contiene la conexión a la base de datos.

## 3.2 Arquitectura hardware

En la arquitectura de hardware se describe el conjunto de dispositivos físicos que harán posible el funcionamiento de la propuesta del proyecto, ya que por medio de la interconexión entre ellos permitirá el funcionamiento correcto del sistema. En la figura 17 se presenta la propuesta de arquitectura de hardware.

Figura 17

*Arquitectura hardware del sistema*



*Fuente:* Elaborado por el tesista

Para la propuesta es necesario al menos dos dispositivos móviles, y un servidor que sea capaz de soportar las tecnologías utilizadas en el desarrollo del sistema.

### 3.2.1 Servidor

Es un aparato informático que almacena, distribuye y suministra información. Los servidores funcionan basándose en el modelo “cliente-servidor”. El cliente puede ser tanto un ordenador como una aplicación que requiere información del servidor para funcionar. Por tanto, un servidor ofrecerá la información demandada por el cliente siempre y cuando el cliente esté autorizado. Los servidores pueden ser físicos o virtuales. (European Knowledge Center, 2019)

Para el sistema es necesario un servidor de tipo web, mismo que guardará y organizará el contenido del BackEnd para la conexión de la plataforma administrativa y las aplicaciones móviles. También servirá para almacenar el FrontEnd del administrador y finalmente la base de datos. Para ello es importante mencionar las características de un servidor del tipo web.

### 3.2.2 Servidor web

Un servidor web es un tipo especial de servidor de aplicaciones que aloja programas y datos solicitados por los usuarios a través de internet o en una intranet. Los servidores web responden a las solicitudes de páginas web u otros servicios basados en la web que llegan de los navegadores o aplicaciones que se ejecutan en los ordenadores o dispositivos cliente. (Romero, 2019)

*Características de los servidores web*

Según (Borges, 2018), un servidor web debe tener las siguientes características:

* **Capacidad de almacenamiento:** La cantidad de gigas disponibles en el disco duro para almacenar tu web, tu correo y todo lo necesario para el correcto funcionamiento del sistema.
* **Transferencia de datos:** Se trata del volumen o cantidad de datos que se transfieren, cada mes, del servidor hacia Internet.
* **Sistema operativo:** Una de las características de los servidores web es que utilizan un sistema operativo. Éste puede ser de diversos tipos como lo son los clásicos Unix, Linux o Windows.
* **Sistema de archivos:** El sistema de archivos es lo que permite que el sistema operativo tenga una idea de la estructura de datos que se almacenan en él. En otras palabras, es una guía lógica para que el sistema pueda ubicar, ordenar y filtrar datos en el disco duro, con el fin de leerlos, modificarlos o eliminarlos.
* **Software servidor HTTP:** El software que permite que se hospeden diferentes sitios web, así como soportar tecnologías como el hosting virtual se denomina software HTTP o servidor HTTP.
* **Panel de control:** Los paneles de control nos permiten y facilitan realizar diferentes tareas dentro de un servidor, haciendo que podamos introducir complejos cambios en el server desde un entorno gráfico con un par de clics.
* **Seguridades:** Dentro de las principales características para considerar un servidor seguro deben estar las siguientes:
  + Imponer límites de acceso por dirección IP.
  + Denegar o permitir acceso a ciertos archivos o URL’s.
  + Solicitar usuario y contraseña por autenticación básica HTTP.
  + Realizar un filtrado de peticiones inseguras con módulos como ModSecurity.
  + Soporte para despachar información cifrada con certificados de seguridad SSL vía HTTPS.
* **Soporte y administración web:** Una de las características más apreciadas y costosas es el departamento técnico que está detrás del teléfono, el día que todo falla. Cuando la web se cae, no es lo mismo tener que esperar 1 semana que 1 hora a que todo vuelva a la normalidad. No es lo mismo tener que pasar por 1 llamada que por 10.

Figura 18

*Servidor web para el sistema*



*Nota.* Características necesarias para tener un buen servidor web

*Fuente:* https://bit.ly/3GUhWID

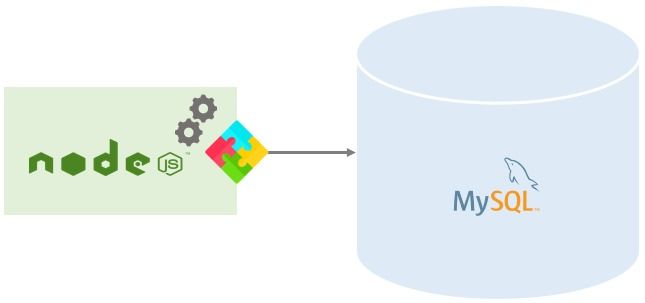
## 3.3 Arquitectura de software

El software utilizado en el desarrollo de la propuesta es de código abierto, no solo para no incurrir en gastos de licencias, sino también para utilizar las herramientas que fueron enseñadas a lo largo de la carrera. Las herramientas usadas para la programación del BackEnd, el FrontEnd, las aplicaciones móviles y la base de datos son las siguientes:

JavaScript con su entorno de ejecución multiplataforma Node.js, mismo que servirá para la creación del BackEnd. Además, se utilizará Express.js que permita construir la aplicación web que sirva de acceso a la información tanto de las aplicaciones móviles como de la plataforma administrativa.

Figura 19

*Conexión entre node.js y MySQL*



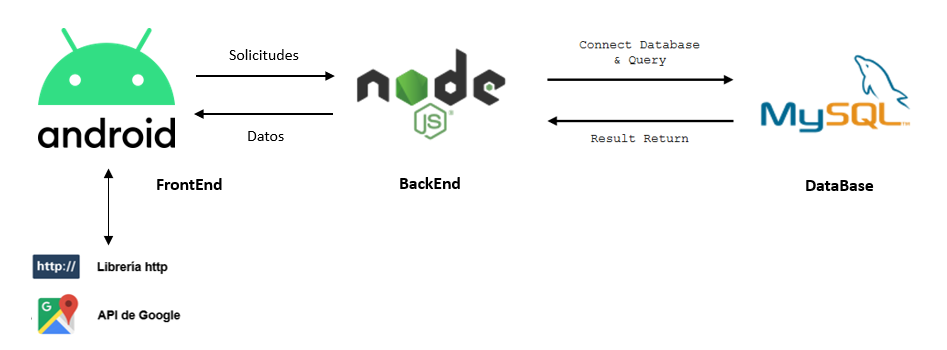
*Nota.* BackEnd conformado por Node.js y MySQL

*Fuente:* https://bit.ly/3Vi39LY

Por otra parte, para las aplicaciones móviles se usará el IDE Android Studio y el lenguaje JAVA para la creación de las aplicaciones móviles, además se hará uso de la librería Http para conectar a las aplicaciones con el BackEnd, mientras que, para las funcionalidades de geolocalización, georreferenciación, entre otras opciones de mapas y localizaciones se hará uso de la API de Google. Más adelante se detallará los conceptos descritos, por ahora en la figura 20 se muestra la arquitectura de software que usará el sistema propuesto.

Figura 20

*Arquitectura de software del sistema*



*Nota.* Herramientas que forman la arquitectura de software

*Fuente:* Elaborado por el tesista

### 3.3.1 Librería HTTP

Según (Smith, 2012), el protocolo más utilizado para la comunicación entre una aplicación Android y la base de datos es HTTP, el mismo que utilizan las páginas web. En Android Studio es posible realizar peticiones al BackEnd para poder enviar y recibir información de la base de datos para ello es necesario consumir los servicios Http por medio de la clase HttpUrlConnection.

El cliente Http asincrónico trabaja sobre librería de Apache, las solicitudes se realizan en un subproceso de la interfaz de usuario de la app, mientras que en el mismo proceso se ejecuta la devolución de mensajes en el controlador de Android. La librería maneja las siguientes características:

* Soporte para parámetros en el request.
* Subida de archivos.
* Soporte para cookies.

***Solicitudes HTTP***

Según (Mdn Web Docs HTTP, 2022), para el intercambio de información es necesario realizar petición por medio del protocolo HTTP, para lo cual se hará uso de los métodos más utilizados y son los siguientes:

* ***GET:*** El método GET es usado para solicitar un recurso específico, las peticiones realizadas por este método deben ser usadas únicamente para recuperar información del BackEnd.
* ***HEAD:*** Este método es similar al GET con la diferencia que no lleva un body o cuerpo con datos.
* ***POST:*** Por medio de este método se puede enviar un formulario o entidad para crear un nuevo recurso en la base de datos. Por lo general este método desencadena un cambio de estado o un efecto secundario en el BackEnd.
* ***DELETE:*** El presente método elimina un recurso especificado por lo general por un ID.

### 3.3.2 API de Google

Según (Trafaniuc, 2022), API es la sigla de Application Programming Interface (Interfaz de Programación de Aplicaciones). Se trata de un conjunto de aplicaciones que permiten la construcción de una interfaz inteligente, configurando un medio por el cual dos sistemas se comunican. Google ofrece un gran número de estas extensiones que se encuentran a disposición en Google Maps, para la integración y visualización en diversos sitios y aplicaciones, como Uber, por ejemplo, que posibilita a los usuarios diversas acciones, tales como:

* Saber dónde se encuentra el conductor
* Cuánto tiempo tardará en llegar
* Acompañamiento del recorrido
* Simulación del costo del viaje
* Tiempo del viaje

Para el presente proyecto es necesario el uso de Maps, Routes y Places las cuales serán descritas a continuación.

*Maps*

Permite a los usuarios visualizar el mundo real, por medio de mapas estáticos o interactivos, que pueden ser personalizados e incorporados a sitios o aplicaciones. Por medio de esta funcionalidad se puede acceder a los SDK de mapas de Google no solo para Android sino también para JavaScript, que en el caso de la propuesta es necesario para la plataforma administrativa. A continuación, todas las funciones a las que se puede acceder con la API de Maps:

* SDK de Maps para Android: permite agregar mapas con base en los datos de Google Maps a tu aplicación para Android.
* SDK de Maps para iOS: permite agregar mapas con base en los datos de Google Maps a tu aplicación para iOS.
* API JavaScript de Maps: permite agregar mapas interactivos y personalizarlos con tu propio contenido e imágenes para exhibición en páginas de la web y dispositivos móviles.
* API Static de Maps: permite incorporar una imagen estática de Google Maps a tu sitio web, sin la necesidad de usar JavaScript o cualquier cargamento dinámico de la página.
* API Street View: permite incorporar un panorama o una miniatura estática (no interactiva) de Street View a tu página Web, sin el uso de JavaScript.
* URL’s de Maps: permite crear una URL universal entre plataformas, que posibilita abrir Google Maps y realizar búsquedas, teniendo acceso a rutas, mapas e imágenes panorámicas.

Figura 21

*API de Maps*



*Nota.* Maps para ser integrados en el sistema

*Fuente:* https://bit.ly/3gHRMxQ

*Routes*

Permite al usuario a encontrar el mejor trayecto hasta su destino, con la sugerencia de rutas y actualizaciones de tránsito en tiempo real. Además de eso, también permite crear itinerarios hasta de 25 puntos de referencia, ideal para un planeamiento de rutas de entrega más eficiente. Las funciones de entrega de cilindros de gas y ubicación de los clientes se podrán realizar por medio de esta opción de la API de Google (Trafaniuc, 2022). A continuación, todas las funciones a las que se puede acceder con la API de Routes:

* API Directions: permite proporcionar rutas de transporte público, bicicleta, auto y a pie.
* API Distance Matrix: proporciona la distancia y el tiempo de viaje entre los puntos de origen y de destino de una ruta recomendada.
* API Roads: posibilita crear itinerarios precisos, determinando el trayecto a ser recorrido, además de proporcionar datos al respecto de esas rutas, como el límite de velocidad.

Figura 22

*API de Routes de Google*



*Nota.* Características disponibles de la API de Routes de Google

*Fuente:* https://bit.ly/3Ue2OIP

*Places*

Places ayuda a los usuarios a conocer y explorar el mundo a su alrededor ofreciendo datos de millones de lugares. Además, posibilita encontrar locales específicos, usando números de teléfono, direcciones o nombres de establecimientos. Se trata de una forma óptima de facilitar a los clientes a encontrar lugares específicos, teniendo acceso a informaciones detalladas sobre negocios o lugares, como nombre del lugar, dirección, evaluaciones y datos de contacto (Trafaniuc, 2022). Las características que permite acceder al usar la API Places de Google son:

* API Places: exhibe información actualizada sobre millones de locales usando solicitudes HTTP. Los locales son definidos dentro de esta API como establecimientos, localizaciones geográficas o puntos de interés.
* SDK de Places para Android: permite crear aplicaciones para Android con reconocimiento de localización que respondan contextualmente a las empresas locales y a otros lugares próximos al dispositivo.
* SDK de Places para iOS: permite crear aplicaciones para iOS con reconocimiento de localización que responda contextualmente a las empresas locales y a otros lugares próximos al dispositivo.
* Places Library, API JavaScript de Maps: permiten que la aplicación investigue lugares como establecimientos, locales geográficos o puntos de interés, contenidos en un área definida, como dentro de los límites de un mapa o en torno de determinado punto.
* API Geocoding: permite convertir direcciones en coordenadas geográficas o viceversa.
* API Geolocation: permite encontrar la localización de un dispositivo con base en datos proporcionados por torres de celular y puntos de Wi-Fi.
* API Time Zone: permite ver la zona horaria de una coordenada específica de latitud y longitud.
* API Elevation: proporciona datos de elevación de cualquier punto del mundo, incluyendo la profundidad, como el fondo del océano (que devuelve valores negativos).

Figura 23

*API de Places de Google*



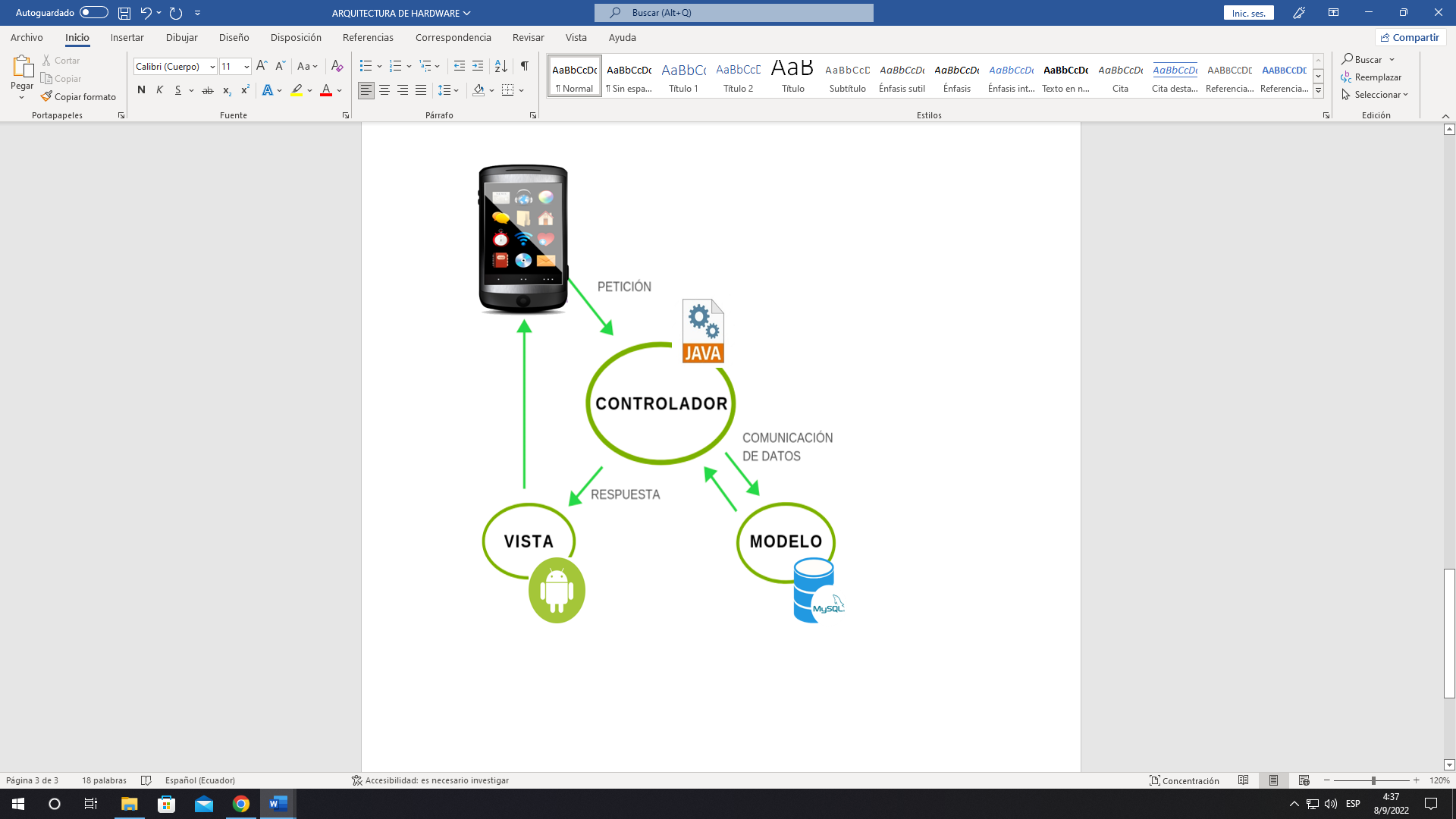
*Nota.* Características de Places de Google

*Fuente:* https://bit.ly/3XT6rHx

## 3.3 MVC de la aplicación móvil

Figura 24

*MVC de la aplicación móvil*



*Fuente:* Elaborado por el tesista

El patrón MVC permite una mejor estructura de un proyecto de software, con lo cual para el presente proyecto el controlador estará en lenguaje JAVA y se encarga de gestionar las peticiones y respuestas. La vista está diseñada en Android y por último el modelo de datos será gestionado por MySQL; de esta forma se cumple con lo requerido para tener un patrón MVC.

# Capítulo cuatro

# Diseño e implementación del prototipo de aplicativo móvil

En el presente capítulo se detalla todos los componentes técnicos necesarios para la especificación de requerimientos que tendrá el sistema, el diseño de los diferentes diagramas de funcionalidad del sistema, los prototipos de interfaz de usuario, el diseño de la base de datos, etc. Todo lo mencionado anteriormente está trabajado bajo la metodología ágil SCRUM.

Para empezar con el diseño y desarrollo de la propuesta se procedió a realizar la lista de requerimientos funcionales y no funcionales que serán necesarios para todo el sistema.

