UNIVERSIDAD POLITÉCNICA SALESIANA
SEDE QUITO

CARRERA:
INGENIERÍA DE SISTEMAS

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

TEMA:
DISEÑO E IMPLEMENTACIÓN DE UNA APLICACIÓN MÓVIL MULTIPLATAFORMA PARA FACILITAR LA COMUNICACIÓN DE LA EMPRESA FERRO TORRES CON SUS CLIENTES

AUTOR:
KEVIN ALEXANDER CÓRDOVA SANTOS

TUTOR:
DANIEL GIOVANNY DÍAZ ORTIZ

Quito, agosto del 2021

CESIÓN DE DERECHOS DE AUTOR

Yo Kevin Alexander Córdova Santos, con documento de identificación N° 1726002106, manifiesto mi voluntad y cedo a la Universidad Politécnica Salesiana la titularidad sobre los derechos patrimoniales en virtud de que soy autor del trabajo de titulación intitulado: DISEÑO E IMPLEMENTACIÓN DE UNA APLICACIÓN MÓVIL MULTIPLATAFORMA PARA FACILITAR LA COMUNICACIÓN DE LA EMPRESA FERRO TORRES CON SUS CLIENTES, mismo que ha sido desarrollado para optar por el título de: INGENIERO DE SISTEMAS, en la Universidad Politécnica Salesiana, quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente.

En aplicación a lo determinado en la Ley de Propiedad Intelectual, en mi condición de autor me reservo los derechos morales de la obra antes citada. En concordancia, suscribo este documento en el momento que hago entrega del trabajo final en formato digital a la Biblioteca de la Universidad Politécnica Salesiana.

……

Kevin Alexander Córdova Santos
1726002106

Quito, agosto del 2021

DECLARATORIA DE COATURÍA DEL DOCENTE TUTOR

Yo declaro que bajo mi dirección y asesoría fue desarrollado el Proyecto Técnico, con el tema: DISEÑO E IMPLEMENTACIÓN DE UNA APLICACIÓN MÓVIL MULTIPLATAFORMA PARA FACILITAR LA COMUNICACIÓN DE LA EMPRESA FERRO TORRES CON SUS CLIENTES, realizado por Kevin Alexander Córdova Santos, obteniendo un producto que cumple con todos los requisitos estipulados por la Universidad Politécnica Salesiana, para ser considerado como trabajo final de titulación.

Quito, agosto 2021

……
Daniel Giovanny Díaz Ortiz
C.I: 1716975501

DEDICATORIA

Dedico el presente proyecto técnico de titulación como primer lugar a Dios por darme la vida, su bendición y la sabiduría suficiente para completar este documento con todas las dificultades que se presentaron en el camino, para él siempre será dedicado todas las metas que realice o planifique.

A mi familia, mis dos padres y mi hermano menor, por regalarme la oportunidad de estudiar y su apoyo, tanto económico como emocional, que siempre están presentes en mi vida para aportar un consejo y ayuda incondicional en las etapas difíciles de este proyecto técnico.

Dedico este proyecto a las personas que requieran construir una aplicación móvil y que este proyecto pueda aportar de mejor manera para maximizar a la tecnología dentro de la vida de cada persona para llevar a un mundo más inclusivo y colaborativo.

Kevin Alexander Córdova Santos.

AGRADECIMIENTO

Agradezco sobre todas las cosas, a Dios, por guiarme por el sendero de la sabiduría y darme la oportunidad de vivir un día a la vez, que su santa palabra me estuvo en constante apoyo para mantener este trabajo guiado hacía su voluntad y cumplir con un propósito de mi vida.

Agradezco a mi hermano menor, Miltón Adrián Córdova, por estar siempre presente en el camino, al creer siempre en mí, de lo que podía dar y sobre todo su cariño conmigo. Estar al pendiente de lo que hacía y como el proyecto avanzaba. También a mis padres, Miltón Córdova y Katty Santos, que formaron parte de este camino difícil para su apoyo incondicional y mantener su interés en los objetivos que tenían planteado para mi vida.

Agradezco de manera especial, a mi líder de Iglesia, Israel Toinga, que tuvo la sabiduría necesaria y su estima conmigo, para guiarme por el sendero de la resolución del proyecto, además de aportar su experiencia para construir un proyecto complicado con mayor rapidez.

Por último, agradezco a las personas Sasa Muisle y Juan Carlos Neira, personas que trabajan en la empresa, por su aporte de amabilidad y facilidad de comunicación, junto con su gran carisma para garantizar que el proyecto cumpla con las expectativas de la empresa para generar un gran valor.

Kevin Alexander Córdova Santos

ÍNDICE GENERAL

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

RESUMEN

El presente proyecto es para diseñar e implementar una aplicación móvil multiplataforma para la empresa Ferro Torre que tiene como objetivo ofrecer una facilidad en la comunicación con sus clientes, porque la tecnología en los dispositivos móviles aporta a la empresa un mayor alcance en su crecimiento y mejora de los productos que oferta a los clientes.

Para construir la aplicación móvil que satisfaga las necesidades de la empresa, se utilizó un prototipo o mockup que permita la validación de los requerimientos funcionales por parte del cliente, para evitar errores sobre la interfaz de usuario sobre el producto real. El diseño de la aplicación móvil está basado en la información de los productos que ofrece la empresa junto con sus características y los medios de comunicación, para incluirlos dentro la aplicación, con el objetivo de mantener los medios de comunicación tradicionales.

En el desarrollo de la aplicación, con un diseño previamente validado, se ha utilizado la metodología Scrum como marco de trabajo para facilitar la construcción y entrega constante hacía el cliente. Por consiguiente, esta aplicación contiene servicios para la administración de pedidos, cotización intuitiva por parte de los usuarios y la integración de diferentes medios de comunicación para facilitar el intercambio de información con el usuario que utiliza la aplicación.

Para concluir, se realizó pruebas de caja negra y caja blanca para validar los controles internos como externos de la aplicación para la entrega de un producto funcional que los usuarios puedan descargar de las tiendas oficiales, Google Play Store y Apple Store.

ABSTRACT

This project is to design and implement a multiplatform mobile application for the company Ferro Torre that aims to offer ease in communication with its customers, because technology in mobile devices gives the company a greater scope in its growth and improvement of the products it offers to customers.

To build the mobile application that meets the needs of the company, a prototype or mockup was used that allows the validation of the functional requirements by the client, to avoid errors on the user interface on the real product. The design of mobile application is based on the information of the products offered by the company together with their characteristics and the means of communication, to be included within the application, with the aim of maintaining the traditional means of communication.

In the development of the application, with a previously validated design, the Scrum methodology has been used as a framework to facilitate the construction and constant delivery to the client. Consequently, this application contains services for order management, intuitive quotation by users and the integration of different means of communication to facilitate the exchange of information with the user who uses the application.

To conclude, black box and white box tests were carried out to validate the internal and external controls of the application for the delivery of a functional product that users can download from the official stores, Google Play Store and Apple Store.

INTRODUCCIÓN

ANTECEDENTES

La empresa Ferro Torre, desde la fecha de su fundación en el año 1972, se ha ido renovando a fin de ofrecer un mayor abanico de productos a sus clientes. A pesar de su constante mejora en el producto, la relación con sus clientes se ve limitada por los medios tradicionales como lo son el contacto telefónico y el correo electrónico; ciertamente estos medios de comunicación son de gran utilidad para la empresa, sin embargo, surgen problemas tal como la saturación de llamadas y por otra parte la acumulación de correo, que limitan la rapidez en concretar una venta.

La tecnología móvil permite a las empresas tener un contacto más ameno con sus clientes puesto que las personas utilizan sus teléfonos para realizar diferentes acciones a lo largo de su día, a través de aplicaciones móviles que el usuario las adquiere según su necesidad. Esto permite a las empresas tener la capacidad de utilizar las aplicaciones móviles para ofrecer tanto sus servicios como sus productos a sus clientes de forma más amigable, sencilla y rápida.

La empresa maneja los diferentes procesos de gestión para cada una de sus actividades referenciados con tecnologías, para el área de ventas mantienen una página web, correo electrónico por cada vendedor y llamadas telefónicas para con sus clientes, que ha permitido estar a la vanguardia de las exigencias de la globalización y comunicación de todo el mundo.

DESCRIPCIÓN DEL PROBLEMA

La importancia de la tecnología con respecto a cómo los usuarios lo utilizan el día a día, y más cuando esta tecnología se puede incluir dentro de un dispositivo móvil, también conocido como Smartphone; supone una gran oportunidad que las empresas pueden utilizar para adquirir nuevos clientes e interactuar directamente con sus usuarios. A diferencia de los computadores, que son más costosos y tienen un tamaño considerable, los Smartphone son portables y su costo de fabricación conlleva al usuario pueda adquirirlos a un precio razonable directamente proporcional con su calidad, por lo que el desarrollo de aplicaciones involucradas para estos

dispositivos supone un valor agregado para que las empresas puedan interactuar con sus clientes y sus prospectos.

Actualmente la mayoría de las compañías utilizan las aplicaciones móviles para impulsar sus ventas o la comunicación con sus clientes para conseguir una ventaja competitiva y así llegar a generar un valor para ellos como para sus colaboradores. Los usuarios, por naturaleza, desean que la información se encuentre al instante y en tiempo real, siendo de importancia para tomar una decisión basado en la información que la empresa dispone o ya sea para consultas, llamadas directas, conversaciones por correo y que esto se encuentre al alcance de una aplicación móvil.

El valor que genera una aplicación móvil para la empresa Ferro Torre, que aportará a los usuarios en una mayor facilidad para comunicarse con la empresa y el proceso de cotización sea más simple para ellos, por ello se ha propuesto una aplicación móvil que se encuentre en los sistemas operativos móviles mayoritarios (Android & IOS) del mercado para abarcar a una gran cantidad de su clientela, para que pueda realizar cotizaciones y mantenga una mayor facilidad en su comunicación.

Las aplicaciones móviles son una herramienta que las empresas pueden utilizar por consiguiente para mejorar significativamente la comunicación con sus clientes, puesto que, pueden realizar diferentes acciones desde un mismo dispositivo y requerir pocos recursos tal como la conexión a Internet.

JUSTIFICACIÓN

La empresa Ferro Torre tiene en cuenta que sus productos pueden ser ofertados por los medios de comunicación tradicionales, aunque surge la necesidad de adaptarse a los nuevos cambios tecnológicos debido a los nuevos hábitos de sus clientes, por consiguiente, la importancia del desarrollo de una aplicación móvil a fin de ofrecer sus productos, realizar

cotizaciones sobre los mismos, asimismo permitir al cliente ser informado sobre el estado de los pedidos.

Debido a la gran variedad de dispositivos y sistemas operativos móviles, como lo son Android e IOS, el desarrollo de una aplicación móvil debe estar en ambos sistemas operativos con la intención de visualizar la información manejada por la empresa Ferro Torre con el fin de que pueda estar no solo en la mayoría de los dispositivos móviles sino también de forma uniforme. En Android, al ser un sistema operativo de código libre, para Javier Querol Morata (2011) "Su virtud reside en que es un sistema abierto, con lo que su evolución es muy rápida y está siendo apoyado por la comunidad." (p. 11). Su crecimiento es gracias a los aportes que la comunidad de desarrolladores del sistema, junto con la empresa Google Inc., aportaron una constante mejora en su performance, seguridad, interfaz y otros factores que afectan la experiencia del usuario con respecto al uso diario. El sistema operativo Android puede ser instalado en un teléfono inteligente, a excepción de un iPhone, para ser utilizado conforme al usuario y modificado para adaptarse a las necesidades del usuario, ya que, su customización en sus componentes lo convierten en un sistema operativo prácticamente moldeable para ser utilizado conforme a ciertas características de los fabricantes de teléfonos inteligentes. Para el sistema operativo iOS, creado por Apple, es un proveedor opuesto al ofrecido por Google.Inc, según lo explica Enrique Blasco Blanquer (2016) señala que "Apple cambió nuestra forma de pensar y ver los dispositivos móviles, nadie, ni en las mejores películas de ciencia ficción, podría haber imaginado cómo terminarían influenciado nuestras vidas estas máquinas." (p. 14); Blanquer expresa sobre como el sistema operativo creado por Apple cambio el paradigma de utilizar un teléfono inteligente, para convertirse, de un teléfono de llamadas, a un dispositivo que los usuarios puedan utilizar para diferentes tareas diarias, como enviar correos, tomar fotografías, enviar mensajes, navegar por internet, ubicación por GPS, entre otros. Para la empresa Ferro Torre resulta conveniente que la aplicación móvil se encuentre en ambos

sistemas operativos porque la mayor parte del mercado de aplicaciones se encuentran dentro de estas plataformas, además de que cada cliente puede tener diferente tipo de celular.

OBJETIVOS

Objetivo General

Diseñar e implementar una aplicación móvil multiplataforma que facilite la comunicación de la empresa Ferro Torre al ofrecer sus productos

Objetivos Específicos

- Diseñar una aplicación móvil que permita a los clientes de la empresa Ferro Torre visualizar los productos que oferta la empresa, realizar cotizaciones y observar el estado de sus pedidos.
- Mejorar la comunicación entre los vendedores y los clientes de tal manera que puedan realizar los pedidos de forma fácil y rápida.
- Desarrollar la aplicación móvil para su uso en los sistemas operativos móviles (Android - IOS).
- Implementar la aplicación móvil en las dos tiendas de aplicaciones, Google Play Store y la Apple Store.
- Realizar pruebas de funcionalidad utilizando emuladores de dispositivos móviles de los sistemas operativos Android e IOS.

MARCO METODOLÓGICO

El uso de una metodología para el desarrollo es la base fundamental de todo proyecto exitoso, pero no todas las metodologías de la actualidad se utilizan para construir esta clase de proyectos de bastante incertidumbre y mucho cambio constante; entonces surge la necesidad de que los proyectos de software deben tener una metodología que abarca su complejidad, por tal razón, surgen las metodologías agiles, que tienen en su naturaleza el cambio, mejora continua y tienen un manejo bueno sobre la incertidumbre.

En la época actual, existen varias metodologías donde todas tienen sus puntos negativos como positivos, por consiguiente, es necesario realizar un análisis sobre las ventajas que puede aportar un gran beneficio para la culminación con éxito del proyecto. A continuación, se indica un cuadro de comparación sobre las metodologías más importantes.

Tabla 1

Comparativa de metodologías ágiles

Características	**Scrum**	**RUP**	**XP**
Descripción breve	"Un marco de trabajo iterativo e incremental para el desarrollo de proyectos, productos y aplicaciones. Estructura el desarrollo en ciclos de trabajo llamados Sprint. Son iteraciones de 1 a 4 semanas, y se van sucediendo una detrás de otra." (Pérez O. A., 2011)	Es una metodología ágil basada en los diagramas UML para representar la estructura del producto de software.	"Es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, promoviendo el trabajo en equipo" (Roche & Suarez, 2009)
Tamaño del proyecto	Todo tamaño	Medianos y Grandes	Pequeños y medianos
Cambios	Se adapta un cambio en cada sprint.	Los cambios se analizan para después ser implementados.	Los cambios surgen y se implementen según haga falta (Respuesta rápida).

Tiempo	Existe un tiempo definido por cada sprint de 1 a 4 semanas como máximo	Una serie de ciclos sin tiempo definido	Un conjunto de fases iterativas con el cliente de entregas rápidas.
Comunicación	El equipo debe ser autoorganizado.	Comunicación con todas las personas que integran el proyecto.	Comunicación continua con el equipo y el usuario
Documentación	Alto	Alto	Bajo
Criterio de selección	SELECCIONADO Mantiene un flujo de trabajo muy adaptado a las iteraciones y cambios que puedan existir en el proceso de desarrollo. La entrega del producto se realiza de forma constante por pequeñas partes de la aplicación	En cada ciclo de iteración se requiere repetir la mayor parte de las fases que incorpora.	Una metodología que lleva al usuario en todo su ciclo de desarrollo y que se dedica exclusivamente al desarrollo mas no a la documentación.