## 4.1. Requerimientos funcionales

* La aplicación debe permitir el registro de un nuevo cliente
* La aplicación debe controlar que no exista un cliente registrado con cédulas de identidad iguales.
* La aplicación debe controlar que el cliente ingrese una cédula de identidad válida.
* La aplicación debe permitir la recuperación de la contraseña por medio de un correo electrónico registrado previamente.
* La aplicación envía un correo con una contraseña temporal para el inicio de sesión.
* La aplicación debe permitir el ingreso a las funcionalidades únicamente por medio del ingreso de credenciales de seguridad.
* La aplicación debe almacenar las credenciales de sesión para cuando el usuario ingrese a la aplicación posteriormente no tenga que loguearse de nuevo.
* La aplicación debe permitir el ingreso a usuarios del tipo Cliente y Repartidor.
* La aplicación debe dirigir al usuario que inicio sesión al menú principal que le corresponda con relación a su rol de usuario.
* La aplicación debe permitir la visualización del perfil de usuario para poder editar esa información.
* La aplicación debe solicitar al usuario permiso para el uso del GPS que será utilizado para la ubicación desde donde realizará el pedido.
* La aplicación debe permitir realizar nuevos pedidos, añadiendo un producto y dando la cantidad.
* La aplicación debe controlar que al crear un nuevo pedido el usuario ha escogido una ubicación en el mapa de Google.
* La aplicación debe crear un nuevo pedido con estado generado.
* La aplicación debe permitir visualizar el historial de pedidos realizados por el usuario.
* La aplicación debe permitir que el usuario repartidor pueda visualizar los pedidos que le han sido asignados.
* La aplicación debe permitir que el usuario repartidor pueda acceder al detalle de un pedido que se le ha sido asignado para poder visualizar la ubicación del cliente.
* La aplicación debe trazar la ruta entre el cliente y el repartidor.
* La aplicación debe permitir que el usuario repartidor finalice un pedido.
* La aplicación debe permitir crear nuevos usuarios repartidores
* La aplicación debe permitir el eliminar a los repartidores
* La aplicación debe permitir visualizar reporte de ventas

### 4.1.1. Requerimientos no funcionales

El sistema ha considerado que ciertas funciones pueden ser importantes para el sistema mas no imprescindibles ya que no afectan directamente al ciclo de funcionamiento de la propuesta. Para ello se han tomado en cuenta los siguientes requerimientos:

* El sistema debe enviar notificaciones a la aplicación, acerca del estado de los pedidos.
* El sistema debe compartir la ubicación real de los usuarios del tipo repartidor a fin que la aplicación pueda asignar pedidos a quienes se encuentren más cerca de la ubicación del cliente.
* El sistema debe compartir el estado de los pedidos en tiempo real.
* La aplicación debe almacenar en el cache del teléfono la información de la sesión iniciada.
* Las contraseñas de los usuarios deben ser encriptadas u ofuscadas a fin de brindar mayor seguridad.

## 4.2. Producto backlog

La tabla 3 muestra el producto backlog en donde se especifica cada uno de los procesos que intervienen en el sistema y serán desarrollados, además de ser la base para obtener cada uno de los sprint.

Tabla 3

*Product backlog del sistema*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ID | Historia de Usuario | Estado | Criterios de aceptación | Prioridad | Responsable | Sprint |
| 1 | El usuario puede ingresar al sistema únicamente con credenciales. | Planificado | * El acceso debe ser mediante un usuario y contraseña. * No pueden acceder al sistema sin autentificarse. * Se puede almacenar las credenciales para futuros ingresos. | Alta | Tesistas | 1 |
| 2 | El usuario puede registrarse desde el login. | Planificado | * Se puede llenar un formulario para la creación de nuevo usuario y repartidor. | Media | Tesistas | 1 |
| 3 | El usuario puede recuperar su contraseña. | Planificado | * Se debe solicitar un correo para el envío de la contraseña olvidada. * El correo ingresado debe coincidir con algún registro almacenado para acceder a la recuperación de contraseña. | Media | Tesistas | 1 |
| 4 | Como usuario puedo crear nuevos usuarios. | Planificado | * Crear nuevo usuario mediante formulario validado. * No se puede registrar un usuario con cédula ya registrada. * Se debe especificar el rol del usuario en el sistema. | Media | Tesistas | 2 |
| 5 | Como usuario se puede eliminar los usuarios. | Planificado | * Se puede editar la información del usuario desde perfil. * No se puede modificar la cedula con la que se registró. * Se puede eliminar un usuario registrado. | Media | Tesistas | 2 |
| 6 | Como usuario se puede crear nuevos clientes. | Planificado | * Crear nuevo cliente mediante formulario validado. * No se puede registrar un cliente con una misma cédula o RUC ya registrado. | Alta | Tesistas | 3 |
| 7 | Como usuario se puede eliminar los clientes. | Planificado | * Se puede eliminar un cliente registrado. | Media | Tesistas | 3 |
| 8 | El usuario puede visualizar el inventario de productos. | Planificado | * Se puede visualizar los productos registrados en el sistema. | Alta | Tesistas | 4 |
| 9 | Como usuario se puede realizar pedidos de productos. | Planificado | * Se puede realizar un pedido a domicilio a través del aplicativo. * Se valida que los pedidos a domicilio deben tener la ubicación activada. * El aplicativo debe solicitar permiso para el uso del GPS. | Alta | Tesistas | 5 |
| 10 | El usuario puede visualizar la lista de pedidos. | Planificado | * Se puede observar el detalle de los pedidos registrados desde el aplicativo. * Se puede visualizar la ubicación del cliente que realizo el pedido. * El sistema debe lanzar una notificación al recibir un nuevo pedido a domicilio. | Alta | Tesistas | 6 |
| 11 | El sistema debe asignar un repartidor de forma automática a los nuevos pedidos generados. | Planificado | * Se puede asignar un pedido a un usuario repartidor que se encuentre disponible. * Se envía una notificación al usuario repartidor para avisar de la nueva asignación. * No se puede asignar un pedido a un usuario repartidor que se encuentra con estado “NO DISPONIBLE”. | Alta | Tesistas | 6 |
| 12 | El sistema debe trazar la ruta de entrega desde la ubicación actual del conductor hacia el cliente. | Planificado | * Se traza la ruta desde la ubicación actual del usuario repartidor, hacia la ubicación registrada por el cliente. * Se debe solicitar permiso para acceder a los datos móviles del dispositivo móvil. * Solo se accede a la ubicación del cliente si el pedido fue asignado al repartidor. | Alta | Tesistas | 7 |
| 13 | El sistema debe notificar cuando el repartidor llega a la ubicación del cliente. | Planificado | * El sistema debe enviar una notificación al cliente, notificando la llegada del usuario repartidor. * El sistema debe permitir a los usuarios repartidor, finalizar un pedido. * La notificación debe tener prioridad alta. | Alta | Tesistas | 7 |
| 14 | El usuario puede finalizar un pedido a domicilio. | Planificado | * El sistema debe permitir finalizar un pedido que ya ha sido entregado. * El sistema debe permitir cancelar un pedido. * El aplicativo debe permitir a un cliente cancelar el pedido. | Alta | Tesistas | 7 |

*Nota.* Detalle de cada uno de los procesos que intervienen en el sistema.

*Fuente:* Elaborado por el tesista

### 4.2.1. Especificación de roles

Para el desarrollo del sistema se identifica como interesado al usuario de tipo administrador, mismo que será el encargado de la creación y asignación de roles de usuario, la administración de los clientes y la gestión de los pedidos. Los tesistas del presente trabajo son los encargados del diseño, revisión e implementación del sistema. Los roles se describen en la tabla 4 a continuación.

Tabla 4

*Roles para el desarrollo del sistema*

|  |
| --- |
| Rol |
| Product Owner (Interesado o dueño del sistema) |
| Scrum Master (director del proyecto) |
| Development Team (Desarrolladores) |
| Stakeholder (Interesados) |

*Nota.* Roles que intervienen en el desarrollo del proyecto.

*Fuente:* Elaborado por el tesista

## 4.3. Sprint 0: Fase previa

En este apartado se contempla el inicio del proceso de desarrollo del sistema mediante la metodología SCRUM, en donde se toma como información inicial el product backlog de la tabla 3 y los roles de la tabla 4, con todo esto identificado se procede a ponerse de acuerdo con el equipo de trabajo para identificar las metas que se desean alcanzar en cada uno de los Sprint.

### 4.3.1. Declaración del producto

La finalidad del proyecto, se fundamenta en el diseño y construcción de un prototipo de aplicativo móvil para la comercialización de GLP, que permita mejorar el servicio de solicitud y entrega de cilindros de gas, por medio de herramientas tecnologías que busquen desarrollar una solución tecnológica eficiente y dirigida a dispositivos móviles que facilite la solicitud de la reposición de un cilindro de una manera rápida.

### 4.3.2. Definición del terminado

Por medio de la definición del terminado se puede verificar paso a paso los requisitos que el sistema debe cumplir para afirmar que se han alcanzado los resultados de forma satisfactoria. Por lo tanto, se elaboró la siguiente ficha que permite colocar un check en los ítems a cumplir. A continuación, se procede a mostrar la ficha:

Tabla 5

*Definición del terminado*

|  |
| --- |
| Definition of Done |
| Se revisó el diseño de la base de datos.  Se revisó el diseño de la interfaz gráfica.  Se revisó el funcionamiento del sistema. |
| Código Completo  La aplicación realizo el Build sin ningún error.  El BackEnd no arroja ningún error.  Código inspeccionado  Código esta comentado. |
| Se actualizó la documentación. |
| Pruebas  Pruebas unitarias  Pruebas de integración |
| Aceptación de la App  Aceptación del BackEnd. |
| Funcional en dispositivos Android. |

*Fuente:* Elaborado por el tesista

### 4.3.3. Diseño

Para describir el funcionamiento que tendrá el sistema se decidió realizar diagramas del sistema los cuales son modelados de acuerdo a cada uno de los procesos que se realizaran en el sistema. Por consiguiente, se realizaron los siguientes diagramas:

* Diagrama de clases
* Diagrama entidad - relación
* Diagrama de caso de uso general
* Diagramas de caso de uso a detalle

***Diagrama de clases***

En base a los sprint del producto backlog se procedió a realizar el modelo de la base de datos lo cual será necesario para identificar cada una de las entidades necesarias para cubrir cada uno de los requerimientos del sistema y de esta forma la información pueda ser almacenada de forma correcta. Por medio del diagrama de clases se podrá esquematizar el modelo de programación orientada a objetos, para posteriormente sean relacionadas y permita la construcción del prototipo del sistema.

Tabla 6

*Diagrama de clases del prototipo*

|  |  |
| --- | --- |
| Clientes | |
| - id  - idUsuario  - lat  - lng  - estado | Int (11) PK  Int (11)  Double (11,6)  Double (11,6)  Varchar (100) |

|  |  |
| --- | --- |
| Inventario | |
| - id  - idProducto  - existencia  - estado | Int (11) PK  Int (11)  Int (11)  Varchar (100) |

|  |  |
| --- | --- |
| Pedidos | |
| - id  - idCliente  - idRepartidor  - fecha  - hora  - subtotal  - iva  - total  - estado  - comentario  - origenLat  - origenLng  - destinoLat  - destinoLng | Int (11) PK  Int (11)  Int (11)  Date  Time  Double (11,2)  Double (11,2)  Double (11,2)  Varchar (20)  Varchar (255)  Double (11,6)  Double (11,6)  Double (11,6)  Double (11,6) |

|  |  |
| --- | --- |
| Pedidosds | |
| - id  - idPedido  - idProducto  - cantidad | Int (11) PK  Int (11)  Int (11)  Int (11) |

|  |  |
| --- | --- |
| Personas | |
| - id  - nombres  - apellidos  - cedula  - teléfono  - dirección  - email  - estado | Int (11) PK  Varchar (255)  Varchar (255)  Varchar (20)  Varchar (20)  Varchar (255)  Varchar (100)  Varchar (50) |

|  |  |
| --- | --- |
| Productos | |
| - id  - producto  - descripción  - costo  - pvp  - imagen | Int (11) PK  Varchar (100)  Varchar (255)  Double (11,2)  Double (11,2)  text |

|  |  |
| --- | --- |
| Repartidores | |
| - id  - idUsuario  - lat  - lng  - estado | Int (11) PK  Int (11)  Double (11,6)  Double (11,6)  Varchar (100) |

|  |  |
| --- | --- |
| Roles | |
| - id  - rol  - descripción | Int (11) PK  Varchar (20)  Varchar (255) |

|  |  |
| --- | --- |
| Usuarios | |
| - id  - idPersona  - idRol  - usuario  - clave  - estado | Int (11) PK  Int (11)  Int (11)  Varchar (255)  Varchar (255)  Varchar (100) |

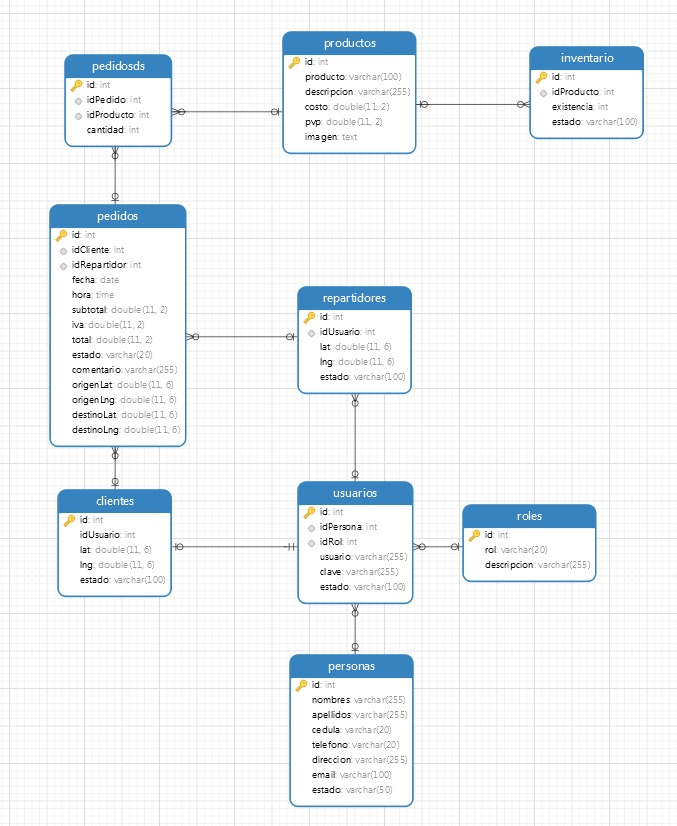
*Nota.* Entidades de la base de datos

*Fuente:* Elaborado por el tesista

***Diagrama entidad – relación***

Figura 25

*Diagrama entidad – relación del prototipo*



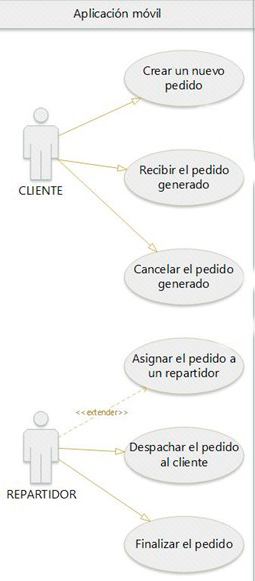
*Fuente:* Elaborado por el tesista

***Diagrama de casos de uso general***

Después de definir cada uno de los requerimientos del sistema, es necesario elaborar un caso de uso general, es decir que contenga todos los usuarios que intervienen en el sistema y los procesos y acciones que se realizan. Los usuarios que intervienen en el sistema son: Administrador, repartidor y cliente. A continuación, se muestra el diagrama elaborado tanto para la parte web como para la móvil:

Figura 26

*Caso de uso general*



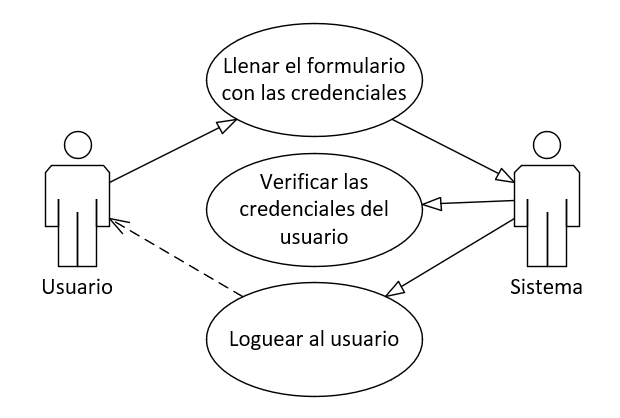
*Fuente:* Elaborado por el tesista

***Diagramas de casos de uso a detalle***

Los diagramas de casos de uso describen el comportamiento de los actores del sistema en un proceso determinado. Para ello se tomará en cuenta cada uno de los Sprint generados en la tabla 3 a continuación cada uno de los diagramas:

Figura 27

*Caso de uso del ingreso al sistema desde la app*



*Fuente:* Elaborado por el tesista

Tabla 7

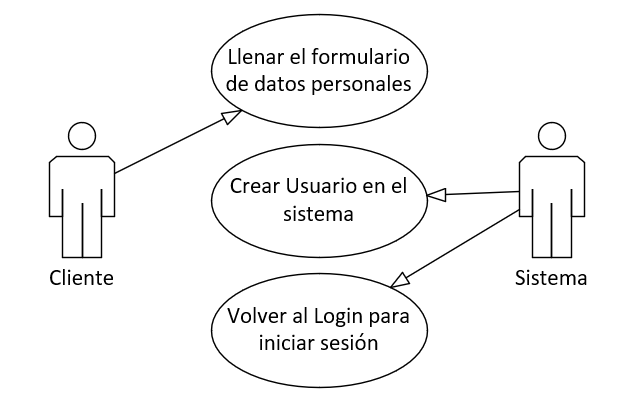
*Requerimientos para el caso de uso del ingreso al sistema*

|  |  |
| --- | --- |
| Ingreso al sistema | |
| Objetivo | Permitir a los usuarios ingresar a las funcionalidades del sistema por medio de credenciales asignadas en el registro. |
| Responsable del sistema | Cliente |
| Requisitos | 1. Nombre de Usuario 2. Password o Contraseña de seguridad |
| Procedimiento | 1. Abrir la aplicación. 2. Llenar el formulario de login con las credenciales personales. 3. Enviar formulario |
| Éxito | El usuario podrá acceder al sistema con normalidad |
| Comentarios | Es necesario que el usuario tenga conexión a internet |

*Fuente:* Elaborado por le tesista

Figura 28

*Caso de uso del registro de clientes desde la app*



*Fuente:* Elaborado por el tesista

Tabla 8

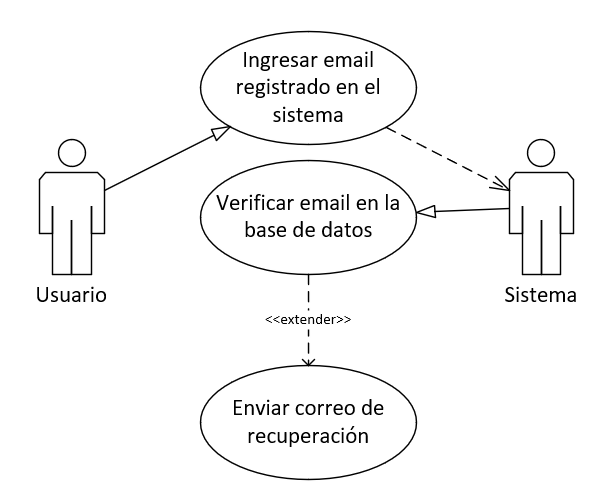
*Requerimientos para el caso de uso del registro de usuario*

|  |  |
| --- | --- |
| Registro en la app | |
| Objetivo | Permitir a los clientes registrarse por medio de la aplicación para poder ingresar al sistema. |
| Responsable del sistema | Cliente |
| Requisitos | 1. Nombres 2. Apellidos 3. Cédula de identidad 4. Teléfono o celular 5. Dirección 6. Email 7. Password o Contraseña |
| Procedimiento | 1. Abrir la aplicación. 2. Dirigirse a la opción de registrarse. 3. Llenar el formulario de información personal. 4. Enviar el formulario. |
| Éxito | El usuario cliente se podrá registrar para poder acceder al sistema. |
| Comentarios | Es necesario que el usuario tenga conexión a internet, ingrese una cédula y email válidos. |