Nota. Tabla comparativa de Metodologías agiles con sus características.

Elaborado por: El autor.

En consecuencia, del análisis comparativo realizado del cuadro anterior, se puede considerar a Scrum como marco de trabajo para construir la aplicación móvil, puesto que su flujo de iteraciones está basado en la entrega de productos pequeños y que estos contienen un feedback por parte del cliente o dueño del producto. De esta manera, se desarrolla la aplicación en versiones pequeñas donde el cliente es un participe del proceso de construcción; además no está limitada a pasos rígidos a seguir, más bien, mantiene un ciclo de incremento constante en base a la organización propia del desarrollador y así agilizar el proceso de gestión de proyecto.

CAPITULO I
MARCO TEÓRICO

1.1. MARCO INSTITUCIONAL

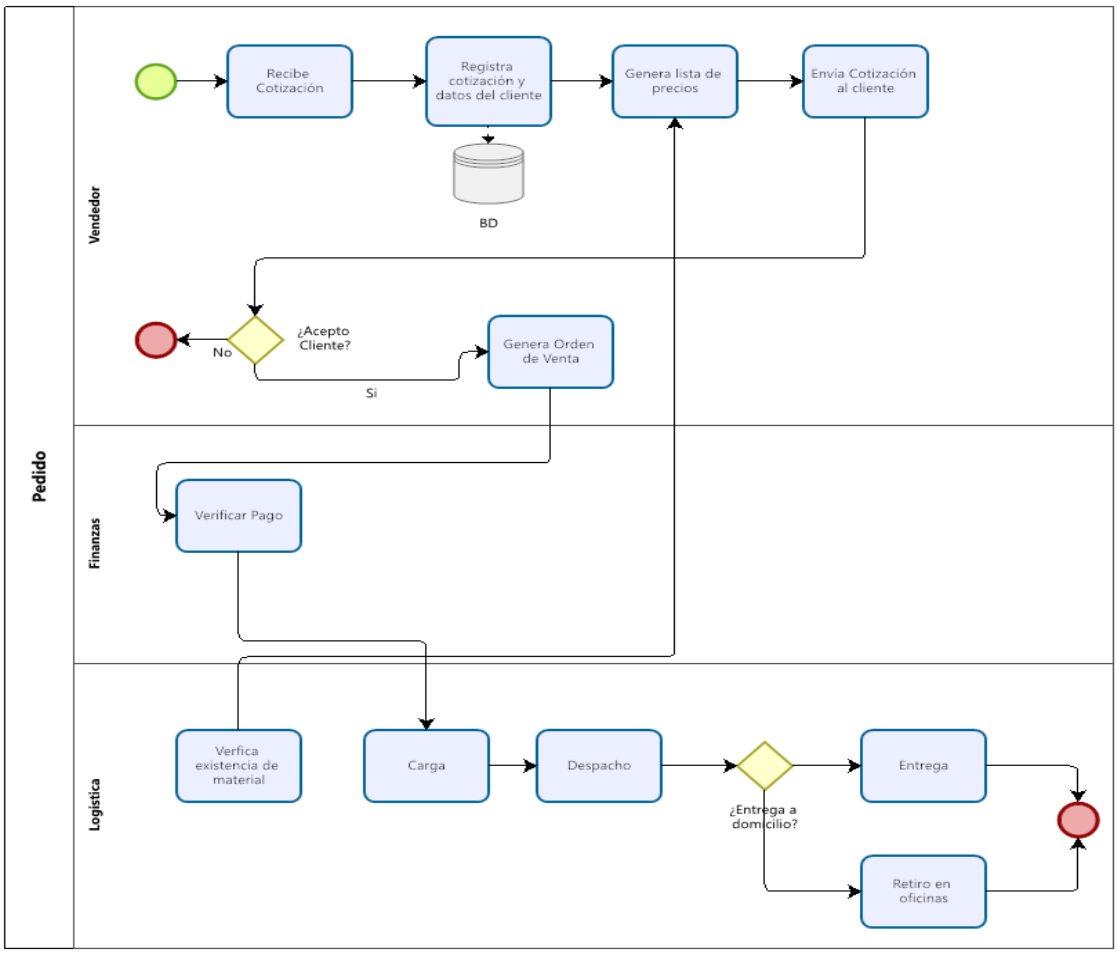
La empresa Ferro Torre es una empresa ecuatoriana que se dedica a la fabricación y comercialización de materiales de acero, que utiliza tecnología actual para la mejora continua en cada uno de sus procesos, de acuerdo con las exigencias de un mercado especializado. A través de la visión y el esfuerzo de Antonio Ferro Torre permite su creación en el año de 1972 con el objetivo de comercializar productos de acero, desde ese año se empieza un cambio constante para seguir agregando valor con la fabricación de productos de acero, tales como perfiles, tuberías, bobinas, etc. (Ferro torre, 2021).

Diagrama de Procesos

En el grafico 1, se puede observar que a partir de una conversación transmitida por una de las personas del área de ventas y/o cotizaciones, se indica el proceso de un pedido que se maneja dentro de la empresa, desde la petición de cotización por parte del usuario hasta la entrega y/o retiro de productos (gestionado por diferentes áreas), el diagrama de procesos esta realizado con el software gratuito denominado Bizagi Modeler.

Figura 1

Diagrama actual de pedido



Nota. Diagrama de pedido desde la cotización por parte del cliente hasta la entrega del producto.

Elaborado por: El autor

1.2. ANTECEDENTES CONCEPTUALES

La tenencia de una aplicación móvil dentro de una empresa implica que la empresa tiene en cuenta que la tecnología móvil es el futuro para lograr conseguir más clientes, que, a su vez, puedan alcanzar a que sus procesos se automaticen. El desarrollo de una aplicación móvil tiene la limitante de que funcionara solo en los sistemas operativos por la que fue desarrollado, es decir, una aplicación desarrollada en IOS no funcionará en un sistema operativo Android, porque su arquitectura es totalmente diferente. De este concepto de incompatibilidad entre

sistemas, surgen las aplicaciones móviles multiplataforma que permiten al desarrollador realizar un trabajo en ambas plataformas, porque los usuarios demandan esta clase de apps por la gran cantidad de dispositivos móviles en el mercado, tal como menciona Jefferson Torres en su trabajo de titulación lo siguiente “En la actualidad, el uso de dispositivos móviles como medio de comunicación y su aplicabilidad como herramienta de trabajo, es relevante, originando así una gran demanda en el uso de aplicaciones multiplataforma” (Torres, 2021).

Los teléfonos inteligentes existen en gran variedad, con diferentes tamaños, velocidad, resolución de cámara, micrófono, etc. Aunque muchos de ellos, comparten la característica de tener un sistema operativo definido para que este interactúe con el hardware del dispositivo, lo que facilita la interacción por parte del usuario con el teléfono, esto se logra a partir de aplicaciones integradas en el sistema operativo donde cada una tiene un objetivo específico para satisfacer esta interacción con el hardware e incluyen algunos elementos para relacionarse con el software. Este tipo de aplicaciones son parte nativa del sistema operativo que proveen funcionalidades básicas para cubrir algunas necesidades que un usuario requiere, por tal motivo, cada sistema creo una tienda de aplicaciones móviles donde los desarrolladores pueden subir y distribuir sus propias aplicaciones para un complemento de valor que el usuario adquiere en su teléfono inteligente.

Las tiendas de aplicaciones móviles, como lo son Google Play store y App Store, permiten a los desarrolladores publicar aplicaciones en base a los sistemas operativos que albergan estas tiendas, tal que, el desarrollo de una aplicación móvil debe estar sujeto a una tienda en específico para su publicación, como lo es, el tipo de archivos que se suben, precios, lenguajes de programación aceptados, etc. Estas restricciones no han limitado la creatividad de los desarrolladores que han creado un enorme conjunto de tecnologías para construir una aplicación móvil sin la necesidad de utilizar un lineamiento técnico por parte de la tienda de aplicaciones móviles; de esta forma surgen las aplicaciones nativas, hibridas, web y

multiplataformas para satisfacer ciertas necesidades técnicas que pueden surgir en una aplicación.

La disrupción entre los dos sistemas operativos móviles (Android - iOS) ha conllevado a los desarrolladores tomar una decisión de utilizar cierta tecnología para solo un determinado sistema operativo, y si fuere el caso, publicar la misma aplicación móvil construida en una determinada tecnología llevarla hacía el otro sistema es algo sumamente complicado por la incompatibilidad que existen entre ambos. Aunque, gracias a la gran creatividad que reflejan los desarrolladores para solventar este problema, en el mercado existen una enorme cantidad de tecnologías para el desarrollo de aplicaciones que funcionan en ambos sistemas operativos, solo utilizando el mismo código. De forma visual, el usuario no detecta cambios utilizando cualquiera de los dos sistemas operativos, pero en la parte interna de la aplicación ocurren un conjunto de proceso, que dependiendo de la tecnología con la que se desarrolló, pueden utilizar tecnología web llevado al entorno móvil (Web), código nativo con componentes web (Hibridas) o los más modernos utilizar código nativo para ambas plataformas a través de un mismo intermediario (Multiplataforma).

1.3. FUNDAMENTOS TEÓRICOS

1.3.1. Sistemas operativos móviles

En el mercado actual[1] existen dos sistemas operativos predominantes, que son, Android por parte de Google e IOS por parte de Apple, que en su arquitectura interna son completamente diferentes uno del otro, a lo que sus aplicaciones móviles se desarrollan en uno de estos sistemas operativos para traer características únicas de cada uno; a estas aplicaciones se las denomina nativas.

Android. Es un sistema operativo basado en Linux, que tiene una gran acogida por parte de los usuarios y su tienda de aplicaciones denominado Google Play Store, donde la muchas de

[1] Referencia al año 2021

ellas son de índole gratito, que además según las normas de la tienda, todas son de código abierto, porque Linux en su código fuente es de carácter libre. La arquitectura de Android está adaptada al hardware de los teléfonos inteligentes con pantalla táctil, donde el trabajo en conjunto de Android, el núcleo de Linux y algunos controladores del teléfono da la posibilidad de interactuar con sus componentes tales como, cámara, GPS, botones, etc. para ser llevado a cada dispositivo donde se ejecute el sistema operativo Android.

Apple IOS. El sistema operativo con nombre IOS fue lanzado al mercado, por la empresa Apple en el año 2007, que se utilizan exclusivamente en los teléfonos iPhone, por ser un sistema operativo privativo Apple no permite la instalación o ejecución de su sistema operativo en otros teléfonos inteligentes. Un teléfono iPhone tiene la mayoría de las características de otro teléfono inteligente como lo es el GPS, cámara, giroscopio, etc. Las aplicaciones móviles dentro de este sistema operativo se pueden descargar desde la tienda oficial, denominada App Store; y como sistema privativo, Apple no permite instalar aplicaciones fuera de su tienda oficial.

1.3.2. Aplicaciones nativas

El desarrollo de aplicaciones móviles desde el nacimiento con su rápido crecimiento de las tiendas de aplicaciones, como lo son Google Play y App Store, ha llevado a un continuo crecimiento de la cantidad de aplicaciones disponibles en ambas plataformas. En caso anverso de su crecimiento, ha llevado a encontrarse en sistemas operativos totalmente diferentes, incompatibles entre ellos y administrados por empresas distintas (Google Inc., Apple Inc.); esta disrupción de los dos sistemas operativos ha conllevado, a que, cada uno tenga su propio lenguaje de programación, guías de diseño de UI y librerías, sin tener la posibilidad de compatibilidad en otro sistema operativo. A estas aplicaciones móviles construidas exclusivamente para un sistema operativo, son denominadas <nativas> por referencia de su desarrollo está orientado al conjunto de herramientas de desarrollo dedicados a un sistema operativo.

Para el desarrollo de aplicaciones móviles en el entorno de Android, se encuentran los lenguajes de programación Java y Kotlin, con una guía de diseño, recomendada por Google, llamada Material Design y librerías que son exclusivas para ambos lenguajes.

Por otro lado, en el desarrollo de aplicaciones móviles en el entorno de IOS, se utiliza el lenguaje Swift, que es una evolución del lenguaje Objective-C. Para la guía de diseño de interfaz de usuario, Apple creo la denominada "Human Interface Guidelines" para utilizarla, no solo en aplicaciones móviles, en las plataformas que Apple tiene en el mercado (MacOS, tvOS, WatchOS).

1.3.3. Aplicaciones hibridas

Las aplicaciones hibridas utilizan componentes del entorno web junto con el código nativo, en claridad de esta definición sobre este tipo de aplicaciones lo da Jefferson Torres (2021):

Una aplicación híbrida combina aspectos de las aplicaciones nativas y de las aplicaciones web, desarrolladas bajo HTML (…), CSS (…), y Java Script, mismas que se apoyan sobre una capa de abstracción (Framework) para tener acceso a los recursos del equipo móvil, logrando así tener una apariencia visual y funcional como de una aplicación nativa. (p. 8)

El autor hace referencia a los lenguajes de programación utilizados en el entorno web para construir aplicaciones web y la integración que ocurre dentro de una aplicación móvil, para que, en conjunto de código nativo interactuar con el hardware del dispositivo y obtener esa característica de aplicación "nativa".

En este tipo de aplicaciones hibridas su conexión con las capacidades del teléfono se lo realiza a través de una gran cantidad de librerías o plugins, que dan mayor agilidad en el desarrollo de estas; algunas de estas aplicaciones se pueden reutilizar en un entorno web lo que facilita la portabilidad de una aplicación móvil a convertirse en otra plataforma.

1.3.4. Multiplataforma

Los nuevos desarrollos tecnológicos para el entorno móvil han crecido de forma extraordinaria en cuanto a la variedad de celulares, diferentes sistemas operativos, su precio, etc. Da un gran obstáculo a los desarrolladores porque les impide que la aplicación desarrollada no funciona en un celular que tiene gran tendencia en el mercado o llegar a todos los usuarios deseados, por tal motivo, las aplicaciones multiplataforma o hibridas ofrecen una gran solución para solventar este problema para los desarrolladores. En el mercado de lenguajes de programación, frameworks [2] es muy variado y donde cada una de ellas tiene su propia comunidad de desarrolladores en donde aportan soluciones, crecimiento y soporte para tal plataforma de desarrollo, entre algunas que se pueden mencionar dos importantes:

- Xamarin**:** Es un framework de desarrollo de aplicaciones móviles, desarrollado por Microsoft, para construir aplicaciones nativas para Android, IOS y la, ya casi desaparecida, Windos Phone; que tiene su desarrollo en el lenguaje de C#. Lisandro Delía (2017) describe algunas otras características según lo informa lo siguiente "Las aplicaciones escritas con Xamarin y C# disponen de acceso completo a las API de la plataforma subyacente, así como de la capacidad de crear interfaces de usuario nativas y de realizar compilación en código nativo" (pág. 46).
- Ionic**:** Es un framework de código abierto para el desarrollo de aplicaciones móviles hibridas con carácter multiplataforma, utilizando un conjunto de lenguajes de programación como HTML5, CSS y JavaScript para crear las interfaces de usuario, para que se puedan utilizar en los sistemas operativos como Android o IOS.
- SDK-Flutter**:** Es un kit de desarrollo para el desarrollo de aplicaciones móviles, para las plataformas de Android y iOS, que propone un flujo de construcción relativamente rápido

[2] Es un entorno de desarrollo que facilitan el trabajo para desarrollar grandes características de un sistema de información completo, que ya se incluyen solo para ser reutilizadas y así mejorar la experiencia al momento de desarrollo.

y que permite interactuar con el hardware del dispositivo sin mayores complicaciones. Está basado en el lenguaje de programación Dart y puede ser fácilmente descargado de su página oficial sin requerir instalaciones pesadas para ser ejecutados en diferentes editores de código, como lo son Android Studio o Visual Studio Code.

1.3.5. Json

Json (JavaScript Object Notation) es un formato de fácil lectura para almacenar datos y la transferencia de información, es utilizado para almacenar datos que se pueden enviar en grandes tamaños a los diferentes por su simplicidad al construirlos.