*Fuente:* Elaborado por le tesista

Figura 29

*Caso de uso de la recuperación de contraseña desde la app*



*Fuente:* Elaborado por le tesista

Tabla 9

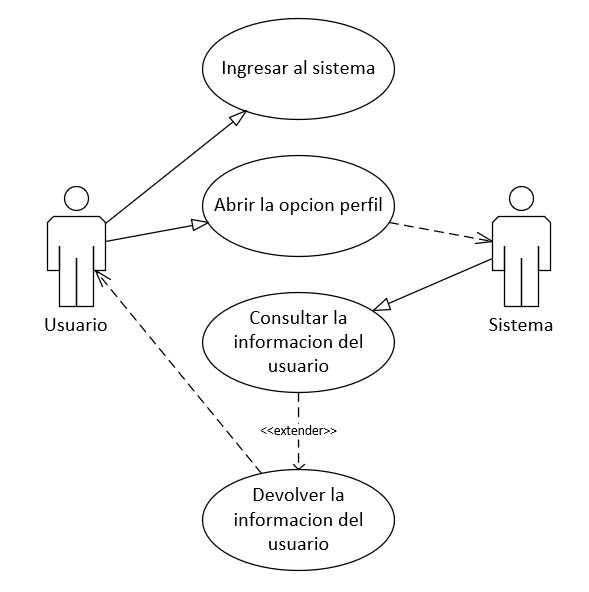
*Requerimientos para el caso de uso de la recuperación de contraseña*

|  |  |
| --- | --- |
| Recuperación de contraseña en la app | |
| Objetivo | Permitir a los clientes registrados poder recuperar su contraseña por medio de verificación de email. |
| Responsable del sistema | Cliente |
| Requisitos | 1. Email |
| Procedimiento | 1. Abrir la aplicación. 2. Dirigirse a la opción de “*Olvide mi contraseña*”. 3. Ingresar el email del usuario. 4. Enviar el formulario. 5. Verificación del email. 6. Envió de correo de recuperación de contraseña. |
| Éxito | El usuario recibirá un email con la nueva contraseña de recuperación. |
| Comentarios | Es necesario que el usuario ingrese el email con el que se registró en la aplicación. |

*Fuente:* Elaborado por le tesista

Figura 29

*Caso de uso de la opción perfil en la app*



*Fuente:* Elaborado por le tesista

Tabla 10

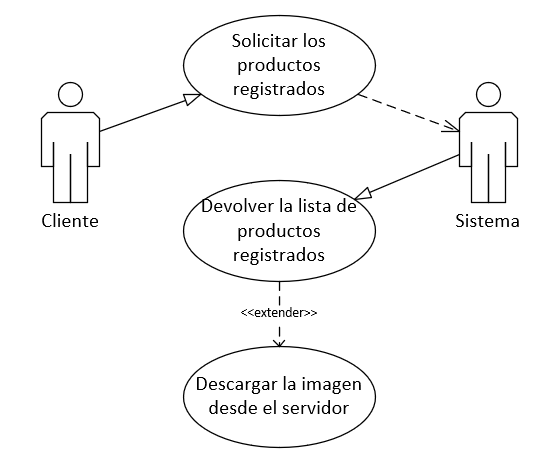
*Requerimientos para el caso de uso de la opción perfil*

|  |  |
| --- | --- |
| Perfil de Usuario en la app | |
| Objetivo | Permitir a los usuarios registrados en el sistema acceder a su perfil de usuario. |
| Responsable del sistema | Cliente |
| Requisitos | 1. Estar registrado en el sistema 2. Tener acceso a la app o a la plataforma web |
| Procedimiento | 1. Abrir la plataforma web o la app. 2. Loguearse con las credenciales personales. 3. Acceder a la opción Perfil. |
| Éxito | El usuario podrá visualizar toda su información personal, además de cambiar su contraseña de acceso al sistema. |
| Comentarios | Es necesario que el usuario este registrado previamente en el sistema. |

*Fuente:* Elaborado por le tesista

Figura 30

*Caso de uso de la lista de productos en la app*



*Fuente:* Elaborado por le tesista

Tabla 11

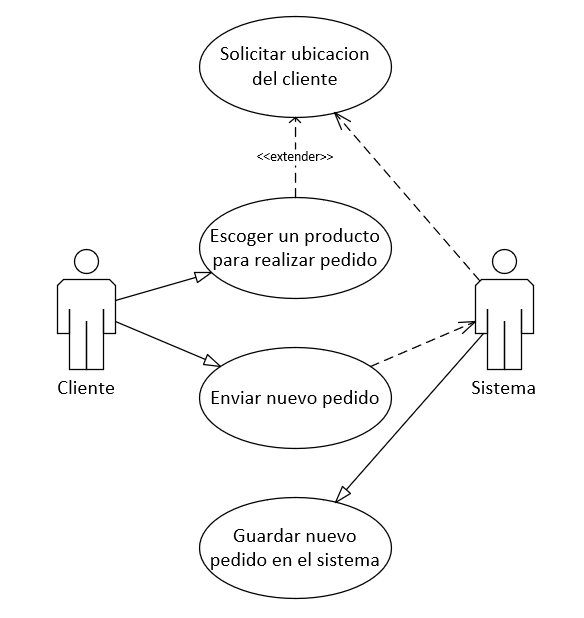
*Requerimientos para el caso de uso de listar productos en la app*

|  |  |
| --- | --- |
| Listar productos en la app | |
| Objetivo | Permitir al cliente ingresar a la lista de productos que la empresa oferta en su aplicación móvil. |
| Responsable del sistema | Cliente |
| Requisitos | 1. Estar registrado en el sistema 2. Conexión a internet |
| Procedimiento | 1. Abrir la aplicación móvil. 2. Loguearse con las credenciales personales. 3. Acceder al menú Productos. 4. Visualizar toda la lista de productos |
| Éxito | El cliente podrá visualizar el producto y sus características. |
| Comentarios | El cliente debe estar registrado en el sistema |

*Fuente:* Elaborado por le tesista

Figura 32

*Caso de uso de realizar un nuevo pedido*



*Fuente:* Elaborado por le tesista

Tabla 12

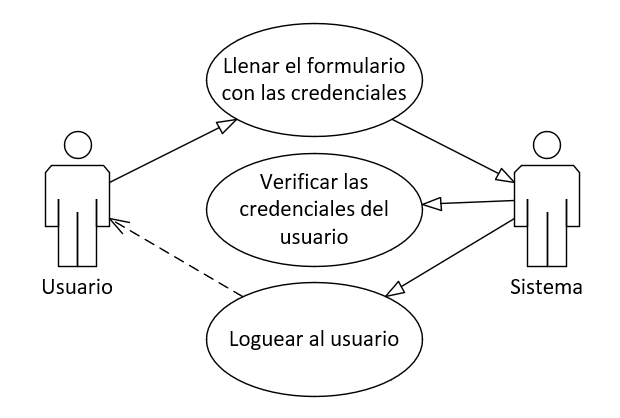
*Requerimientos para el caso de uso de realizar un nuevo pedido*

|  |  |
| --- | --- |
| Nuevo pedido en la app | |
| Objetivo | Permitir al cliente registrar un nuevo pedido de un producto seleccionado, mismo que se almacenara en el sistema. |
| Responsable del sistema | Cliente |
| Requisitos | 1. Estar registrado en el sistema 2. Dar permiso de uso del GPS |
| Procedimiento | 1. Abrir la aplicación móvil. 2. Acceder al menú Pedidos. 3. Seleccionar un producto de la lista de productos 4. Especificar la cantidad del producto 5. Escoger la ubicación a donde se realizará él envió del producto 6. Enviar pedido a guardar |
| Éxito | El cliente podrá visualizar su pedido en el historial de pedidos con estado generado. |
| Comentarios | El cliente debe estar registrado en el sistema  El cliente debe tener acceso a internet o no podrá realizar pedidos  El cliente debe asegurarse de haber dado el permiso de uso del GPS a la aplicación móvil. |

*Fuente:* Elaborado por le tesista

Figura 31

*Caso de uso del ingreso al sistema desde la app*



*Fuente:* Elaborado por el tesista

Tabla 13

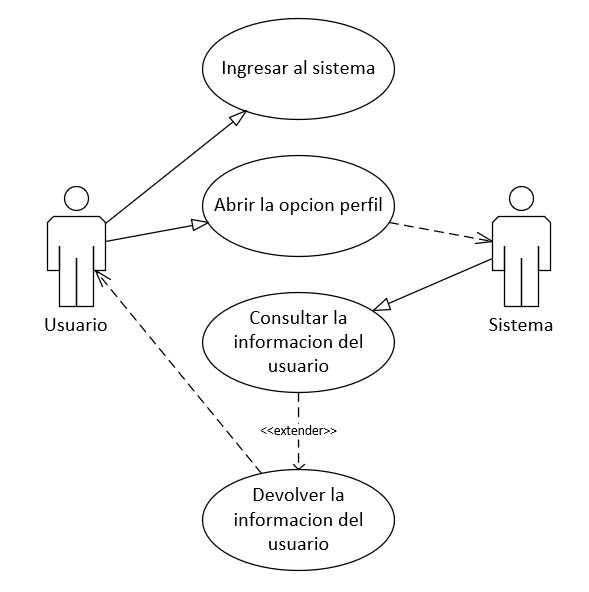
*Requerimientos para el caso de uso del ingreso al sistema*

|  |  |
| --- | --- |
| Ingreso al sistema | |
| Objetivo | Permitir a los usuarios ingresar a las funcionalidades del sistema por medio de credenciales asignadas en el registro. |
| Responsable del sistema | Repartidor |
| Requisitos | 1. Nombre de Usuario 2. Password o Contraseña de seguridad |
| Procedimiento | 1. Abrir la aplicación 2. Llenar el formulario de login con las credenciales personales. 3. Enviar formulario |
| Éxito | El usuario podrá acceder al sistema con normalidad |
| Comentarios | Es necesario que el usuario tenga conexión a internet |

*Fuente:* Elaborado por le tesista

Figura 32

*Caso de uso de la opción perfil en la app*



*Fuente:* Elaborado por le tesista

Tabla 14

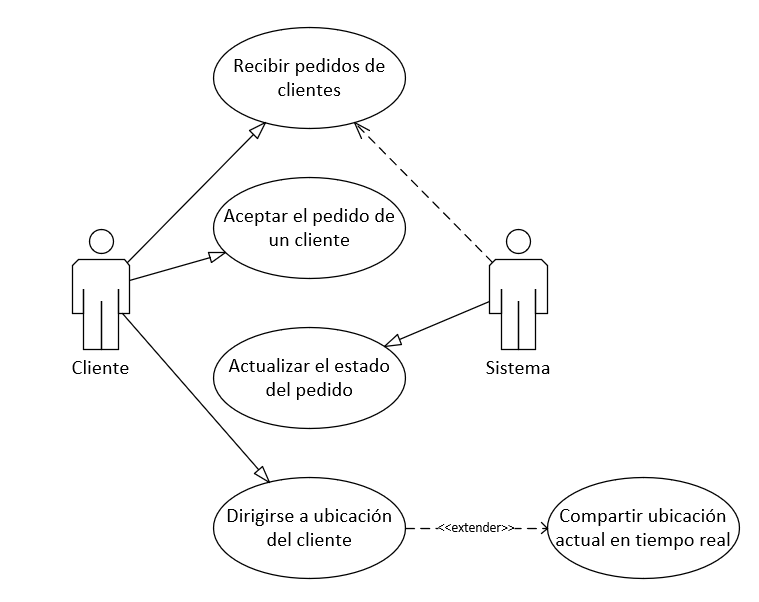
*Requerimientos para el caso de uso de la opción perfil*

|  |  |
| --- | --- |
| Perfil de Usuario en la app | |
| Objetivo | Permitir a los usuarios registrados en el sistema acceder a su perfil de usuario. |
| Responsable del sistema | Repartidor |
| Requisitos | 1. Estar registrado en el sistema 2. Tener acceso a la app o a la plataforma web |
| Procedimiento | 1. Abrir la app. 2. Loguearse con las credenciales personales. 3. Acceder a la opción Perfil. |
| Éxito | El usuario podrá visualizar toda su información personal, además de cambiar su contraseña de acceso al sistema. |
| Comentarios | Es necesario que el usuario este registrado previamente en el sistema. |

*Fuente:* Elaborado por le tesista

Figura 33

*Caso de uso de aceptar un pedido en la app*



*Fuente:* Elaborado por le tesista

Tabla 15

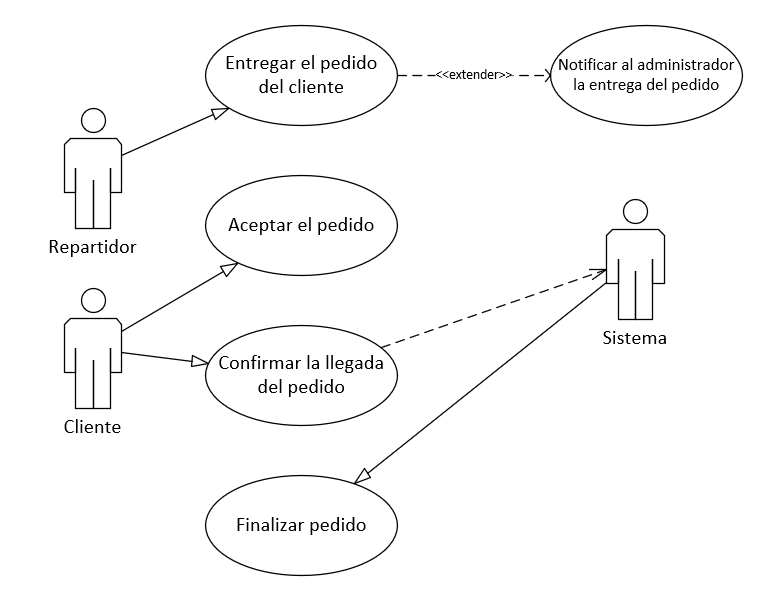
*Requerimientos para el caso de uso de aceptar un pedido*

|  |  |
| --- | --- |
| Aceptar pedido en la app | |
| Objetivo | Permitir al sistema asignar al repartidor más cercano a la ubicación del cliente. |
| Responsable del sistema | Repartidor |
| Requisitos | 1. Ser usuario repartidor |
| Procedimiento | 1. Abrir la app. 2. Loguearse con las credenciales personales. 3. Acceder a la opción Pedidos asignados. 4. Visualizar la ubicación del cliente para la entrega |
| Éxito | El repartidor recibirá un mensaje de que el pedido ha sido aceptado exitosamente. |
| Comentarios | El repartidor debe permitir el uso del GPS por parte de la app |

*Fuente:* Elaborado por le tesista

Figura 34

*Caso de uso de finalizar un pedido en la app*



*Fuente:* Elaborado por le tesista

Tabla 16

*Requerimientos para el caso de uso de finalizar un pedido*

|  |  |
| --- | --- |
| Finalizar pedido en la app | |
| Objetivo | Permitir al cliente finalizar un pedido una vez que el usuario repartidor haya realizado la entrega en la ubicación del cliente. |
| Responsable del sistema | Repartidor |
| Requisitos | 1. Ser usuario repartidor 2. Ser usuario cliente registrado en el sistema |
| Procedimiento | 1. El usuario repartidor abre la ubicación del cliente. 2. Dirigirse a la ubicación del cliente. 3. Enviar notificación de llegada a la ubicación. 4. Entregar el pedido 5. El usuario cliente debe confirmar la llegada del pedido 6. Enviar solicitud de finalizar el pedido |
| Éxito | El cliente recibe la notificación que el pedido ha sido finalizado. |
| Comentarios | El repartidor debe compartir su ubicación real  El cliente debe estar logueado en la app |

*Fuente:* Elaborado por le tesista

### 4.3.4. Prototipado

Conforme al diseño de todo el sistema informático que se detalló, se procede a realizar el prototipo de toda la interfaz de usuario, para lo cual se hará uso de mockups los cuales servirán como vista previa de la interfaz del software desarrollado en el presente proyecto.

***Login***

Figura 35

*Mockup del prototipo de pantalla del login de la app*

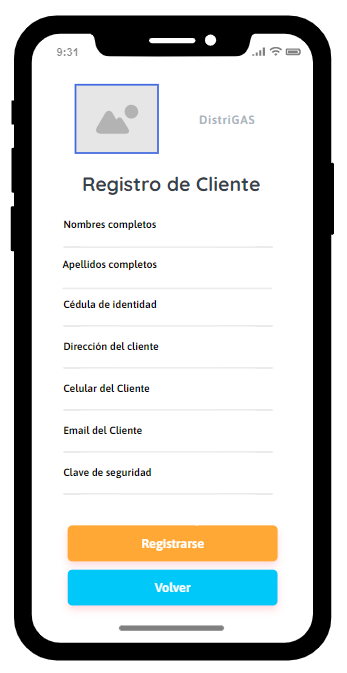


*Fuente:* Elaborado por el tesista

***Registro***

Figura 36

*Mockup del prototipo de pantalla del registro en la app*



*Fuente:* Elaborado por el tesista

***Recuperación de contraseña***

Figura 37

*Mockup del prototipo de pantalla de recuperación de contraseña en la app*

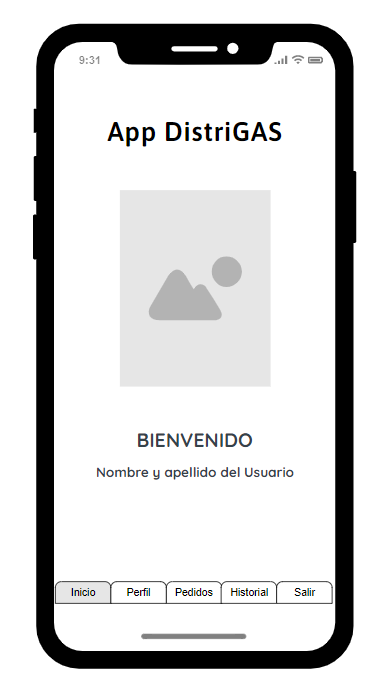
****

*Fuente:* Elaborado por el tesista

***Menú Principal***

Figura 38

*Mockup del prototipo de pantalla del menú principal en la app*

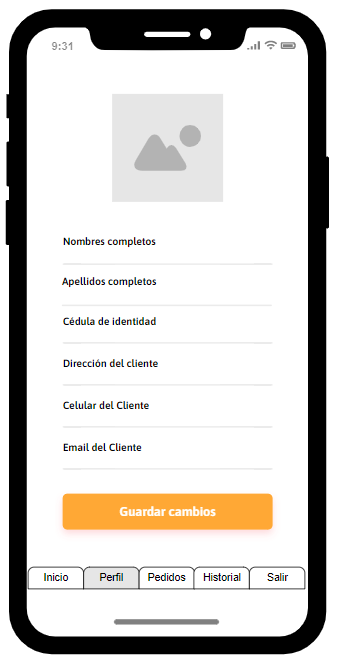


*Fuente:* Elaborado por el tesista

***Perfil del Usuario***

Figura 39

*Mockup del prototipo de pantalla del perfil del usuario en la app*



*Fuente:* Elaborado por el tesista

**Pedidos**

Figura 40

*Mockup del prototipo de pantalla de pedidos en la app*

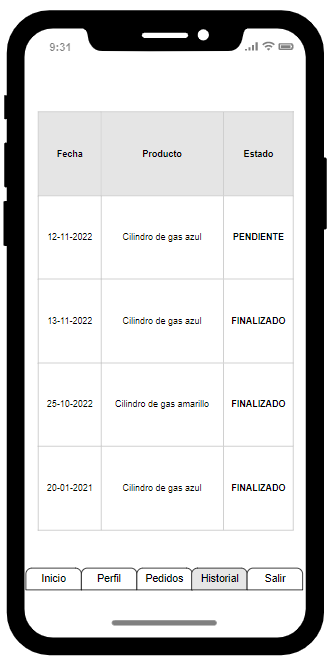


*Fuente:* Elaborado por el tesista

***Historial de pedidos***

Figura 41

*Mockup del prototipo de pantalla del historial de pedidos en la app*



*Fuente:* Elaborado por el tesista

## 4.4. Sprint 1: Módulo de seguridad, login, registro y recuperación de contraseña

Este Sprint estuvo planificado para 3 semanas aproximadamente, lo cual empezó con la creación de las interfaces graficas para la parte móvil en base a los diseños presentados en el Sprint 0 del capítulo anterior. Además, se implementó la seguridad en el Login, el registro del cliente en la parte de la aplicación móvil y las prácticas de seguridad necesarias para la encriptación de las contraseñas y en él envió controlado de correos de recuperación de contraseña disponible en el aplicativo.