Json surge por la necesidad de suplir a las bases de datos relacionales, que tienen una estructura muy cerrada para realizar modificaciones en su información y conforme su escalamiento es más alto tiende a convertirse lento, por tal razón, las bases de datos no relacionales tienen un auge importante en la industria de la tecnología que junto a json como formato puede ser de mucha ayuda para las aplicaciones modernas, "nacen como una solución para aquellas aplicaciones que demandan el manejo de gran cantidad de información… también alguna de estas maneja datos en tiempo real como observables de los datos" (Torres, 2021), características que la aplicación móvil a desarrollar pueda necesitar conforme la empresa Ferro Torre va creciendo y permite el escalamiento de forma ordenada.

1.3.6. Figma

Figma es un software que permite la creación de prototipos y diseño de interfaces de usuario que se ejecuta en el navegador o en un programa destinado para varios sistemas Operativos, esta herramienta es una de las más importantes dentro del todo el proceso de desarrollo de la aplicación móvil, porque permite validar, en colaboración con el cliente, la etapa del diseño. Además de su aporte de valor para la colaboración, Figma permite al diseñador ver código para ciertos lenguajes de programación y facilitar así el paso de un diseño "dibujado" a código para así simplificar el trabajo.

La importancia de tener un prototipo de la interfaz de usuario validado por la empresa Ferro Torre es clave para desarrollar la parte interna (backend) de la aplicación y acelerar el desarrollo de la aplicación móvil. Otra ventaja de Figma con respecto a otros programas de diseño, al tener la capacidad de que las pantallas, botones, imágenes y otros componentes, puedan ser interactivos al usuario, es decir, el usuario puede presionar sobre ellos y estos realizaran cierta acción que anteriormente se configuro.

Durante el ciclo iterativo de la solución tecnológica surge la posibilidad sobre lo que el usuario piensa sobre el diseño realizado, por tanto, Figma tiene esta peculiaridad en su proceso de diseño porque permite a los usuarios o diseñadores, observar los cambios en tiempo real por su inherente propiedad de colaboración. Los diseñadores y desarrolladores tienen una retroalimentación en el paso que se construye la interacción gráfica de las pantallas para con los usuarios, de manera que, el software Figma provee unas pequeñas animaciones para hacer una presentación con bastante fluidez para que el usuario pueda tener un claro panorama de la interfaz de usuario que se va a desarrollar en código. La persona que diseña las pantallas de la aplicación tiene en cuenta que el producto de software va a ser construido en código para convertirse en un producto real y usable por el usuario, de ahí que, Figma provea herramientas como creación de cuadrados, texto, vectores, entre otros; para complementar el diseño de interacción completo y facilite el trabajo para la persona que codifique la interfaz.

Una aplicación móvil está sometida a diferentes tipos de pantalla por la variedad de dispositivos que se encuentran en el mercado, la persona que fabrica la interfaz gráfica de usuario está obligada a esbozar en base a los diferentes tipos de pantallas existentes, a este proceso se lo denomina, diseño responsivo, con el propósito de que la interfaz de usuarios se adapte en su totalidad a la pantalla del dispositivo que pueda tener un usuario común; el software Figma provee a los diseñadores la capacidad de solventar un problema común en todos los dispositivos móviles, como lo son el Padding, Auto Layout, Resizing, entre otros. En el

entorno para dibujar las pantallas se encuentran lienzos predeterminados de tamaño de un celular real, con el objetivo de que para la persona que diseña las pantallas le resulte más más sencillo adaptarse a ciertos modelos de celulares, sin perder la propiedad de responsivo.

Con un prototipo validado el camino del desarrollo se convierte en un ambiente más controlable y con pocos cambios, algo muy difícil de conseguir en el ámbito de desarrollo tecnológico, por tal motivo, el diseño de la interfaz de usuario (UI) permite al desarrollador tener una alta concentración en la generación de código y de una aplicación móvil funcional, para que durante el proceso surjan cambios a nivel de arquitectura que no afecten a los tiempos del proyecto.

1.3.7. SDK Flutter

Fue creado por la empresa Google Inc., con el objetivo de crear aplicaciones móviles con mayor rapidez y que su compilación se haga de forma nativa. Está basado en el lenguaje de programación denominado Dart, la característica más importante es lo que menciona (Daniel, 2020) en su tesis, “Una de las ventajas de utilizar Flutter es que permite ver los cambios de interfaz gráfico a tiempo real” (pág. 14). Por tal motivo, el desarrollo tiene mayor rapidez para su salida al mercado de aplicaciones.

La construcción de aplicaciones móviles para los dos sistemas operativos móviles, IOS y/o Android, ha sido de interés para grandes compañías, entre ellas, Google, que ha creado un kit de desarrollo (SDK) llamado Flutter, que según Villegas y Loor (2020) lo describen así “una forma sencilla de desarrollar aplicaciones móviles, ha empaquetados las soluciones de interfaz más comunes como navegación, botones, animaciones, tipografías, iconos, etc. En widgets que pueden reutilizarse para prototipar muy rápido” (pág. 26). El término “sencillo” tiene un gran aportador de valor para este kit de desarrollo, porque genera un desarrollo rápido y en tiempo real en el aplicativo al momento de realizar cualquier cambio, tal como ocurre en la construcción de sitios web.

Google ha creado Flutter con el lenguaje de programación Dart, que en la página oficial (flutter.dev) permite a los widgets tener todos los elementos interactivos dentro de una aplicación móvil, tales como, iconos, imágenes, navegación, paginas, etc.; que se compilan nativamente, es decir, que se recopilan en lenguaje de máquina nativo de los sistemas operativos donde es ejecutado la aplicación. Es importante mencionar, que este SDK es de código fuente abierto, para ser cambiado con el objetivo de constante mejoras en su performance, codificación, librerías, etc.

Para la construcción de las aplicaciones móviles, en la parte visual, Flutter tiene unos componentes denominados Widgets, que es cualquier objeto visual mostrado en la pantalla del dispositivo y donde el usuario interactúa para realizar ciertas acciones. Flutter tiene una gran cantidad de Widgets en su documentación oficial, dando lugar a que los desarrolladores puedan utilizarlos de forma rápida; además de tener una gran compatibilidad entre ellos, por lo que, cada componente creado en los archivos del lenguaje Dart puede ser utilizado en otra pantalla o incluso dentro de un mismo Widget. En el proceso de desarrollo, su lenguaje de programación permite importar estos componentes en sus archivos con extensión (.dart) sin la necesidad de configuraciones preliminares donde la estructura de organización depende de cada desarrollador. El equipo de desarrollo de Flutter ha desarrollado estos componentes visuales para cada tipo de sistema operativo móvil, es decir, dentro del ambiente de Flutter se pueden encontrar objetos visuales únicos de un determinado sistema operativo, por consiguiente una aplicación móvil desarrollada en este ambiente puede contener objetos visuales de un sistema operativo en específico y funcionar del mismo modo que en otro sistema operativo, por ejemplo, una aplicación desarrollada en este SDK que se ejecuta en el sistema operativo Android con sus componentes basados en Material Design, deberá observarse igual en el sistema operativo iOS (basado en Human Design).

Durante el proceso de desarrollo es necesario validar la codificación realizada, puesto que, las personas que crean la aplicación móvil tienden a cometer errores, por esta razón Flutter ha realizado una abstracción del desarrollo que se realiza para la web con el propósito de observar los cambios realizados en código son reflejados directamente en la pantalla del dispositivo. Esta potencial característica da la capacidad de codificación con mayor rapidez porque la compilación del código en desarrollo se lo realiza al instante, asimismo Flutter tiene diferentes formas de recargar o compilar la aplicación para observar los cambios realizados, de manera que tiene tres tipos de compilación para reflejar los cambios, se detallan a continuación:

- Hot reload**:** Carga los cambios realizados en el código y construye nuevamente la estructura de los Widgets, manteniendo el estado de la aplicación. (Flutter, 2021)
- Hot restart**:** Carga los cambios realizados en el código y reinicia la aplicación, donde se dibujan nuevamente los Widgets al igual que el estado. (Flutter, 2021)
- Full restart**:** Los cambios realizados en el código se lo cargan en un reinicio total de compilación del ambiente de desarrollo y su aplicación. Este proceso toma los archivos nativos de cada sistema operativo para ejecutar la compilación. (Flutter, 2021)

Por tanto, el método de compilación, recarga en caliente (hot reload) es el más adecuado para observar cambios al instante, donde solo se efectúa un cambio en la estructura de los Widgets además de cualquier información mostrada, por ejemplo, de una base de datos, se mantiene.

El rendimiento es una propiedad que tienen las aplicaciones móviles nativas, porque la compilación de su código se ejecuta en el sistema operativo base. Flutter posee esta característica al implementar en su desarrollo el código nativo de cada plataforma por lo que lleva a que, tanto sus widgets, iconos y fuentes, están basados en los sistemas operativos que soporta (Android & iOS), por lo tanto, su rendimiento en comparación con las aplicaciones nativas no tiende a ser muy diferencial. Del mismo modo que una aplicación móvil nativa se

compila directamente en el procesador del dispositivo, abstraído por el sistema operativo, Flutter realiza la misma tarea porque tiene incrustado el código de cada plataforma en su proyecto, por esta razón, su similitud en el rendimiento sigue siendo muy equitativa.

La gran cantidad de frameworks, SDK y lenguajes de programación para el desarrollo de aplicaciones móviles multiplataforma es muy variada, donde cada uno tiene tanto sus ventajas como desventajas para el desarrollador, entre todas se ha decidió utilizar la tecnología Flutter por su gran rapidez al momento de escribir código y su kit de desarrollo muy completo para la parte de la interfaz gráfica; el lenguaje de programación, Dart, permite codificar con mucha rapidez y la conexión entre archivos muy eficiente lo convierte en un lenguaje muy útil.

Finalmente, la comunidad de desarrolladores que mantienen actualizado los cambios constantes sobre el SDK ofrece la posibilidad de encontrar toda la información sobre nuevas librerías, solución de errores y apoyo para continuar con la mejora de Flutter. Esta es la base de cualquier lenguaje de programación, frameworks o nueva tecnología que surge, porque durante el proceso de desarrollo que tiene una persona, la posibilidad a cometer errores es muy alta, por tanto, el apoyo de la comunidad para aportar nuevas ideas, elaborar librerías y consejos, sigue siendo fundamental para el crecimiento constante de Flutter como un SDK de desarrollo para aplicaciones móviles multiplataforma.

1.3.8. Firebase

Es un servicio en la nube creado por Google para proveer a los desarrolladores una mayor facilidad y rapidez al crear una infraestructura de una aplicación móvil (también compatible en el entorno web). Esta plataforma utiliza la infraestructura de Google para proveer su servicio en una capa gratuita para apoyar a las nuevas aplicaciones en su surgimiento y facilitar su escalabilidad en el transcurso del tiempo. Firebase tiene en su infraestructura una gran cantidad de servicios que permiten a los creadores de aplicaciones móviles construir una aplicación, publicarla y después analizar su crecimiento. Entre las variadas soluciones que provee Firebase, se encuentran las bases de datos (Cloud FireStore, Real time Database),

analytics (Google Analytics), usuarios (Firebase Authentication), hosting, entre otros. Sus productos se encuentran enmarcados en tres procesos una aplicación sigue para salir al mercado:

- Construcción**:** En el contenedor de construcción se encuentran todos los productos que estan basados en aportar un gran valor en el proceso de desarrollo de una aplicación, facilitando una base de datos online, análisis de errores, manejo de usuarios, entre otros. (Firebase, 2021)
- Publicación y Monitoreo: En este contenedor se encuentran los productos donde participan la publicación del programa en el mercado de aplicaciones junto con un monitoreo después de su lanzamiento. En este conjunto se encuentran herramientas como análisis de las tiendas, rendimiento en cada dispositivo, facilidad para distribuir aplicaciones, detección de errores de producción. (Firebase, 2021)Escalabilidad: Por último, el contenedor que permite a los desarrolladores aprender de sus usuarios para impartir una mayor experiencia de usuario (UX) e incrementar el éxito de la aplicación dentro del mercado, provee productos de aprendizaje de maquina (Machine learning), predicciones, comunicación de mensajes, entre otros. (Firebase, 2021).

Los servicios en la nube que ofrece Firebase tiene un valor intrínseco durante el proceso de desarrollo de una aplicación informática, facilitar el flujo de usuario para mejorar la experiencia de usuario durante el inicio de creación de una aplicación implica una solución de otros problemas futuros cuando la aplicación sale al mercado. Después del lanzamiento al mercado es necesario que el usuario obtenga actualizaciones constantes para permitir que la aplicación vaya creciendo conforme a la experiencia del usuario y aprender sobre las diferentes acciones que realiza un usuario, su cultura, idioma o el uso único que el usuario tiene para una aplicación, con las herramientas de Firebase para dar un mayor porcentaje a la seguridad en cada actualización que se lance al mercado y el usuario pueda adaptarse a estos nuevos cambios

sobre la aplicación que usa. La arquitectura que tiene Firebase da una mayor confianza al creador de aplicaciones porque no es necesario configurar parámetros complejos, archivos pesados y es compatible en varios lenguajes de programación con librerías ya desarrolladas.

Las aplicaciones multiplataformas están dirigidas a las diferentes plataformas y que su arquitectura se adapte a las necesidades de cada sistema operativo móvil, para Firebase esta adaptación significa un gran valor para los desarrolladores porque provee herramientas de desarrollo para implementarlas en la aplicación a crear, para cuando, se utilize por los usuarios puedan obtener la misma funcionalidad sin importar el sistema operativo móvil base y facilitar a las aplicaciones multiplataformas para que este servicio se encuentren equitativamente para los usuarios, independiente del dispositivo que utilizan.

Los servicios de Firebase proveen datos importantes para los creadores de software, para que, en base a los datos obtenidos por la plataforma, para empujar su crecimiento exponencialmente y que guste a los usuarios. El conocimiento de cada evento que se ejecuta por un usuario dentro de la aplicación permite al desarrollador crear una mejor experiencia de usuario para una próxima actualización, para así, incrementar su crecimiento y popularidad con el público. Esta información puede ser segmentada por cada conjunto de usuarios para facilitar ciertos cambios de una aplicación en base a diferentes regiones y crear una aplicación que provea mayor accesibilidad para un conjunto determinado de usuarios que se les dificulte el uso de la aplicación por sus condiciones son diferentes a otro conjunto de usuarios.

Firebase Authentication

Una de las herramientas que contiene la plataforma de Firebase, es la de Authentication, que básicamente, permite al usuario registrarse en una aplicación para utilizarla y al desarrollador controlar que usuarios están utilizando la aplicación, junto con su forma de ingreso. En Firebase Authentication, el registro de un usuario es un proceso complicado para el desarrollador y por tal motivo, ha creado esta herramienta para facilitar el ingreso de usuario

para utilizar una aplicación que, por ejemplo, requiera de un registro para realizar una compra en una aplicación de E-commerce. Los diferentes servidores de correos tienen una diferencia sustancial para ser compatibles entre ellos, por su diversidad en tecnología, pero Firebase Authentication ha realizado esta fusión de la variedad de servicios de correo electrónico para que sean gestionados por una misma plataforma, en este caso, Firebase en conjunto.

Puesto que, los usuarios tienen una forma única seleccionar un servicio de correo electrónico, ya sea, por sus gustos o facilidades de uso; Firebase Authentication tiene una importante variedad de servicios de correo en su arquitectura, tal como, Google Sign In, Apple Sign In, correo electrónico genérico, Github, Twitter, entre otros. Para una aplicación móvil que posee las diferentes formas de autenticación como funcionalidad, tiene un valor añadido para el usuario porque le permite la facilidad uso en base a los gustos que el consumidor de aplicaciones suele utilizar y que estos servicios de correo electrónico estén estrechamente interconectados con la aplicación. La construcción de las pantallas para cada servicio de correo electrónico es una tarea tediosa para el desarrollador, por lo que, Firebase Authentication utiliza las diferentes formas de autenticación de cada proveedor para aprovisionar una vista de usuario sin la necesidad de una codificación por parte del desarrollador de la aplicación.