Tabla 17

*Sprint Backlog 1*

|  |  |  |  |
| --- | --- | --- | --- |
| Sprint Backlog 1 | | | |
| Tarea ID | **Historia** | **Estimado**  **(Días)** | **Prioridad** |
| 1 | El usuario puede ingresar al sistema únicamente con credenciales. | 8 | Alta |
| 2 | El usuario puede registrarse desde el login. | 8 | Media |
| 3 | El usuario puede recuperar su contraseña. | 5 | Media |

*Fuente.* Elaborado por el tesista

Tabla 18

*Actividades del Sprint 1*

|  |  |
| --- | --- |
| Actividades | Estimado/ (horas) |
| Diseño de la interfaz principal de la aplicación móvil | 8 |
| Diseño de la interfaz del login en la aplicación móvil | 10 |
| Diseño del formulario para el registro de nuevos clientes en la aplicación móvil | 15 |
| Diseño del formulario para la recuperación de contraseña en la aplicación móvil | 10 |
| Codificación del login en la aplicación móvil para el ingreso de los clientes y repartidores | 20 |
| Codificación del registro de clientes en la aplicación móvil para la creación de nuevos registros | 15 |
| Codificación de la recuperación de contraseña en la aplicación móvil, para clientes que olvidaron su clave para el acceso al sistema | 20 |
| Implementación de lo desarrollado en la aplicación móvil | 5 |
| TOTAL (horas trabajadas) | **124** |

*Fuente.* Elaborado por el tesista

Figura 42

*Codificación para el login en el Sprint 1*

login(req, res) {

        const usuario = req.body.usuario;

        const clave = req.body.clave;

        Usuario.FindByUsername(usuario, async (err, data) => {

            if (err) {

                return res.status(501).json({

                    success: false,

                    mensaje: "hubo un error",

                    error: err

                })

            }

            if (!data) {

                return res.status(401).json({

                    success: false,

                    mensaje: "El Usuario no fue encontrado",

                })

            }

            const isPasswordValid = await bcrypt.compare(clave, data.clave)

            if (isPasswordValid) {

                const token = Jwt.sign({

                    id: data.id,

                    usuario: data.usuario

                }, keys.secretOrKey, {

                })

                const dataJwt = {

                    idUsuario: data.idUsuario,

                    usuario: data.usuario,

                    estado: data.estado,

                    nombres: data.nombres,

                    apellidos: data.apellidos,

                    cedula: data.cedula,

                    telefono: data.telefono,

                    direccion: data.direccion,

                    email: data.email,

                    rol: data.rol,

                    idPersona: data.idPersona,

                    idRol: data.idRol,

                    session\_token: 'JWT ' + token

                    //todo lo que vaya a devolver

                }

                return res.status(201).json({

                    success: true,

                    mensaje: "El Usuario fue autenticado con exito",

                    data: dataJwt

                })

            } else {

                return res.status(401).json({

                    success: false,

                    mensaje: "El Usuario o la contraseña son incorrectas"

                })

            }

        })

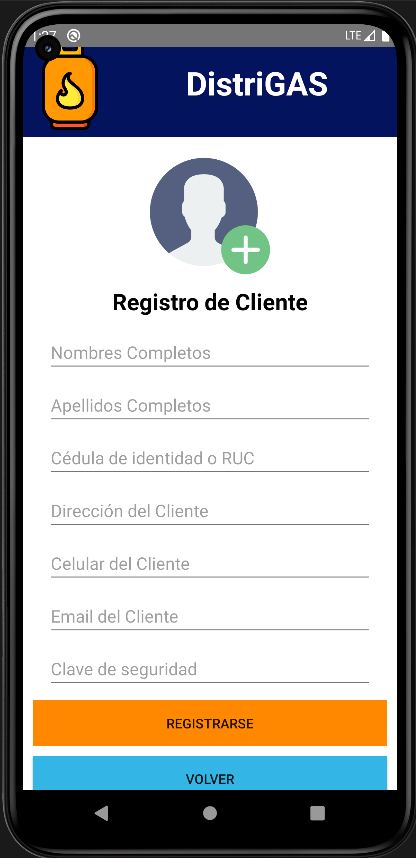
    }

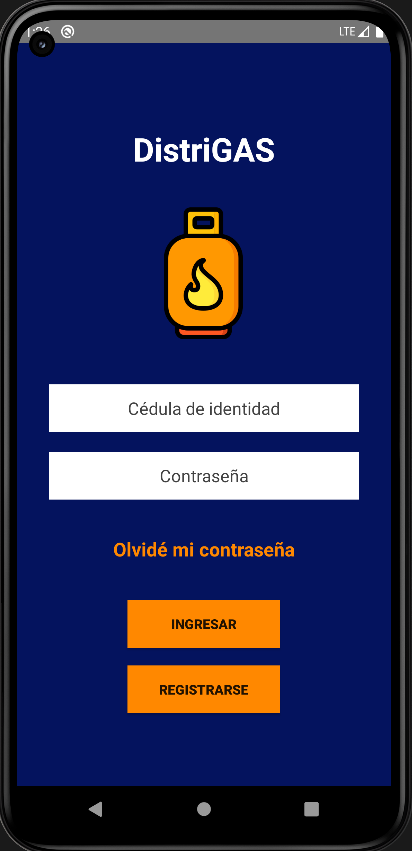
*private void* iniciarSesion(String usuario, String clave){  
 RequestQueue queue = Volley.*newRequestQueue*(*this*);  
 String url = Api.***API\_RUTA*** + "login";  
 StringRequest stringRequest = *new* StringRequest(Request.*Method*.***POST***, url,  
 *new* Response.Listener<String>() {  
 @Override  
 *public void* onResponse(String response) {  
 *try* {  
 JSONObject data = *new* JSONObject(response);  
 JSONObject usuarioBD = *new* JSONObject(String.*valueOf*(data.getJSONObject("data")));  
 *if*(!usuarioBD.getString("rol").toString().equals("Administrador")){  
 Toast.*makeText*(getApplicationContext(), data.getString("mensaje"),Toast.***LENGTH\_LONG***).show();  
 }  
 guardarSesion(usuarioBD.getString("cedula"), clave, usuarioBD.getString("nombres"), usuarioBD.getString("apellidos"), usuarioBD.getString("direccion"),usuarioBD.getString("telefono"),usuarioBD.getString("email"), usuario, usuarioBD.getString("idUsuario"), usuarioBD.getString("rol"));  
 } *catch* (JSONException e) {  
 e.printStackTrace();  
 }  
 }  
 }, *new* Response.ErrorListener() {  
 @Override  
 *public void* onErrorResponse(VolleyError error) {  
 *if*(error.networkResponse.statusCode == 401) {  
 String responseBody = *null*;  
 *try* {  
 responseBody = *new* String(error.networkResponse.data, "utf-8");  
 JSONObject data = *new* JSONObject(responseBody);  
 String message = data.optString("mensaje");  
 Toast.*makeText*(getApplicationContext(), message,Toast.***LENGTH\_LONG***).show();  
 } *catch* (UnsupportedEncodingException e) {  
 e.printStackTrace();  
 } *catch* (JSONException e) {  
 e.printStackTrace();  
 }  
 }  
 }  
 })  
 {  
 *protected Map*<String, String> getParams() {  
 *Map*<String, String> params = *new* HashMap<>();  
 params.put("usuario", usuario);  
 params.put("clave", clave);  
 *return* params;  
 }  
 };  
 VolleySingleton.*getInstance*(getApplicationContext()).addToRequestQueue(stringRequest);  
 Limpiar();  
}

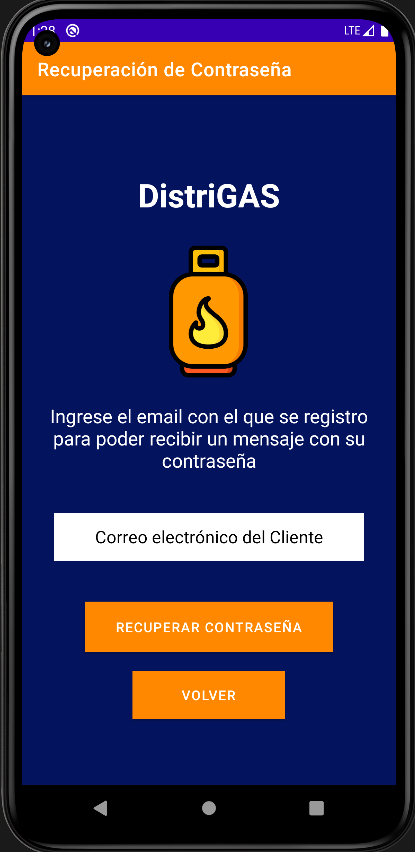
*Fuente.* Prototipo de DistriGAS

A continuación, se mostrarán algunas capturas de pantalla de lo desarrollado en el sprint 1:

Figura 43

*Capturas de pantalla del Sprint 1*





*Fuente.* Prototipo de DistriGAS

## 4.5. Sprint 2: Módulo de Usuarios

Este Sprint estuvo planificado para 2 semanas aproximadamente, el cual consiste en la creación de nuevos usuarios de tipo repartidor y clientes, mismos que permitirán la interacción con la aplicación móvil. Además, los mismos pueden ser eliminados desde la interfaz del administrador. Finalmente se implementó los usuarios y clientes en el aplicativo para que se pueda visualizar los usuarios y clientes registrados y también poder eliminarlos.

Tabla 19

*Sprint Backlog 2*

|  |  |  |  |
| --- | --- | --- | --- |
| Sprint Backlog 2 | | | |
| Tarea ID | **Historia** | **Estimado**  **(Días)** | **Prioridad** |
| 4 | Como usuario puedo crear nuevos usuarios. | 3 | Alta |
| 5 | Como usuario se puede eliminar los usuarios. | 3 | Media |
| 6 | Como usuario se puede crear nuevos clientes. | 2 | Media |
| 7 | Como usuario se puede y eliminar los clientes. | 5 | Media |

*Fuente.* Elaborado por el tesista

Tabla 20

*Actividades del Sprint 2*

|  |  |
| --- | --- |
| Actividades | Estimado/ (horas) |
| Diseño de la interfaz de la creación de nuevos usuarios | 6 |
| Diseño de la función de eliminar clientes | 6 |
| Diseño de la interfaz de la lista de clientes | 8 |
| Diseño del botón para eliminar clientes | 4 |
| Codificación de la validación de la creación de clientes en la aplicación móvil | 6 |
| Codificación de la lista de usuarios y clientes | 6 |
| Codificación del botón para eliminar usuarios y clientes | 2 |
| Implementación de la lista de usuarios y clientes | 3 |
| Implementación de las validaciones en el registro de clientes en la aplicación móvil | 1 |
| TOTAL (horas trabajadas) | **42** |

*Fuente.* Elaborado por el tesista

Figura 44

*Codificación del registro de clientes del Sprint 2*

registro(req, res) {

        const usuario = req.body;

        Usuario.registro(usuario, (err, data) => {

            if (err) {

                return res.status(501).json({

                    success: false,

                    mensaje: "hubo un error",

                    error: err

                })

            }

            return res.status(201).json({

                success: true,

                mensaje: "El Cliente ha sido registrado exitosamente",

                data: data

            })

        })

}

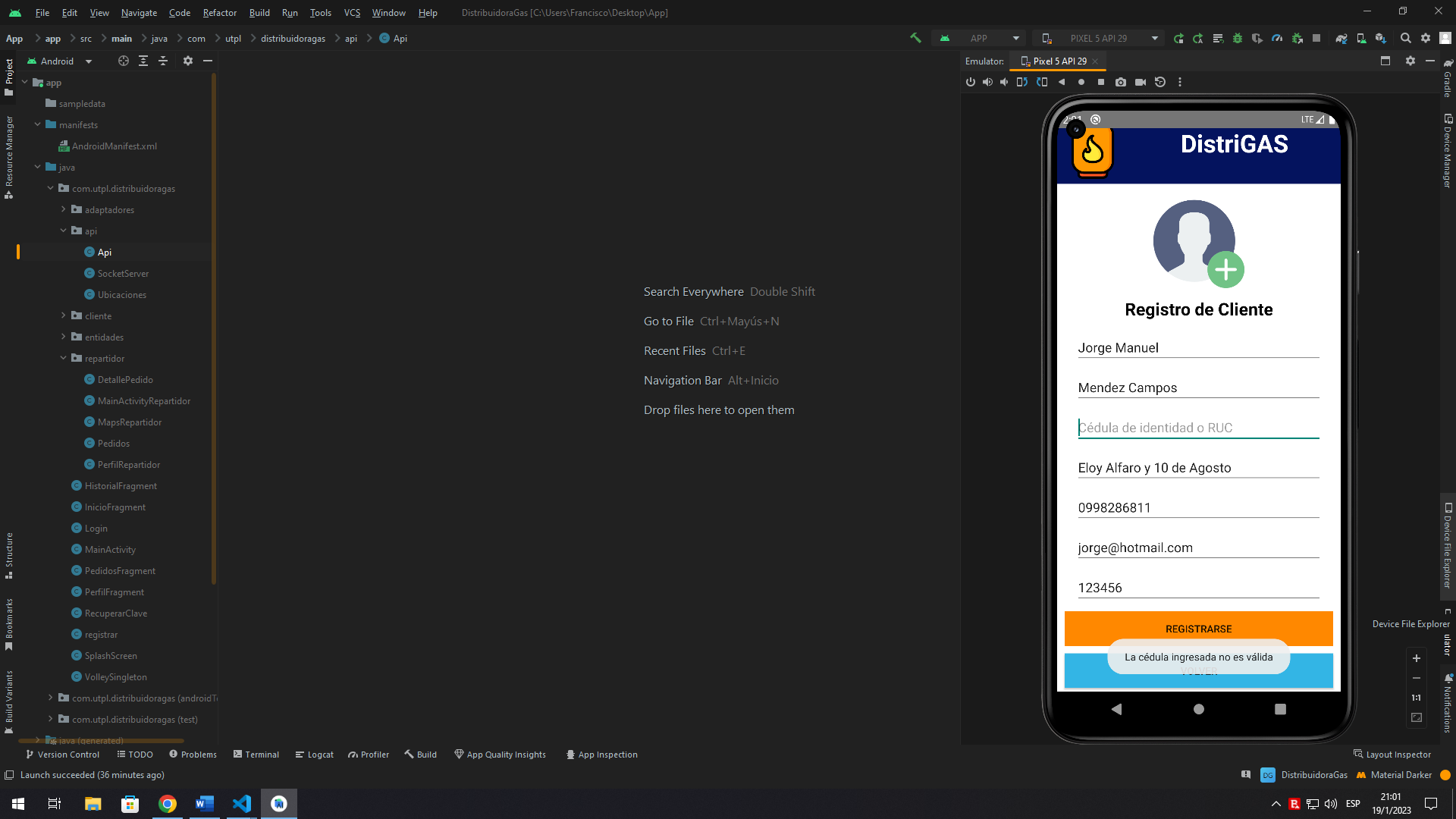
*private void* registrarse(String nombres, String apellidos, String cedula, String direccion, String celular, String email, String clave){  
 RequestQueue queue = Volley.*newRequestQueue*(*this*);  
 String url = Api.***API\_RUTA*** + "usuarios/registro";  
 StringRequest stringRequest = *new* StringRequest(Request.*Method*.***POST***, url,  
 *new* Response.Listener<String>() {  
 @Override  
 *public void* onResponse(String response) {  
 *try* {  
 JSONObject data = *new* JSONObject(response);  
 Toast.*makeText*(getApplicationContext(), data.getString("mensaje"),Toast.***LENGTH\_LONG***).show();  
 Intent intent = *new* Intent(getApplicationContext(), Login.*class*);  
 startActivity(intent);  
 } *catch* (JSONException e) {  
 e.printStackTrace();  
 }  
 }  
 }, *new* Response.ErrorListener() {  
 @Override  
 *public void* onErrorResponse(VolleyError error) {  
 *if*(error.networkResponse.statusCode == 401) {  
 String responseBody = *null*;  
 *try* {  
 responseBody = *new* String(error.networkResponse.data, "utf-8");  
 JSONObject data = *new* JSONObject(responseBody);  
 String message = data.optString("mensaje");  
 Toast.*makeText*(getApplicationContext(), message,Toast.***LENGTH\_LONG***).show();  
 } *catch* (UnsupportedEncodingException e) {  
 e.printStackTrace();  
 } *catch* (JSONException e) {  
 e.printStackTrace();  
 }  
 }  
 }  
 })  
 {  
 *protected Map*<String, String> getParams() {  
 *Map*<String, String> params = *new* HashMap<>();  
 params.put("nombres", nombres);  
 params.put("apellidos", apellidos);  
 params.put("cedula", cedula);  
 params.put("direccion", direccion);  
 params.put("telefono", celular);  
 params.put("email", email);  
 params.put("usuario", cedula);  
 params.put("clave", clave);  
 params.put("rol", "Cliente");  
 *return* params;  
 }  
 };  
 VolleySingleton.*getInstance*(getApplicationContext()).addToRequestQueue(stringRequest);  
 Limpiar();  
}

*Fuente.* Prototipo de DistriGAS

A continuación, se mostrarán algunas capturas de pantalla de lo desarrollado en el sprint 2:

Figura 45

*Capturas de pantalla del Sprint 2*



*Fuente.* Prototipo de DistriGAS

## 4.7. Sprint 3: Módulo de productos

Este Sprint estuvo planificado para 3 semanas aproximadamente, lo cual empezó con la creación de las interfaces graficas en la parte web para la creación de nuevos productos, la edición y eliminación de los mismos. Mientras que en la parte de la aplicación móvil se procedió a la implementación de la visualización de productos para posteriormente se puedan realizar los pedidos.