El gestionar las contraseñas es un tema vital para cualquier aplicación que utiliza registro e ingreso de usuarios como una funcionalidad, de ahí que, esta herramienta provea una seguridad para las mismas utilizando su arquitectura para almacenarlas. Del mismo modo, que las guarda en su base de datos, provee una mayor seguridad que una integración realizada por el desarrollador de la aplicación. Las contraseñas se guardan de forma seguridad en el entorno protegido por Firebase Authentication, es decir, que el desarrollador solo debe interesarse en la implementación de este servicio para con sus usuarios sin el riesgo de vulnerabilidades que conlleva un inicio de sesión común. Esta información ingresada por el usuario para identificarse dentro de la aplicación se guarda en el ambiente de Firebase, de ahí que, el creador del software

solo pueda gestionar las diferentes cuentas que se crearon dentro de la aplicación, por tanto, es una alternativa cómoda para los usuarios (que desean evitar procesos de autenticación tediosos) y para el desarrollador (donde la construcción de aplicaciones sea más fácil de desarrollar).

Por otra parte, este servicio tiene una gran variedad de proveedores de servicios de correo electrónico implementados en su arquitectura, otro aspecto que este servicio tiene para facilitar, aún más, el desarrollo de aplicaciones es la identificación de usuarios por otras formas de autenticación como lo son los números de teléfono y una autenticación anónima (permite al usuario ingresar a la aplicación sin la necesidad de registrarse, pero durante un periodo corto de tiempo). Firebase Authentication proporciona estas formas de identificar a un usuario dentro de la aplicación porque cada usuario tiene sus preferencias, de acuerdo, a como el consumidor utiliza el dispositivo móvil dentro de la rutina diaria. Del mismo modo, que la plataforma permite la creación e identificación de los usuarios, abastece de varios kits de desarrollos relacionados con la autenticación, tal como, el cambio de contraseña que da la facultad al usuario de realizar cambios en su configuración de identificación dentro de la aplicación para que se adapte a las nuevas necesidades que el usuario pueda tener; verificar los correos electrónicos es otra herramienta que ofrece este servicio, en especial a los desarrolladores, porque permite validar a los usuarios que utilizan la aplicación son personas reales y no maquinas automatizadas que intentan destruir el flujo de la aplicación u obtener datos importantes de una aplicación.

Este producto, del catálogo de Firebase, aporta un gran valor para la aplicación móvil a desarrollar porque permite la creación de usuarios y gestión de los mismos sin configuraciones adicionales, además de permitir a la aplicación tener una mayor confianza para la persona que lo utiliza porque su cuenta no se ve afectada por factores externos, como lo es la seguridad o la confidencialidad de los datos, ya que, los datos del usuario son gestionados por Firebase Authentication y almacenados dentro de la arquitectura de Firebase, para lo cual, solo el

desarrollo puede acceder a ciertos datos de la cuenta para ser utilizados en el módulo de roles de usuario junto con la gestión de pedidos por usuario.

Finalmente, el servicio de autenticación de usuarios provisto por Firebase es parte de una capa gratuita que tiene esta plataforma para los creadores de aplicaciones, por eso, es una excelente solución para personas que se encuentran en etapas tempranas de una aplicación móvil y aspiren a un crecimiento sustancial con el paso del tiempo porque este servicio tiene la característica de la facilidad de implementación, por el contrario, un servicio de autenticación personalizada requiere de una gran cantidad de personas para construirla, mantenerla y escalarla conforme los nuevos requerimientos tecnológicos que puedan surgir.

Firebase Cloud Firestore

Para almacenar la base de datos, es requerido una aplicación móvil que pueda estar conectado a Internet y que el almacenamiento de información se encuentre activo en todo el ciclo de vida de la aplicación, por tal motivo, Firebase Cloud Firestore es una solución de base de datos online que permite guardar información y que este sea fácilmente escalable con el tiempo, además de mantener una base de datos constantemente en línea para consumo de la aplicación, este producto tiene la característica de cambios con muy baja latencia, en otras palabras, la información que se cambia en un lugar se ve reflejada en todos los que consumen el servicio con un corto tiempo de espera.

A diferencia de una base de datos tradicional o relacional, que tienen una estructura estática para cada entidad de información y cualquier cambio en la base de datos requiere de un análisis para realizarlo, por lo que, Firebase Cloud Firestore es una base de datos NoSQL que permite la creación de estructuras dinámicas para cada entidad de información, puesto que, permite la escalabilidad cuando existan nuevos requerimientos y se puedan actualizar las características hacía las exigencias de las características que aparezcan. Con el propósito de que la aplicación salga al mercado pueda adaptarse fácilmente con las nuevas exigencias de los

usuarios y mantener un esquema para cada tipo de usuario según la necesidad que pueda tener el cliente que utilice la aplicación.

La complejidad que conlleva construir y levantar un servidor es conocido por varios desarrolladores que pueden o no tener conocimiento sobre este tema de servicios, por consiguiente, este producto de Firebase ofrece a las personas que desarrollan aplicaciones una oportunidad de desplegar el software hacía los usuarios sin tomar mucho tiempo de configuración para la base de datos y que este software se encuentre en línea en todo momento. Para la aplicación, estar en línea es un valor añadido que un usuario aprecia porque permite estar siempre actualizado sobre la información que la aplicación le pueda ofrecer, para así, lograr que el usuario utilice con mayor frecuencia a la aplicación, es decir, conlleva a una mayor interacción del usuario para ofertar una mejor calidad en la provisión de la información.

La globalización ha influido en que los usuarios requieran que la información se encuentre disponible al instante que lo soliciten, por tal motivo, este producto mantiene esta exigencia en sus características debido a su arquitectura que permite realizar cambios en un dispositivo y observarse reflejado en otro dispositivo lejano con unos milisegundos de tardanza. Para ello, un usuario que realice cambios sobre un determinado campo que otro usuario observar, es convierte en una característica fundamental para complementar las exigencias de los clientes hiperconectados en el mundo del Internet.

En el catálogo de productos de Firebase se encuentran dos bases de datos: Realtime Database y Cloud Firestore. La base de datos Realtime Database es la primera base de datos creada por Firebase que tiene la característica de tener los datos sincronizados en tiempo real, su estructura de información aparece en el formato JSON para un crecimiento constante y cambios rápidos de la estructura de la información por parte de los usuarios. A diferencia de Cloud Firestore que almacena sus datos en una estructura de colecciones y documentos que tienen una estructura similar al formato JSON. Ambas bases de datos tienen una cantidad de

características en común que lo hacen elegible para las diferentes preferencias del usuario; si bien, Realtime Database tiene características que son interesantes para los usuarios que tienen cierta familiaridad con las bases de datos no relacionables y configurables en tiempo de ejecución, Cloud Firestore tiene una mayor facilidad para crear una estructura de información porque mantienen la información ordenada en colecciones y su atributos se encuentran distribuidos en los documentos dentro de cada colección, ya que, lo hace ser manejable por los administradores, no desarrolladores, para mantener una escalabilidad continua conforme avance la evolución de la aplicación. (Firebase Cloud Firestore, 2021)

1.3.9. Material Design

Para la construcción de la interfaz gráfica de usuario se tiene en cuenta la plataforma o sistema operativo móvil que va dirigido la aplicación porque cada sistema tiene componentes únicos para la interacción con los usuarios, por lo cual, cada proveedor del sistema operativo ofrece un conjunto de herramientas para los desarrolladores y así facilitar la construcción de la aplicación móvil. Con los parámetros ofrecidos por cada proveedor, los componentes visuales se adaptan a la plataforma para no incurrir en errores de la interacción del usuario y que la plataforma ofrezca una mayor facilidad a la aplicación para con otras que se encuentren instaladas en el dispositivo.