Tabla 21

*Sprint Backlog 3*

|  |  |  |  |
| --- | --- | --- | --- |
| Sprint Backlog 3 | | | |
| Tarea ID | **Historia** | **Estimado**  **(Días)** | **Prioridad** |
| 8 | El usuario puede visualizar el inventario de productos. | 8 | Alta |

*Fuente.* Elaborado por el tesista

Tabla 22

*Actividades del Sprint 3*

|  |  |
| --- | --- |
| Actividades | Estimado/ (horas) |
| Diseño de la visualización de productos en la aplicación móvil | 25 |
| Codificación de la visualización de productos en la aplicación móvil | 34 |
| Implementación de la visualización de productos en la aplicación móvil | 18 |
| TOTAL (horas trabajadas) | **77** |

*Fuente.* Elaborado por el tesista

Figura 46

*Codificación de los productos en el Sprint 3*

registro(req, res) {

        const producto = req.body;

        console.log("en productosController:"+JSON.stringify(req.body))

        Producto.registro(producto, (err, data) => {

            if (err) {

                return res.status(501).json({

                    success: false,

                    mensaje: "Hubo un error",

                    error: err

                })

            }

            return res.status(201).json({

                success: true,

                mensaje: "El Producto ha sido registrado exitosamente",

                data: data

            })

        })

    },

    listaProductos(req,res){

        Producto.listaProductos(req.body,(err, data) => {

            if (err) {

                return res.status(501).json({

                    success: false,

                    mensaje: "Hubo un error",

                    error: err

                })

            }

            return res.status(201).json({

                success: true,

                data: data

            })

        })

    },

    modificarProducto(req,res){

        const producto = req.body;

        Producto.modificarProducto(producto, (err, data) => {

            if (err) {

                return res.status(501).json({

                    success: false,

                    mensaje: "hubo un error",

                    error: err

                })

            }

            return res.status(201).json({

                success: true,

                mensaje: "Actualizacion realizada correctamente",

                data: data

            })

        })

    },

    eliProducto(req,res){

        const producto = req.body;

        console.log(producto.id)

        Producto.eliProducto(producto, (err, data) => {

            if (err) {

                return res.status(501).json({

                    success: false,

                    mensaje: "hubo un error",

                    error: err

                })

            }

            return res.status(201).json({

                success: true,

                mensaje: "Eliminacion correcta",

                data: data

            })

        })

    }

*private void* cargarProductos() {  
 dialog = *new* ProgressDialog(getContext());  
 dialog.setMessage("Cargando Productos");  
 dialog.show();  
 String url = Api.***API\_RUTA*** + "productos/listaProductos";  
 RequestQueue queue = Volley.*newRequestQueue*(getContext());  
 StringRequest stringRequest = *new* StringRequest(Request.*Method*.***GET***, url,  
 *new* Response.Listener<String>() {  
 @Override  
 *public void* onResponse(String response) {  
 Producto producto = *null*;  
 *try* {  
 JSONObject data = *new* JSONObject(response);  
 JSONArray json = data.optJSONArray("data");  
 *for*(*int* x=0; x<json.length();x++){  
 producto = *new* Producto();  
 JSONObject jsonObject = *null*;  
 jsonObject = json.getJSONObject(x);  
 producto.setId(jsonObject.optInt("id"));  
 producto.setProducto(jsonObject.optString("producto"));  
 producto.setDescripcion(jsonObject.optString("descripcion"));  
 producto.setCosto(jsonObject.optString("costo"));  
 String precioPublico = String.*format*( "%.2f", jsonObject.optDouble("pvp"));  
 producto.setPvp("$ " + precioPublico);  
 producto.setImagen(Api.***API\_RUTA\_IMG*** + jsonObject.optString("imagen"));  
 listaProductos.add(producto);  
 }  
 dialog.hide();  
 AdaptadorProductos adaptadorProductos = *new* AdaptadorProductos(listaProductos, getContext());  
 adaptadorProductos.setOnClickListener(*new* View.OnClickListener() {  
 @Override  
 *public void* onClick(View view) {  
 Intent intent = *new* Intent(getContext(), GenerarPedido.*class*);  
 intent.putExtra("idProducto", listaProductos.get(recyclerProductos.getChildAdapterPosition(view)).getId());  
 intent.putExtra("nombre", listaProductos.get(recyclerProductos.getChildAdapterPosition(view)).getProducto());  
 intent.putExtra("descripcion", listaProductos.get(recyclerProductos.getChildAdapterPosition(view)).getDescripcion());  
 intent.putExtra("pvp", listaProductos.get(recyclerProductos.getChildAdapterPosition(view)).getPvp());  
 intent.putExtra("imagen", listaProductos.get(recyclerProductos.getChildAdapterPosition(view)).getImagen());  
 startActivity(intent);  
 }  
 });  
 recyclerProductos.setAdapter(adaptadorProductos);  
 } *catch* (JSONException e) {  
 e.printStackTrace();  
 }  
 }  
 }, *new* Response.ErrorListener() {  
 @Override  
 *public void* onErrorResponse(VolleyError error) {  
 *if*(error.networkResponse.statusCode == 401) {  
 String responseBody = *null*;  
 *try* {  
 responseBody = *new* String(error.networkResponse.data, "utf-8");  
 JSONObject data = *new* JSONObject(responseBody);  
 String message = data.optString("mensaje");  
 Toast.*makeText*(getContext(), message,Toast.***LENGTH\_LONG***).show();  
 } *catch* (UnsupportedEncodingException e) {  
 e.printStackTrace();  
 } *catch* (JSONException e) {  
 e.printStackTrace();  
 }  
 }  
 }  
 });  
 VolleySingleton.*getInstance*(getContext()).addToRequestQueue(stringRequest);  
 Limpiar();  
}

*Fuente.* Prototipo de DistriGAS

## 4.8. Sprint 4: Módulo de pedidos, generar un nuevo pedido

Este Sprint estuvo planificado para 2 semanas aproximadamente, lo cual empezó con la creación de las interfaces graficas en la aplicación móvil desde donde se podrán generar los nuevos pedidos por parte del cliente. Además, se procedió a la primera implementación de un mapa de Google para poder escoger la ubicación a donde se llevará el pedido solicitado por el cliente.

Tabla 23

*Sprint Backlog 4*

|  |  |  |  |
| --- | --- | --- | --- |
| Sprint Backlog 4 | | | |
| Tarea ID | **Historia** | **Estimado**  **(Días)** | **Prioridad** |
| 10 | Como usuario se puede realizar pedidos de productos. | 12 | Alta |

*Fuente.* Elaborado por el tesista

Tabla 24

*Actividades del Sprint 4*

|  |  |
| --- | --- |
| Actividades | Estimado/ (horas) |
| Diseño de la interfaz del formulario para nuevo pedido en la aplicación móvil | 18 |
| Diseño de la interfaz para el mapa de la ubicación del Cliente en la aplicación móvil | 15 |
| Diseño de las notificaciones usadas en la aplicación móvil | 10 |
| Codificación del formulario para nuevo pedido | 8 |
| Codificación de las funciones de geolocalización para la obtención de la posición GPS en la aplicación móvil | 6 |
| Codificación del mapa de Google para la visualización de marcadores en la aplicación móvil | 8 |
| Codificación de las notificaciones en la aplicación móvil | 12 |
| Implementación del nuevo pedido en la aplicación móvil | 8 |
| Implementación del mapa de Google en la aplicación móvil | 12 |
| Implementación de las notificaciones en la aplicación móvil | 12 |
| TOTAL (horas trabajadas) | **109** |

*Fuente.* Elaborado por el tesista

Figura 47

*Codificación de la creación de un nuevo pedido en el Sprint 4*

crearPedido(req, res){

        var pedido = req.body;

        Cliente.findByUsuario(pedido.idCliente, (err, data) => {

            if (err) {

                return res.status(501).json({

                    success: false,

                    mensaje: "Hubo un error",

                    error: err

                })

            }

            pedido.idCliente = data.id

            Pedido.crearPedido(pedido, (err, data) => {

                if (err) {

                    return res.status(501).json({

                        success: false,

                        mensaje: "Hubo un error",

                        error: err

                    })

                }

                return res.status(201).json({

                    success: true,

                    mensaje: "El Pedido ha sido registrado exitosamente",

                    data: data

                })

            })

        })

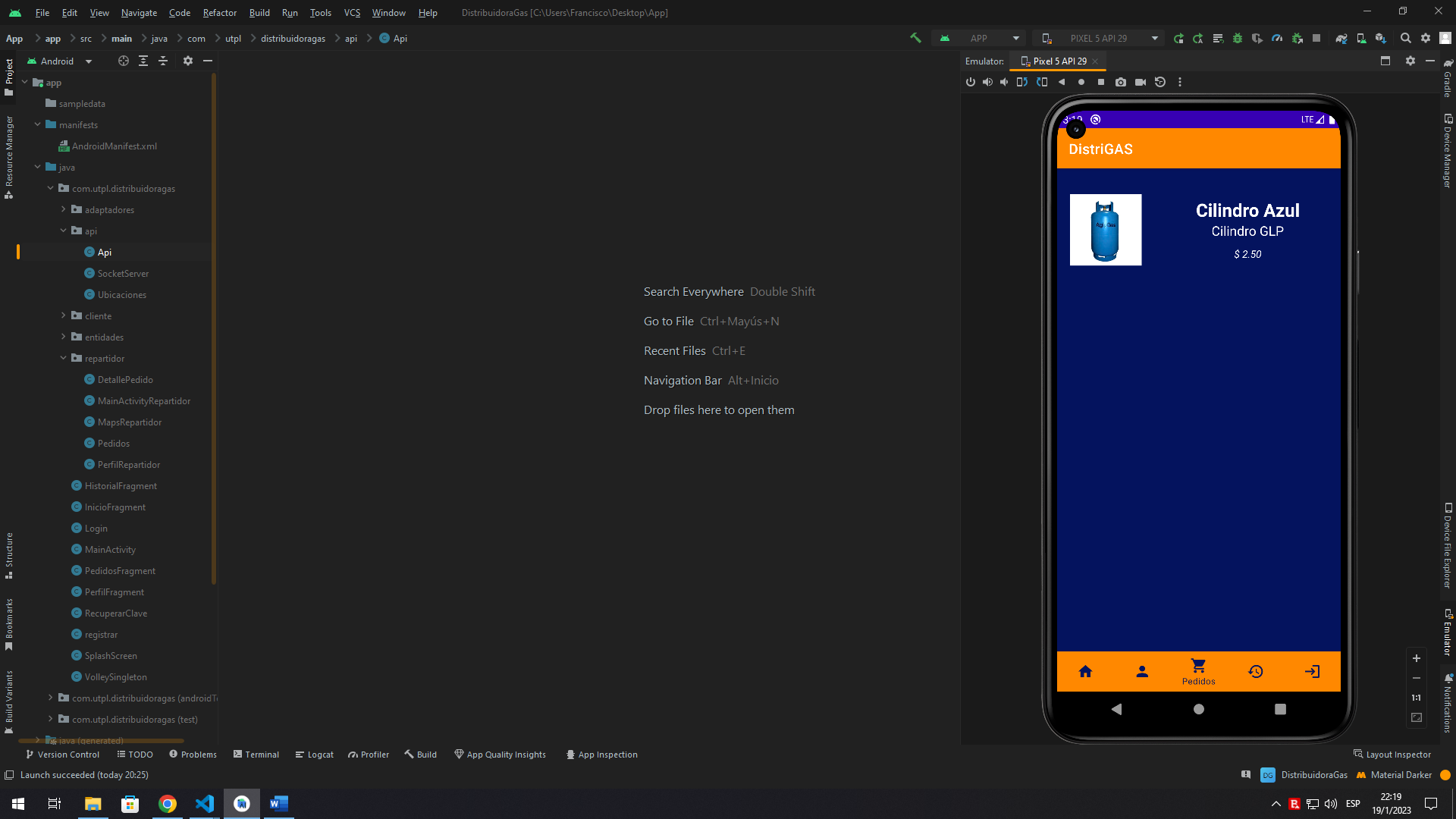
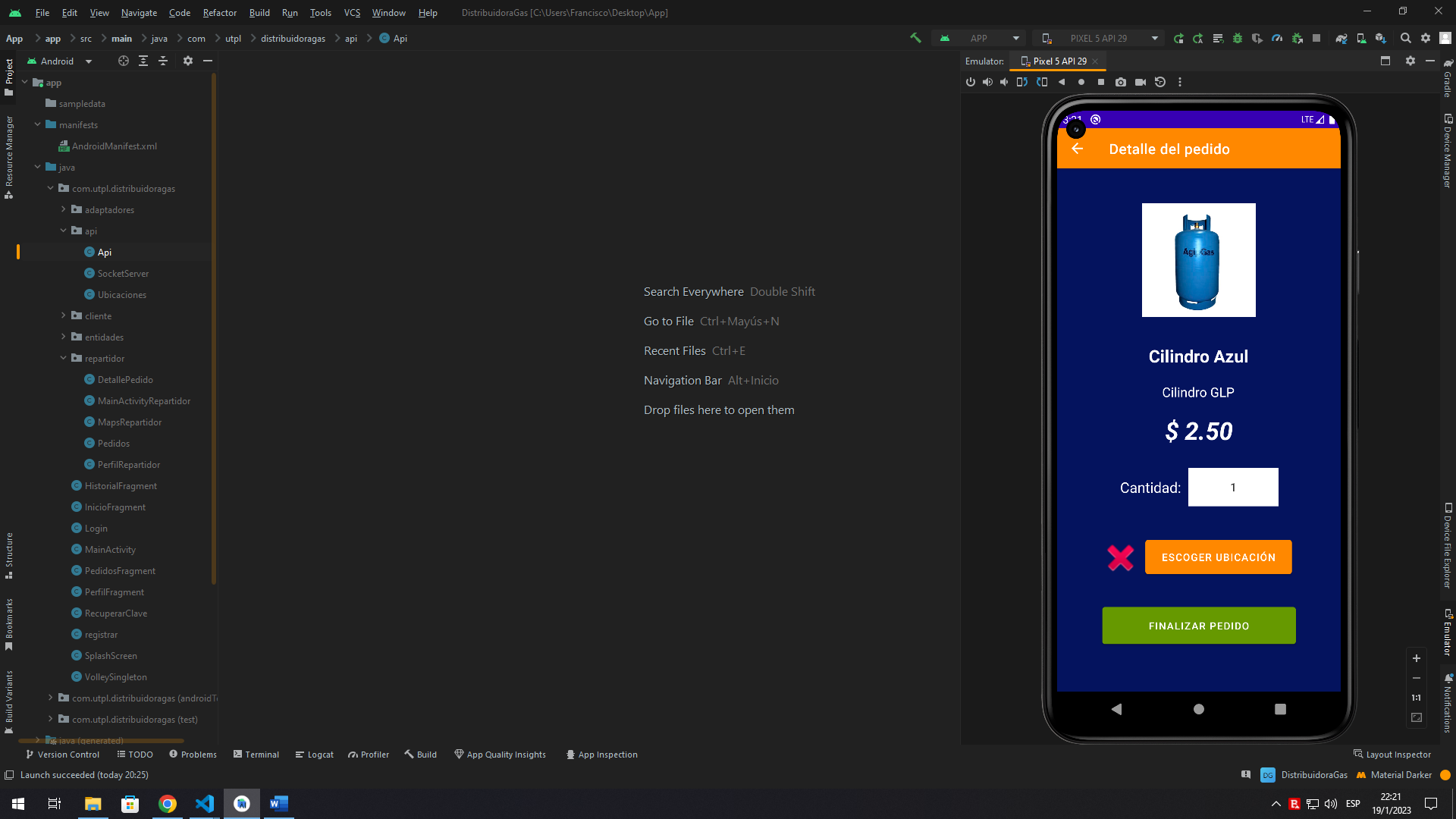
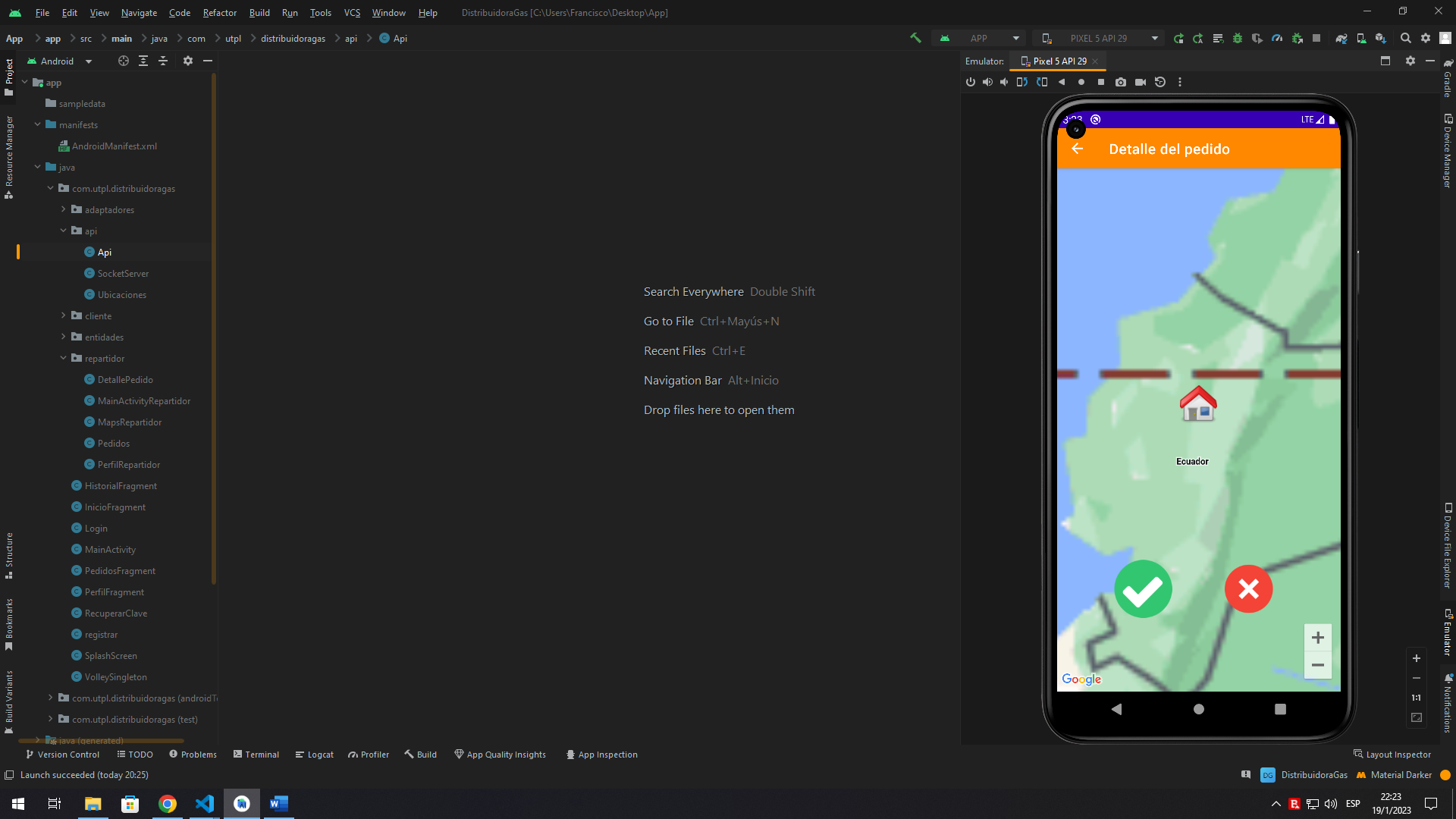
    }

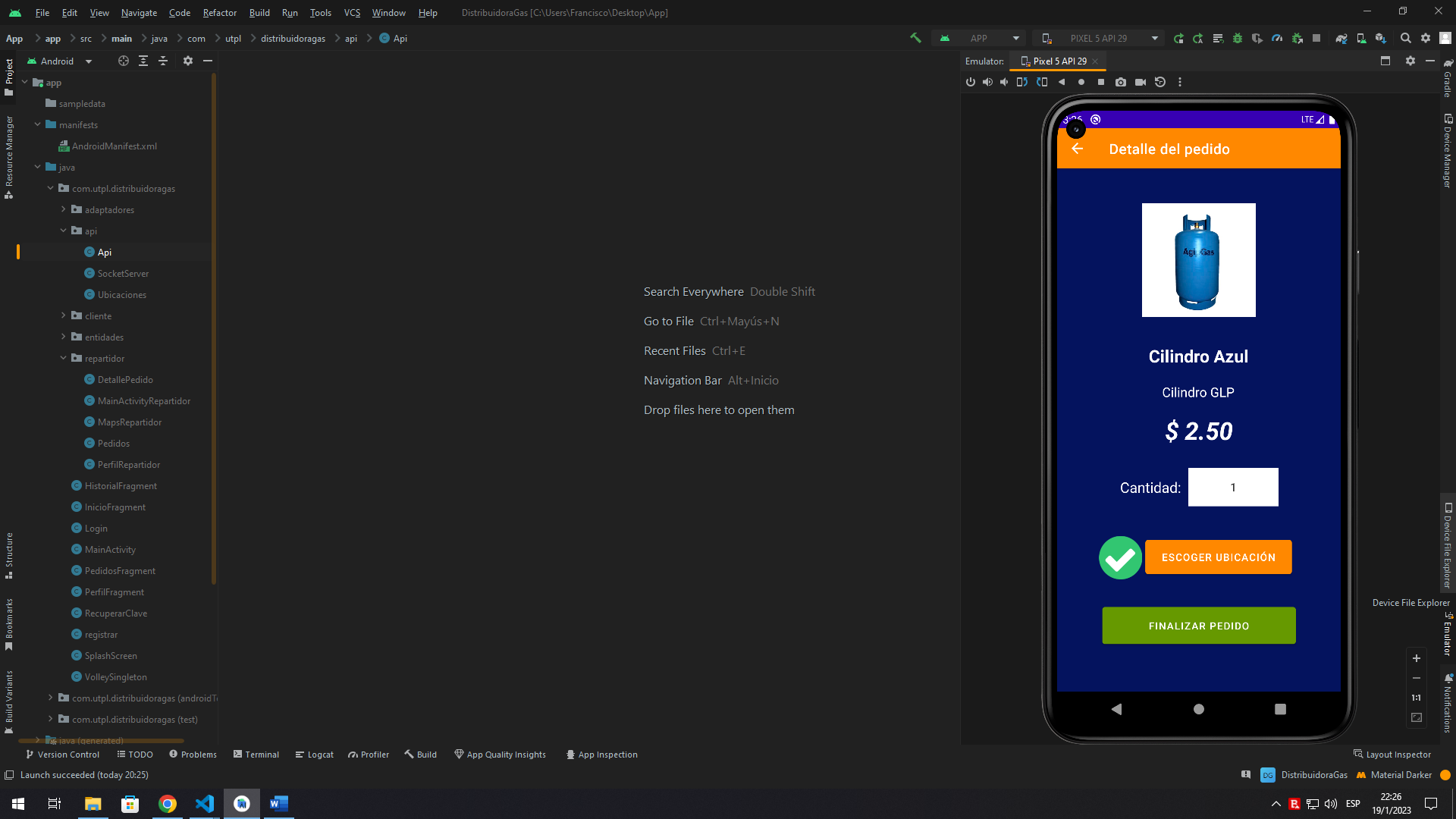
*private void* guardarPedido(){  
 String fechaActual = *new* SimpleDateFormat("yyyy-MM-dd").format(*new* Date());  
 String horaActual = *new* SimpleDateFormat("HH:mm:ss").format(*new* Date());  
 total = precio.getText().toString();  
 StringBuilder MyString = *new* StringBuilder(total);  
 MyString = MyString.deleteCharAt(0);  
 total = String.*valueOf*(MyString);  
 iva = Double.*parseDouble*(total) - (Double.*parseDouble*(total) / 1.12);  
 subtotal = (Double.*parseDouble*(total) / 1.12);  
 RequestQueue queue = Volley.*newRequestQueue*(*this*);  
 String url = Api.***API\_RUTA*** + "pedidos/crearPedido";  
 StringRequest stringRequest = *new* StringRequest(Request.*Method*.***POST***, url,  
 *new* Response.Listener<String>() {  
 @Override  
 *public void* onResponse(String response) {  
 *try* {  
 JSONObject data = *new* JSONObject(response);  
 *int* idPedidoBD = data.getInt("data");  
 idPedidoBD = idPedidoBD + 1;  
 Toast.*makeText*(getApplicationContext(), data.getString("mensaje"),Toast.***LENGTH\_LONG***).show();  
 mSocket = IO.*socket*(SocketServer.***SOCKET\_RUTA***);  
 mSocket.connect();  
 JSONObject datos = *new* JSONObject();  
 datos.put("idCliente", idUsuario);  
 datos.put("idProducto", String.*valueOf*(idProducto));  
 datos.put("cantidad", String.*valueOf*(cantidadEscogida));  
 mSocket.emit("position", datos);  
 buscarRepartidor(idPedidoBD);  
 } *catch* (JSONException e) {  
 e.printStackTrace();  
 } *catch* (URISyntaxException e) {  
 e.printStackTrace();  
 }  
 }  
 }, *new* Response.ErrorListener() {  
 @Override  
 *public void* onErrorResponse(VolleyError error) {  
 *if*(error.networkResponse.statusCode == 401) {  
 String responseBody = *null*;  
 *try* {  
 responseBody = *new* String(error.networkResponse.data, "utf-8");  
 JSONObject data = *new* JSONObject(responseBody);  
 String message = data.optString("mensaje");  
 Toast.*makeText*(getApplicationContext(), message,Toast.***LENGTH\_LONG***).show();  
 } *catch* (UnsupportedEncodingException e) {  
 e.printStackTrace();  
 } *catch* (JSONException e) {  
 e.printStackTrace();  
 }  
 }  
 }  
 })  
 {  
 *protected Map*<String, String> getParams() {  
 *Map*<String, String> params = *new* HashMap<>();  
 params.put("idCliente", idUsuario);  
 params.put("fecha", fechaActual);  
 params.put("hora", horaActual);  
 params.put("total", total);  
 params.put("subtotal", String.*valueOf*(subtotal));  
 params.put("iva", String.*valueOf*(iva));  
 params.put("idProducto", String.*valueOf*(idProducto));  
 params.put("cantidad", String.*valueOf*(cantidadEscogida));  
 *return* params;  
 }  
 };  
 VolleySingleton.*getInstance*(getApplicationContext()).addToRequestQueue(stringRequest);  
 Limpiar();  
}

*Fuente.* Prototipo de DistriGAS

A continuación, se mostrarán algunas capturas de pantalla de lo desarrollado en el sprint 4:

Figura 48

*Capturas de pantalla del Sprint 4*



*Fuente.* Prototipo de DistriGAS

## 4.9. Sprint 5: Módulo de pedidos, lista de pedidos y asignación de repartidor

Este Sprint estuvo planificado para 4 semanas aproximadamente, lo cual empezó con la creación de las interfaces graficas en la aplicación móvil en donde se implementó el historial de pedidos para el cliente, los pedidos asignados para el repartidor y la opción de asignar repartidor a los pedidos para que se pueda seguir con el ciclo del funcionamiento del prototipo.

Tabla 25

*Sprint Backlog 5*

|  |  |  |  |
| --- | --- | --- | --- |
| Sprint Backlog 5 | | | |
| Tarea ID | **Historia** | **Estimado**  **(Días)** | **Prioridad** |
| 11 | El usuario puede visualizar la lista de pedidos. | 15 | Alta |
| 12 | Como usuario se puede asignar un repartidor para suplir un pedido a domicilio. | 18 | Alta |

*Fuente.* Elaborado por el tesista

Tabla 26

*Actividades del Sprint 5*

|  |  |
| --- | --- |
| Actividades | Estimado/ (horas) |
| Diseño de la interfaz para el historial de pedidos del cliente en la aplicación móvil | 12 |
| Diseño de la interfaz para la lista de pedidos asignados del repartidor en la aplicación móvil | 10 |
| Codificación del historial de pedidos del cliente en la aplicación móvil | 12 |
| Codificación de la lista de pedidos asignados al repartidor en la aplicación móvil | 12 |
| Codificación de ubicaciones actuales de los repartidores en el mapa de Google | 10 |
| Implementación del historial de pedidos del cliente en la aplicación móvil | 8 |
| Implementación de la lista de pedidos asignados al repartidor en la aplicación móvil | 8 |
| TOTAL (horas trabajadas) | **105** |

*Fuente.* Elaborado por el tesista

Figura 49

*Codificación de la asignación automática de repartidor del Sprint 5*

asignarRepartidorAutomatico(req,res){

        Pedido.asignarRepartidorAutomatico(req.body, (err, data) => {

            if (err) {

                return res.status(501).json({

                    success: false,

                    mensaje: "hubo un error",

                    error: err

                })

            }

            return res.status(201).json({

                success: true,

                mensaje: "Repartidor asignado con éxito"

            })

        })

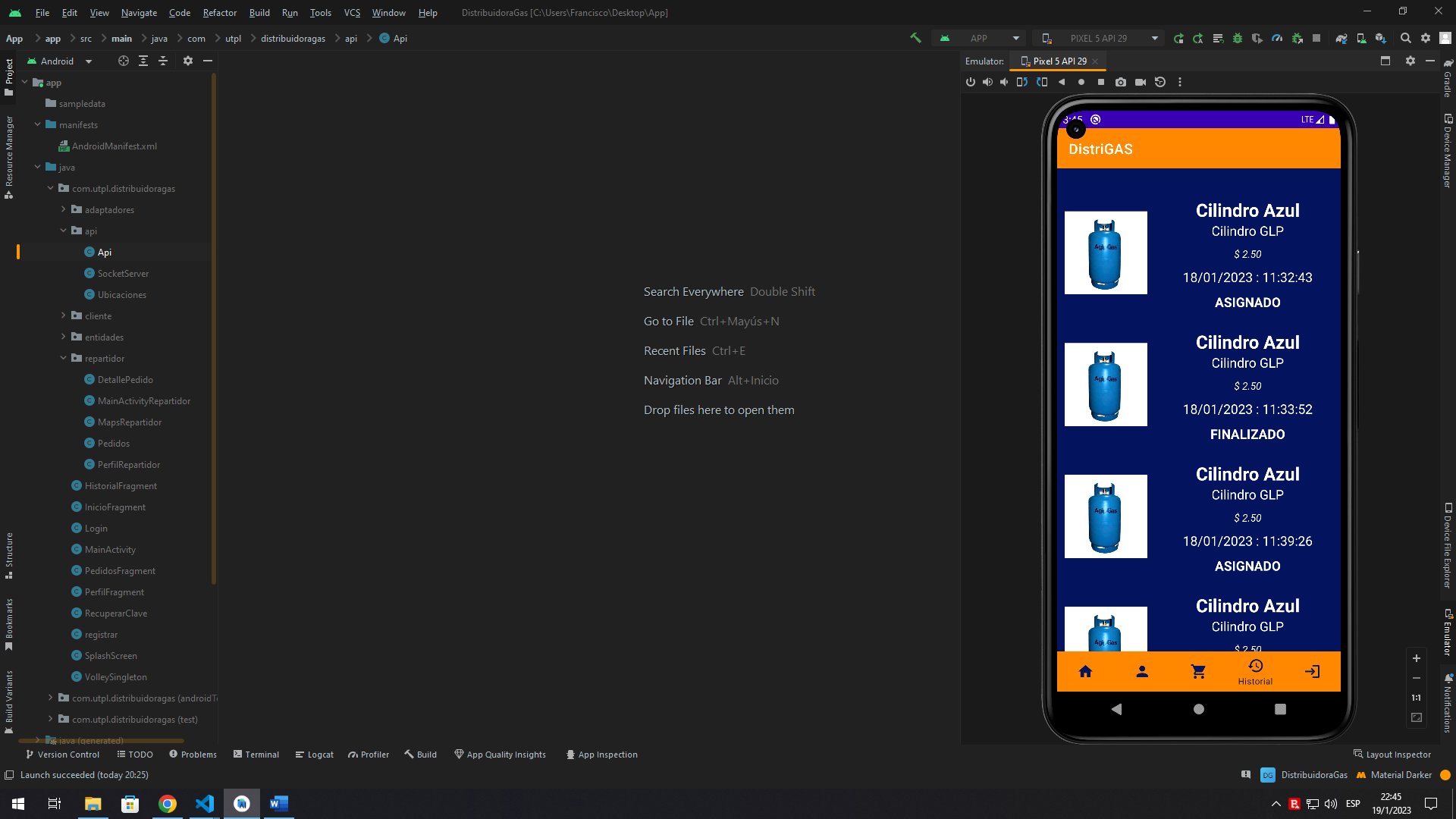
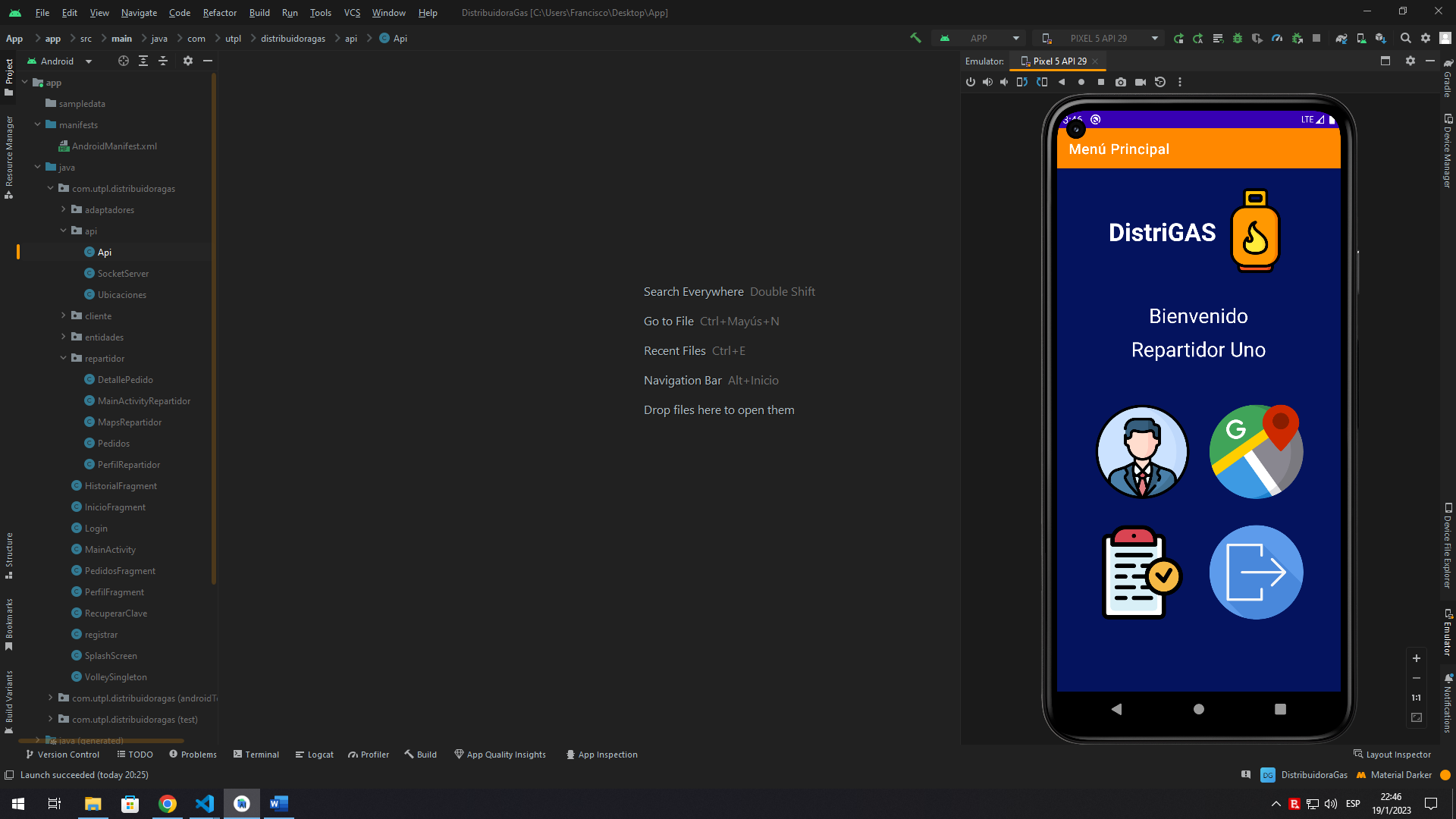
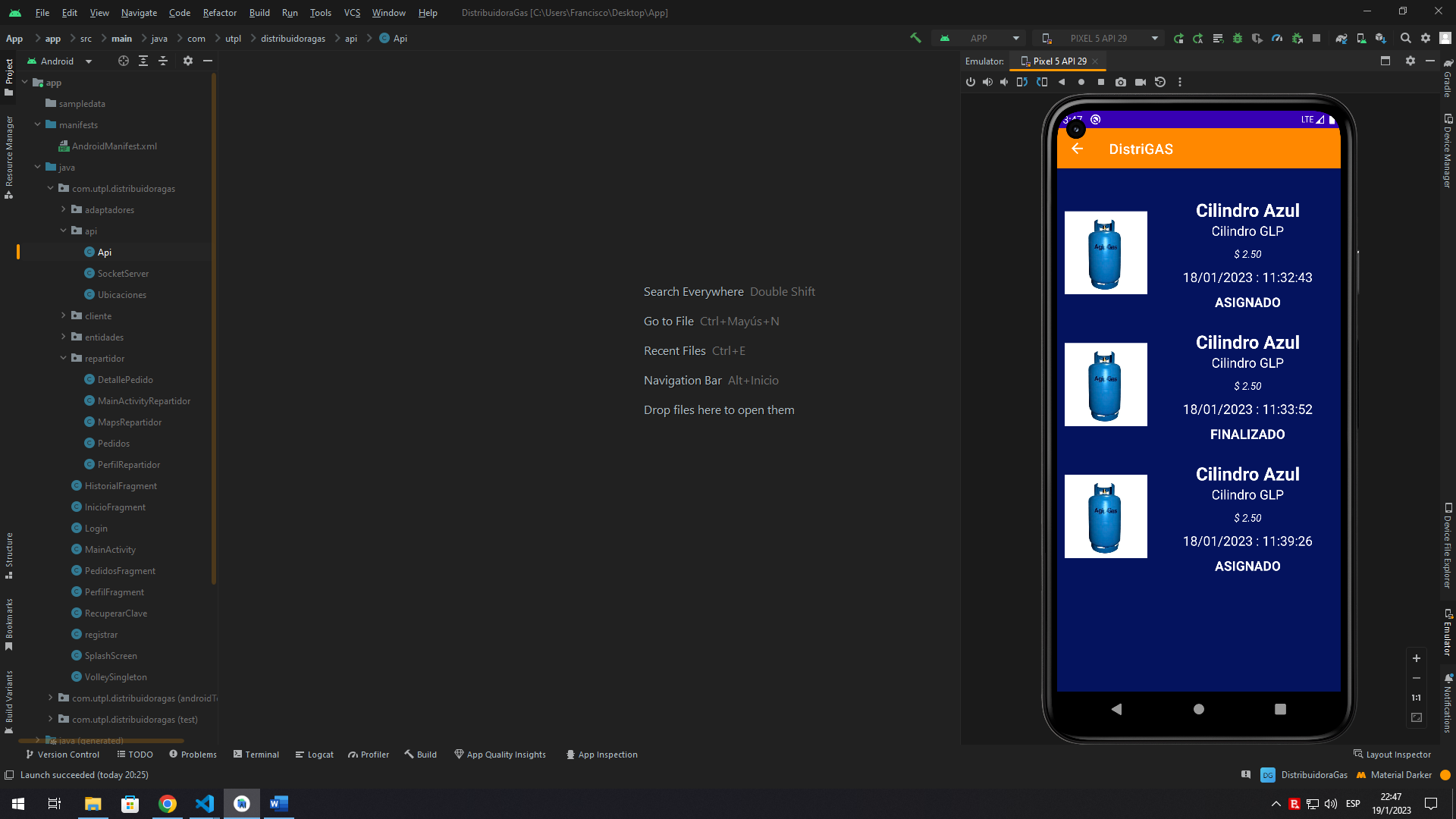
    }

*private void* buscarRepartidor(*int* idPedido){  
 dialog = *new* ProgressDialog(*this*);  
 dialog.setMessage("Espere mientras se asigna un repartidor");  
 dialog.show();  
 String url = Api.***API\_RUTA*** + "pedidos/buscarRepartidores";  
 RequestQueue queue = Volley.*newRequestQueue*(*this*);  
 StringRequest stringRequest = *new* StringRequest(Request.*Method*.***POST***, url,  
 *new* Response.Listener<String>() {  
 @Override  
 *public void* onResponse(String response) {  
 *float* masCorta = 100000;  
 *float* resultado;  
 *int* idRepartidor = 0;  
 Double latRepartidor = 0.0;  
 Double lngRepartidor = 0.0;  
 *try* {  
 JSONObject data = *new* JSONObject(response);  
 JSONArray json = data.optJSONArray("data");  
 *for*(*int* x=0; x<json.length();x++){  
 JSONObject jsonObject = *null*;  
 jsonObject = json.getJSONObject(x);  
 *if*(jsonObject.getDouble("lat") != 0) {  
 Location locationA = *new* Location("Cliente");  
 locationA.setLatitude(Ubicaciones.*latitudeApp*);  
 locationA.setLongitude(Ubicaciones.*longitudeApp*);  
 Location locationB = *new* Location("Repartidor");  
 locationB.setLatitude(jsonObject.getDouble("lat"));  
 locationB.setLongitude(jsonObject.getDouble("lng"));  
 resultado = locationA.distanceTo(locationB);  
 *if*(resultado < masCorta){  
 masCorta = resultado;  
 idRepartidor = jsonObject.getInt("id");  
 latRepartidor = jsonObject.getDouble("lat");  
 lngRepartidor = jsonObject.getDouble("lng");  
 }  
 }  
 }  
 asignarRepartidor(idRepartidor, latRepartidor, lngRepartidor, idPedido);  
 dialog.hide();  
 } *catch* (JSONException e) {  
 e.printStackTrace();  
 }  
 }  
 }, *new* Response.ErrorListener() {  
 @Override  
 *public void* onErrorResponse(VolleyError error) {  
 *if*(error.networkResponse.statusCode == 401) {  
 String responseBody = *null*;  
 *try* {  
 responseBody = *new* String(error.networkResponse.data, "utf-8");  
 JSONObject data = *new* JSONObject(responseBody);  
 String message = data.optString("mensaje");  
 Toast.*makeText*(getApplicationContext(), message,Toast.***LENGTH\_LONG***).show();  
 dialog.hide();  
 } *catch* (UnsupportedEncodingException e) {  
 e.printStackTrace();  
 } *catch* (JSONException e) {  
 e.printStackTrace();  
 }  
 }  
 }  
 })  
 {  
 *protected Map*<String, String> getParams() {  
 *Map*<String, String> params = *new* HashMap<>();  
 params.put("idCliente", idUsuario);  
 *return* params;  
 }  
 };  
 VolleySingleton.*getInstance*(*this*).addToRequestQueue(stringRequest);  
 Limpiar();  
}  
  
*private void* asignarRepartidor(*int* idRepartidor, Double latRepartidor, Double lngRepartidor, *int* idPedido){  
 String url = Api.***API\_RUTA*** + "pedidos/asignarRepartidorAutomatico";  
 RequestQueue queue = Volley.*newRequestQueue*(*this*);  
 StringRequest stringRequest = *new* StringRequest(Request.*Method*.***POST***, url,  
 *new* Response.Listener<String>() {  
 @Override  
 *public void* onResponse(String response) {  
 *try* {  
 JSONObject data = *new* JSONObject(response);  
 Toast.*makeText*(getApplicationContext(), data.getString("mensaje"),Toast.***LENGTH\_LONG***).show();  
 Intent intent = *new* Intent(getApplicationContext(), MainActivity.*class*);  
 startActivity(intent);  
 } *catch* (JSONException e) {  
 e.printStackTrace();  
 }  
 }  
 }, *new* Response.ErrorListener() {  
 @Override  
 *public void* onErrorResponse(VolleyError error) {  
 *if*(error.networkResponse.statusCode == 401) {  
 String responseBody = *null*;  
 *try* {  
 responseBody = *new* String(error.networkResponse.data, "utf-8");  
 JSONObject data = *new* JSONObject(responseBody);  
 String message = data.optString("mensaje");  
 Toast.*makeText*(getApplicationContext(), message,Toast.***LENGTH\_LONG***).show();  
 } *catch* (UnsupportedEncodingException e) {  
 e.printStackTrace();  
 } *catch* (JSONException e) {  
 e.printStackTrace();  
 }  
 }  
 }  
 })  
 {  
 *protected Map*<String, String> getParams() {  
 *Map*<String, String> params = *new* HashMap<>();  
 params.put("idPedido", String.*valueOf*(idPedido));  
 params.put("idRepartidor", String.*valueOf*(idRepartidor));  
 params.put("origenLat", String.*valueOf*(latRepartidor));  
 params.put("origenLng", String.*valueOf*(lngRepartidor));  
 params.put("destinoLat", String.*valueOf*(Ubicaciones.*latitudeApp*));  
 params.put("destinoLng", String.*valueOf*(Ubicaciones.*longitudeApp*));  
 *return* params;  
 }  
 };  
 VolleySingleton.*getInstance*(*this*).addToRequestQueue(stringRequest);  
 Limpiar();  
}

*Fuente.* Elaborado por el tesista

A continuación, se mostrarán algunas capturas de pantalla de lo desarrollado en el sprint 5:

Figura 50

*Capturas de pantalla del Sprint 5*

*Fuente.* Prototipo de DistriGAS

## 4.10. Sprint 6: Módulo de pedidos, traza de rutas, notificaciones y fin de pedidos

Este Sprint estuvo planificado para 5 semanas aproximadamente, lo cual empezó con la creación de las interfaces graficas en la aplicación móvil en donde se implementó el trazado de rutas tanto para el cliente como para el repartidor, también se configuraron las notificaciones para avisar al cliente la llegada del repartidor y finalmente se implementó la finalización de un pedido. Este es uno de los sprint más importantes ya que el trazado de rutas y la entrega de pedidos es la funcionalidad más importante del prototipo.

Tabla 27

*Sprint Backlog 6*

|  |  |  |  |
| --- | --- | --- | --- |
| Sprint Backlog 6 | | | |
| Tarea ID | **Historia** | **Estimado**  **(Días)** | **Prioridad** |
| 13 | El sistema debe trazar la ruta de entrega desde la ubicación actual del conductor hacia el cliente. | 18 | Alta |
| 14 | El sistema debe notificar cuando el repartidor llega a la ubicación del cliente. | 8 | Alta |
| 15 | El usuario puede finalizar un pedido a domicilio. | 8 | Alta |

*Fuente.* Elaborado por el tesista

Tabla 28

*Actividades del Sprint 6*

|  |  |
| --- | --- |
| Actividades | Estimado/ (horas) |
| Diseño de la interfaz del mapa de Google para el trazado de rutas para el cliente en la aplicación móvil | 20 |
| Diseño de la interfaz del mapa de Google para el trazado de rutas para el repartidor en la aplicación móvil | 12 |
| Diseño de la interfaz del detalle del pedido para el repartidor en la aplicación móvil | 18 |
| Diseño de la notificación para avisar al cliente la llegada del repartidor en la aplicación móvil | 12 |
| Diseño del botón para la finalización de un pedido en la aplicación móvil | 10 |
| Codificación del historial de pedidos en el cliente para que pueda acceder al mapa con la ubicación del repartidor en la aplicación móvil | 8 |
| Codificación del mapa de Google para el trazado de rutas para el cliente en la aplicación móvil | 22 |
| Codificación del mapa de Google para el trazado de rutas para el repartidor en la aplicación móvil | 12 |
| Codificación de las notificaciones de llegada del repartidor al cliente en la aplicación móvil | 8 |
| Codificación del botón para la finalización de un pedido en la aplicación móvil | 10 |
| Implementación de las rutas en el mapa de Google del cliente en la aplicación móvil | 12 |
| Implementación de las rutas en el mapa de Google del repartidor en la aplicación móvil | 12 |
| Implementación del detalle de un pedido asignado al repartidor en la aplicación móvil | 8 |
| Implementación de las notificaciones al cliente en la aplicación móvil | 8 |
| Implementación del botón de finalizar pedido para el repartidor en la aplicación móvil | 6 |
| TOTAL (horas trabajadas) | **178** |

*Fuente.* Elaborado por el tesista

Figura 51

*Codificación de los mapas de Google y trazado de rutas del Sprint 6*

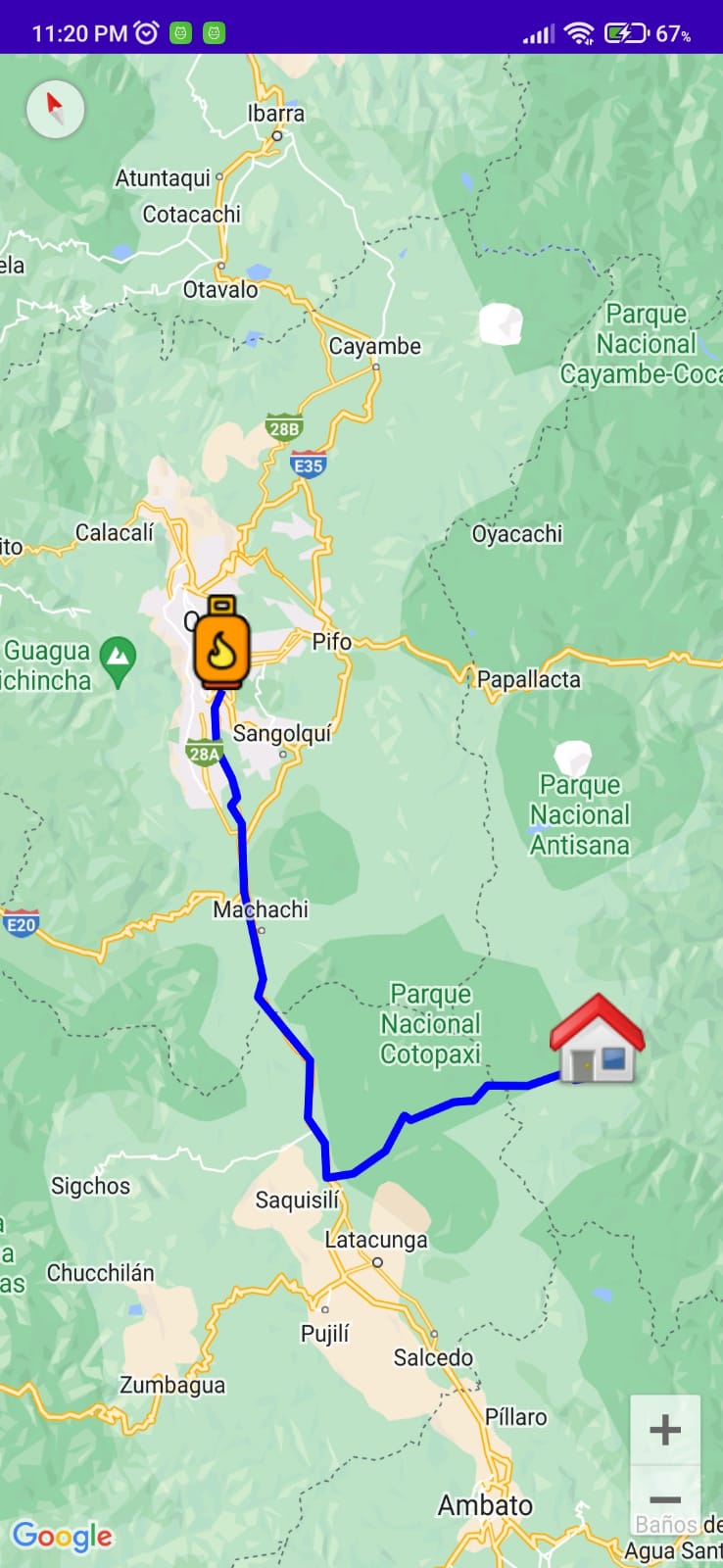
@Override  
*public void* onMapReady(GoogleMap googleMap) {  
 mMap = googleMap;  
 obtenerUbicacion();  
 idPedido = getIntent().getExtras().getInt("id");  
 latRepartidor = getIntent().getExtras().getDouble("lat");  
 lngRepartidor = getIntent().getExtras().getDouble("lng");  
 LatLng ubicacionActual = *new* LatLng(Ubicaciones.*latitudeApp*, Ubicaciones.*longitudeApp*);  
 LatLng ubicacionRepartidor = *new* LatLng(latRepartidor, lngRepartidor);  
 BitmapDescriptor icon = BitmapDescriptorFactory.*fromResource*(R.drawable.***cilindromapa***);  
 BitmapDescriptor iconCasa = BitmapDescriptorFactory.*fromResource*(R.drawable.***casa***);  
 mMap.addMarker(*new* MarkerOptions().position(ubicacionActual).title("Ubicacion actual").draggable(*false*).icon(iconCasa));  
 mMap.addMarker(*new* MarkerOptions().position(ubicacionRepartidor).title("Repartidor").draggable(*false*).icon(icon));  
 mMap.moveCamera(CameraUpdateFactory.*newLatLng*(ubicacionActual));  
 mMap.moveCamera(CameraUpdateFactory.*newLatLng*(ubicacionActual));  
 mMap.animateCamera(CameraUpdateFactory.*zoomTo*(18), *null*);  
 mMap.getUiSettings().setZoomControlsEnabled(*true*);  
 drawPolylines(ubicacionActual, ubicacionRepartidor);  
}

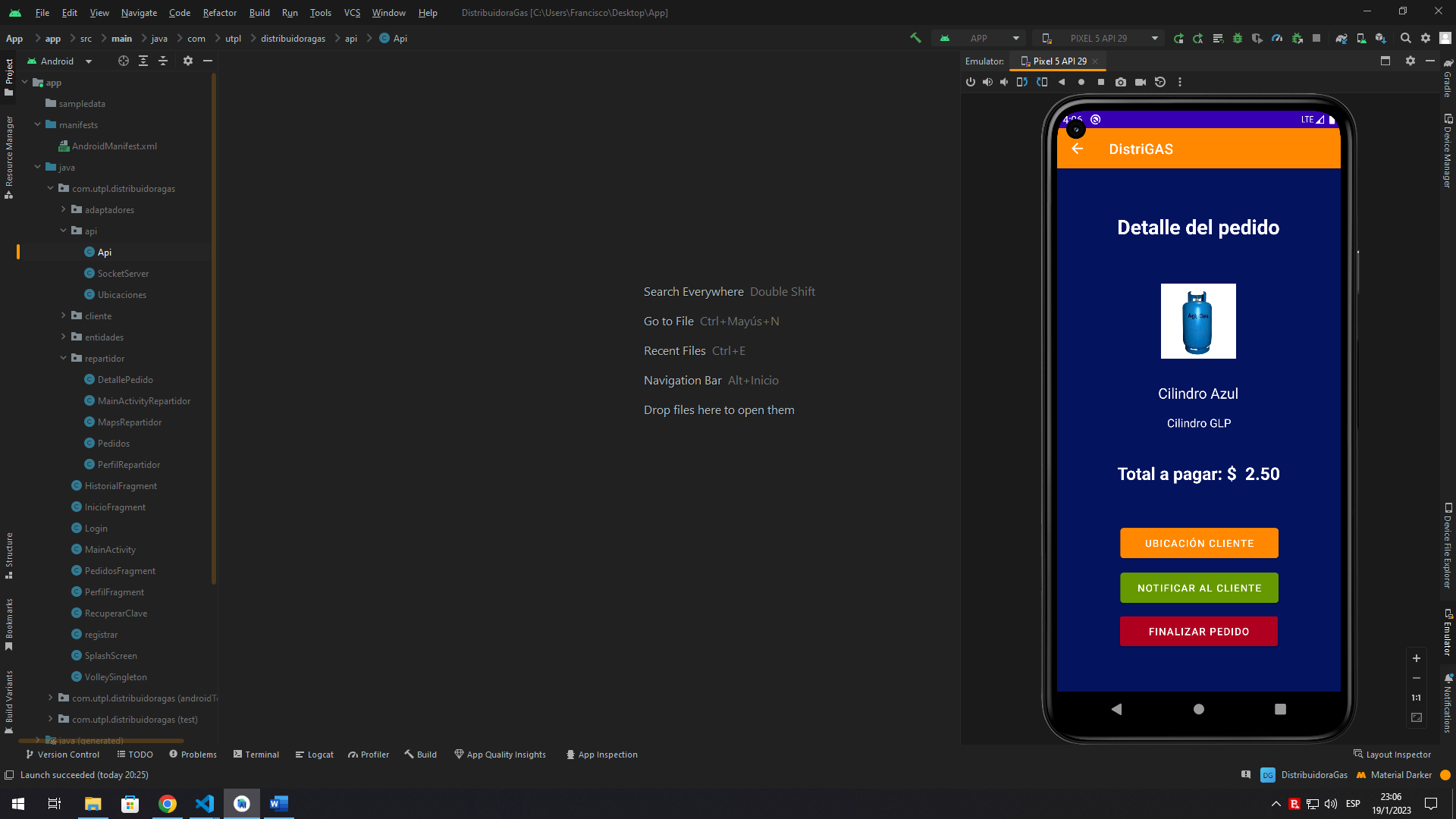
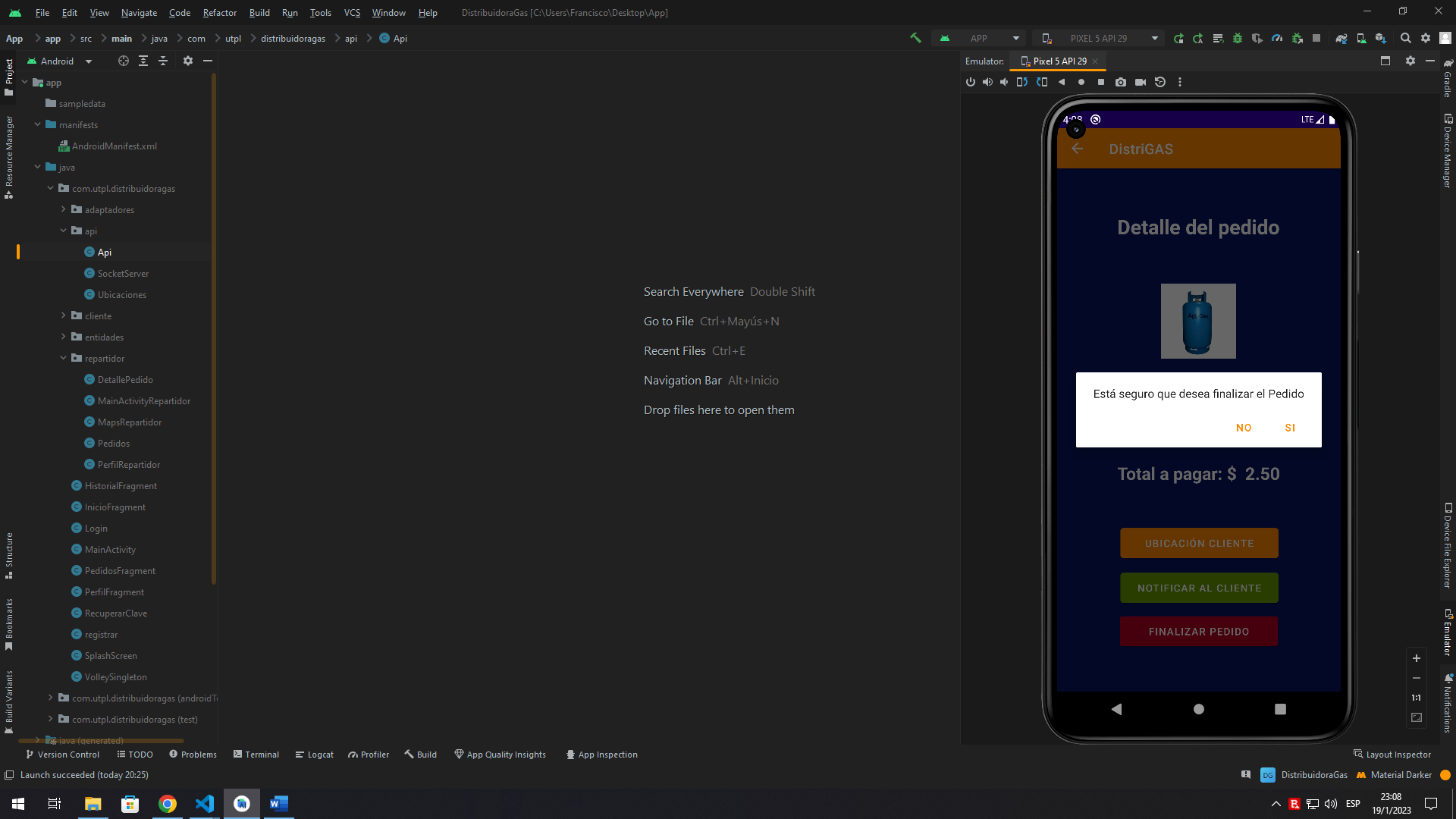
*private void* drawPolylines(LatLng origen, LatLng destino) {  
 progressDialog = *new* ProgressDialog(MapsCliente.*this*);  
 progressDialog.setMessage("Espere, Mientras se traza la ubicación del Repartidor.");  
 progressDialog.setCancelable(*false*);  
 progressDialog.show();  
 String url = getDirectionsUrl(origen, destino);  
 Log.*d*("url", url + "");  
 MapsCliente.DownloadTask downloadTask = *new* MapsCliente.DownloadTask();  
 downloadTask.execute(url);  
}  
  
*private class* DownloadTask *extends* AsyncTask<String, Void, String> {  
 @Override  
 *protected* String doInBackground(String... url) {  
 String data = "";  
 *try* {  
 data = downloadUrl(url[0]);  
 } *catch* (Exception e) {  
 Log.*d*("Background Task", e.toString());  
 }  
 *return* data;  
 }  
 @Override  
 *protected void* onPostExecute(String result) {  
 *super*.onPostExecute(result);  
 MapsCliente.ParserTask parserTask = *new* MapsCliente.ParserTask();  
 parserTask.execute(result);  
 }  
}

*Fuente.* Prototipo de DistriGAS

A continuación, se mostrarán algunas capturas de pantalla de lo desarrollado en el sprint 7:

Figura 52

*Capturas de pantalla del Sprint 6*



*Fuente.* Prototipo de DistriGAS

# Bibliografía

Acurio Acurio, C. J. (agosto de 2017). *Desarrollo de una aplicación móvil que envíe una solicitud de requerimiento de gas doméstico a los distribuidores o a los vehículos de distribución más cercanos a una ubicación en la ciudad de Quito mediante Georeferenciación*. Obtenido de https://dspace.ups.edu.ec/handle/123456789/14528

Alvarez, M. A. (28 de julio de 2020). *Qué es MVC*. Obtenido de https://desarrolloweb.com/articulos/que-es-mvc.html

Angel, R. (23 de septiembre de 2019). *Qué es Python: Características, evolución y futuro.* Obtenido de https://openwebinars.net/blog/que-es-python/

Angulo, R. (2013). *Aplicaciones móviles híbridas: lo mejor de dos mundos.* Obtenido de https://184.182.233.153/rid=1NTQ9NMKD-R1SKBP-24M4/Aplicaciones%20moviles%20hibridas-%20lo%20mejor%20de%20dos%20mundos.pdf

Arias Melo, Y. H. (2017). *Retos de la seguridad informática en servidores Node.js*. Obtenido de http://repository.unipiloto.edu.co/handle/20.500.12277/2671

Ayudaley. (2020). *Bases de datos relacional ¿Qué es y sus características?* Obtenido de https://ayudaleyprotecciondatos.es/bases-de-datos/relacional/

Bautista García, I. J. (30 de marzo de 2021). *Backend y Frontend, ¿Qué es y cómo funcionan en la programación?* Obtenido de https://www.servnet.mx/blog/backend-y-frontend-partes-fundamentales-de-la-programaci%C3%B3n-de-una-aplicaci%C3%B3n-web

Bayas, G. (12 de julio de 2022). *¿Qué es CSS?* Obtenido de https://www.hostinger.es/tutoriales/que-es-css

Beynon Davies, P. (2018). *Sistemas de bases de datos*. Obtenido de https://books.google.es/books?hl=es&lr=&id=XjbeDwAAQBAJ&oi=fnd&pg=PR5&dq=base+de+datos+sistemas&ots=DIB\_zOKYGR&sig=fxTda7uS9f0oF-qrrHNchLbmGdM#v=onepage&q&f=false

Borges, E. (23 de noviembre de 2018). *Caracteristicas de un servidor web*. Obtenido de https://blog.infranetworking.com/caracteristicas-servidor-web/#:~:text=Los%20paneles%20de%20control%20nos,Crear%2C%20modificar%20y%20eliminar%20dominios.

Caihuara Sossa, F. D. (2019). *APLICACIONES WEB PROGRESIVAS.* Obtenido de http://dicyt.uajms.edu.bo/revistas/index.php/ciencia-sur/article/view/728/728

Cantabria. (29 de diciembre de 2020). *Estándar de codificación Java*. Obtenido de https://bit.ly/3FcxJRQ

Carlosama Jingo, S. S., & Fárez Rodriguez, V. N. (mayo de 2022). *Desarrollo de un sistema prototipo para la automatización de los procesos de venta e inventario de un distribuidor de gas licuado de petróleo.* Obtenido de https://bibdigital.epn.edu.ec/handle/15000/22804

Carranza, A. (06 de abril de 2022). *Las ventajas y desventajas de Java que debes conocer antes de programar tu primer software*. Obtenido de https://www.crehana.com/blog/desarrollo-web/ventajas-desventajas-java/

Cesar, R. (2015). *PostgreSQL: qué es, cómo funciona y cuáles son sus ventajas.* Obtenido de https://platzi.com/blog/que-es-postgresql/

Chavez, J. (2020). *¿Qué es un Sistema informático? Componentes, características y ejemplos*. Obtenido de https://www.ceupe.com/blog/sistema-informatico.html?dt=1660580785710

Deyimar, A. (03 de junio de 2022). *¿Qué es Bootstrap? – Una guía para principiantes*. Obtenido de https://www.hostinger.es/tutoriales/que-es-bootstrap

Digital Guide. (07 de 08 de 2019). *Lenguajes de programación web: los más usados en Internet.* Obtenido de https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/lenguajes-de-programacion-web/

Drew. (03 de diciembre de 2019). *Ventajas y desventajas de la metodología Scrum*. Obtenido de https://blog.wearedrew.co/productividad/-ventajas-y-desventajas-de-la-metodologia-scrum

Drumond, C. (2021). *Scrum*. Obtenido de https://www.atlassian.com/es/agile/scrum

Eras Camacho, F., & Hidalgo Lascano, P. (20 de enero de 2021). *Aplicación móvil en Android para la adquisición de Gas Licuado de Petróleo (GLP) doméstico*. Obtenido de https://dialnet.unirioja.es/servlet/articulo?codigo=7741837

European Knowledge Center. (08 de julio de 2019). *Servidores web*. Obtenido de https://www.ticportal.es/glosario-tic/servidores

García Monsálvez, J. C. (2017). *Python como primer lenguaje de programación textual en la Enseñanza Secundaria.* Obtenido de http://digital.casalini.it/10.14201/eks2017182147162

Gutierrez. (Septiembre de 2009). *JavaScript Conceptos básicos y avanzados*. Obtenido de https://books.google.es/books?id=gsxVpvEC4iUC&lpg=PA19&ots=UxXdc6Ie6k&dq=ventajas%20de%20javascript&lr&hl=es&pg=PP1#v=onepage&q=ventajas%20de%20javascript&f=false

Henriquez, R. (2018). *Desarrollador Web PHP*. Obtenido de https://books.google.es/books?id=cjBpDwAAQBAJ&printsec=frontcover&hl=es#v=onepage&q&f=false

HostingPedia. (24 de enero de 2019). *MySQL*. Obtenido de https://hostingpedia.net/mysql.html

Humanñahui Aedo, E. (2021). *Sistema web para la optimización de la gestión de ventas de una empresa comercializadora*. Obtenido de http://repositorio.uwiener.edu.pe/bitstream/handle/20.500.13053/5544/T061\_70021798\_T.pdf?sequence=1&isAllowed=y

Imbaquingo Esparza, D. E. (10 de diciembre de 2018). *Estudio de la herramienta “android studio” con aplicativo de gestión de proveedores, clientes y gestión de proformas para el taller mecánico “El Golpe Mágico"*. Obtenido de http://repositorio.utn.edu.ec/handle/123456789/8682

INFO-IUTEPI. (2017). *PHP*. Obtenido de https://sites.google.com/site/phpinfoiutepi/caracteristicas-de-php-1

JAVA. (2022). *¿Qué es la tecnología Java y por qué la necesito?* Obtenido de https://www.java.com/es/download/help/whatis\_java.html#:~:text=Java%20es%20una%20plataforma%20inform%C3%A1tica,crean%20muchos%20servicios%20y%20aplicaciones.

KeepCoding. (20 de enero de 2022). *Los 5 lenguajes más usados para la programación en Android*. Obtenido de https://keepcoding.io/blog/programacion-en-android/

Kinsta. (25 de julio de 2022). *¿Qué es Express.js? Todo lo que Debes Saber*. Obtenido de https://kinsta.com/es/base-de-conocimiento/que-es-express/

Lucas, J. (04 de septiembre de 2019). *Qué es NodeJS y para qué sirve*. Obtenido de https://openwebinars.net/blog/que-es-nodejs/

Maida, E. G., & Pacienzia, J. (2015). *Metodologías de desarrollo de software*. Obtenido de https://repositorio.uca.edu.ar/handle/123456789/522

Maldeadora, N. (2018). *Qué es Frontend y Backend: diferencias y características - Platzi*. Obtenido de https://platzi.com/blog/que-es-frontend-y-backend/

Martinez, C. (09 de junio de 2021). *Arquitectura de software y sus beneficios*. Obtenido de https://devexperto.com/arquitectura-del-software/

Mdn Web Docs. (13 de agosto de 2022). *Fundamentos de JavaScript*. Obtenido de https://developer.mozilla.org/es/docs/Learn/Getting\_started\_with\_the\_web/JavaScript\_basics

Mdn Web Docs HTTP. (07 de noviembre de 2022). *Métodos de petición HTTP*. Obtenido de https://developer.mozilla.org/es/docs/Web/HTTP/Methods

Mendoza A., A., & López, R. (07 de febrero de 2018). *Base de datos*. Obtenido de https://repositorio.uchile.cl/handle/2250/151632

Miteris. (2018). *¿Qué es Javascript? Características y Librerías*. Obtenido de https://www.miteris.com/blog/que-es-javascript-caracteristicas-librerias/

Molina Montero, B., Vite Cevallos, H., & Cuesta, D. (2018). *Metodologías ágiles frente a las tradicionales en el proceso de desarrollo de software*. Obtenido de https://www.researchgate.net/profile/Harry-Vite-Cevallos/publication/327537074\_Metodologias\_agiles\_frente\_a\_las\_tradicionales\_en\_el\_proceso\_de\_desarrollo\_de\_software/links/5b942061a6fdccfd542a2b13/Metodologias-agiles-frente-a-las-tradicionales-en-el-proce

Molina, D. (09 de diciembre de 2021). *Qué es un product backlog y cómo hacer uno Guía Scrum*. Obtenido de https://www.iebschool.com/blog/que-es-un-product-backlog-y-como-hacer-uno-guia-scrum-agile-scrum/

Muradas, Y. (29 de enero de 2021). *Kotlin vs Java*. Obtenido de https://openwebinars.net/blog/kotlin-vs-java/#:~:text=JDK%20(Java%20Development%20Kit)%3A,JRE%20que%20describiremos%20en%20breve.

Nielfa S, J. (2020). *Android Studio: El entorno de desarrollo oficial de Android*. Obtenido de https://scoreapps.com/blog/es/android-studio/

Pérez Ibarra, S. G., Quispe, J. R., Mullicundo, F. F., & Lamas, D. A. (abril de 2021). *Herramientas y tecnologías para el desarrollo web desde el FrontEnd al BackEnd*. Obtenido de http://sedici.unlp.edu.ar/handle/10915/120476

Ramirez Vique, R. (2019). *Métodos para el desarrollo de aplicaciones móviles*. Obtenido de http://190.57.147.202:90/jspui/bitstream/123456789/464/1/Tecnologia\_y\_desarrollo\_en\_dispositivos\_moviles.pdf

Ríos Marín, J. S. (2019). *Aplicación móvil*. Obtenido de https://hdl.handle.net/10656/11141

Robledano, A. (24 de septiembre de 2019). *Qué es MySQL: Características y ventajas*. Obtenido de https://openwebinars.net/blog/que-es-mysql/

Romero, C. (06 de noviembre de 2019). *CARACTERÍSTICAS MÁS IMPORTANTES DE UN SERVIDOR WEB*. Obtenido de https://alsurestudio.es/caracteristicas-mas-importantes-de-un-servidor-web/

Sáez Hurtado, J. (03 de diciembre de 2021). *Cómo funciona la Metodología Scrum: Qué es y cómo utilizarla*. Obtenido de https://www.iebschool.com/blog/metodologia-scrum-agile-scrum/

San Juan, V. (27 de abril de 2016). *Ventajas de los sistemas web*. Obtenido de https://www.aeurus.cl/blog/ventajas-de-los-sistemas-web

Santaella, J. (26 de abril de 2022). *¿Qué es Android Studio?* Obtenido de https://talently.tech/blog/que-es-android-studio/

Sarzosa Bombón, C. E. (13 de noviembre de 2018). *ESTUDIO DEL FRAMEWORK OPENSOURCE BOOTSTRAP;SISTEMA;SEGUIMIENTO DE ACTIVIDADES ADMINISTRATIVAS*. Obtenido de http://repositorio.utn.edu.ec/handle/123456789/8641

Smith, J. (2012). *Cliente Http asincrónico de Android Una biblioteca cliente Http basada en devolución de llamada para Android*. Obtenido de https://programmerclick.com/article/67251435597/

Trafaniuc, V. (10 de enero de 2022). *¿Sabes qué es Google APIs? Descúbrelo ahora mismo*. Obtenido de https://maplink.global/blog/es/que-son-google-apis/

Vasquez Lozano, Y. Y. (2019). *ESTRUCTURA Y BASE DE DATOS. Concepto y tipos de datos, Arrays, listas enlazadas, pilas, colas. Organización de base de datos. Tipos de base de datos, entidades y objetos.* Obtenido de https://repositorio.une.edu.pe/handle/20.500.14039/5113

Vazquez Ortíz, Y., Mier Pierre, L., & Sotolongo León, A. R. (2016). *Características no relacionales de PostgreSQL: incremento del rendimiento en el uso de datos JSON*. Obtenido de http://scielo.sld.cu/scielo.php?pid=S2227-18992016000600006&script=sci\_arttext&tlng=en