En cada plataforma se encuentran los lineamientos de diseño, porque las características de estas plataformas son diferentes por la variedad de dispositivos móviles. Material Design es una guía de diseño provisto por el sistema operativo Android para la forma visual, interacción, animaciones entre los diferentes dispositivos que soportan este sistema operativo (Android Developers, 2021). Esta guía de diseño permite a los desarrolladores la creación de elementos visuales a partir de instancias creadas por Material Design y solo realizar pequeñas modificaciones sobre ellas.

Este lineamiento de diseño tiene la característica de ser open-source donde el código es modificado por la comunidad, por tanto, puede ser transferible hacía otras plataformas

diferentes al sistema operativo que lo mantiene. Se adapta a las necesidades de cada aplicación porque mantiene componentes que se adaptan según los requisitos funcionales del software a desarrollar, por lo que, en un entorno multiplataforma esta guía de diseño se adapta a los diferentes sistemas operativos en los que se ejecuta la aplicación. Las características que tiene Material Design para cada aplicación en las tiendas oficiales se adapta a las necesidades de la empresa, por ejemplo, los colores de la empresa se pueden agregar a la aplicación para generar una diferente con otras aplicaciones de funcionalidades parecidas, pero con una parte visual única para mejorar la experiencia de usuario.

1.3.10. SQLite

Es una base de datos relacional que se pueden encontrar en los diferentes sistemas operativos, sea móvil o de escritorio, donde sus características principales se encuentran el tamaño y su extensibilidad en las diferentes plataformas no requiere de librerías externas para su utilización, para ello, se SQLite ha sido desarrollado en el lenguaje de programación C, debido a que, es compatible con la mayoría de plataformas por ser un lenguaje base para otros que se encuentran desarrollados (Apolinario, 2020)

Al ser una base de datos que se encuentran en los ambientes móviles, es una buena opción para utilizarla y solventar los problemas de organización de archivos, manejo de información, creación de tablas, etc. Por lo tanto, las aplicaciones móviles tienden a utilizar esta base de datos por su integración con el dispositivo ha sido solventada por los diferentes sistemas operativos que alberga, además, de ofrecer herramientas de desarrollo para las personas que construyen aplicaciones para simplificar el almacenamiento de información por parte de los usuarios. Dado que, SQLite tiene características de una base de datos relacional, es posible la creación, inserción, actualización y eliminación de información por parte de la aplicación móvil para seguir los mismos lineamientos que otras bases de datos que se encuentran en el mercado.

Las aplicaciones tienen su propia instancia de SQLite para ser utilizada por las aplicaciones para guardar información persistente en el dispositivo durante toda la vida útil del

dispositivo, para evitar instalaciones de proveedores externos dentro del dispositivo y lograr que el usuario utilice la aplicación desde el momento de su instalación. Esta base de datos tiene un tamaño pequeño en comparación con bases de datos dirigidos a plataformas de escritorio, sin embargo, contiene las funcionalidades importantes de una base de datos para gestionar de mejor manera los datos y que el desarrollador pueda migrar algunos scripts, realizados en otras plataformas para SQLite.

Es una ventaja que SQLite se encuentren en las plataformas de Android o iOS porque permite a las aplicaciones multiplataformas funcionar en ambos sistemas operativos sin la necesidad de configuraciones iniciales, por consiguiente, da ese valor añadido de guardar información local en el dispositivo utilizando los mismos comandos, independientemente de la plataforma. Por tanto, el uso de esta base de datos para las aplicaciones multiplataforma interactúa con el mismo para construir aplicaciones que evolucionan con el tiempo y a la vez con la plataforma que los soporta.

CAPITULO II
ANÁLISIS Y DISEÑO

2.1. REQUERIMIENTOS FUNCIONALES Y NO FUNCIONALES

Para esta esta sección se definen los requerimientos funcionales que abarca la aplicación móvil a su salida en las tiendas de aplicaciones.

Requerimientos Funcionales

Para definir los requerimientos funcionales se utiliza la identificación de RF1, que significa, Requerimiento Funcional y la secuencia del número. En la tabla siguiente se pueden encontrar los requerimientos funcionales:

Tabla 2

Requerimientos funcionales

Código	Requerimiento
RF1	El usuario pude ver la lista de productos y/o servicios con sus características
RF2	Registro y Autenticación de cuentas de usuarios
RF3	El usuario puede observar el estado de los pedidos y el vendedor registra pedidos.
RF4	Cotización de productos para enviar por correo.
RF5	Descargar los catálogos de productos.
RF6	Comunicación del usuario con la empresa (formularios, llamadas, correo, mensajes)
RF7	Carrito de compras para agrupar los productos y cotizar en conjunto.

Nota. Lista de los requerimientos funcional que alberga la aplicación.

Elaborado por: El autor.

Requerimientos No Funcionales

Para la identificación de los requerimientos no funcionales se emplea el siguiente formato, RNF1 que significa, Requerimiento No Funcional y la secuencia del número. A continuación, se encuentra una tabla de lista de requerimientos no funcionales.

Tabla 3

Requerimientos no funcionales

Código	**Requerimiento**
RNF1	Fácil de utilizar
RNF2	Presentación de la aplicación (Onboarding)
RNF3	Colores de la empresa reflejadas en la aplicación
RNF4	Escalabilidad Para incrementar la funcionalidad de servicios, comunicación y productos.

Nota. Tabla de los requerimientos no funcionales.

Elaborado por: El autor.

2.2. PERFILES DE USUARIO

Para el manejo de los pedidos por parte de la aplicación, se utilizan los roles de usuario, que permiten distinguir a los diferentes actores que utilizan la aplicación para diferentes fines pero que conlleva al mismo objeto, de la administración y seguimiento de pedidos. Se ha identificado los siguientes perfiles de usuario:

Usuario Cliente. El tipo de usuario cliente, es la persona que se descarga la aplicación para ejecutar las funcionalidades principales de la aplicación y utilizarla como una persona externa a la empresa, para conllevar una mejor comunicación con la misma. El cliente tiene la capacidad de crear una cuenta de usuario a partir de un correo electrónico o utilizando una

cuenta de Google, para utilizar la funcionalidad de visualización de pedidos por parte de la empresa, esta clase de usuario solo puede acceder a la información visual (no editable) del estado de los pedidos que realizo.

***Usuario Vendedor*.** El tipo de usuario vendedor es una persona que trabaja en la empresa Ferro Torre o cliente interno que realiza toda la gestión de ventas, pedidos y contacto con el cliente para ofrecer a los clientes externos los productos que la empresa tiene. El vendedor tiene la capacidad de ingresar, en la misma aplicación, el correo electrónico y una contraseña para acceder a la pantalla de administración de pedidos, donde en cada pedido; el vendedor tiene la potestad de cambiar el estado del pedido, para que, el usuario pueda observar el cambio en la sesión del usuario cliente. El vendedor puede crear un nuevo pedido con los datos básicos, como lo son, fecha, mensaje, cliente registrado y estado inicial del nuevo ingreso, para cuando el usuario requiera observar el pedido realizado a través del vendedor dentro de la misma aplicación.

Usuario Administrador. El tipo de usuario administrador es la persona que, de igual manera trabaja en la empresa Ferro Torre como vendedor, pero mantiene un rol con mayor importancia sobre los otros vendedores de la empresa; por tal motivo, este usuario tiene la capacidad de observar todos los pedidos de los otros vendedores y de sí mismo, facilitando el trabajo de administración de pedidos sobre cada uno de los vendedores. El administrador puede crear nuevos pedidos, con la fecha, mensaje, cliente registrado y estado inicial, pero además puede asignar un vendedor para el nuevo pedido a crear, para que, el vendedor pueda administrar este pedido de forma individual. Los usuarios administradores tienen la opción de crear nuevos usuarios vendedores o administradores dentro de la base de datos de Firebase firestore para facilitar a las personas con poco conocimiento en tecnología, la incorporación de nuevas personas en el área de ventas.

2.3. ARTEFACTOS UML

2.3.1. Diagrama de Casos de Uso

Para la construcción de la aplicación móvil en base a los requerimientos funcionales antes descritos, se utilizan la metodología de diagramado de comportamiento como acciones que ejecutan los usuarios con la aplicación.

Para la tabla 4 y figura 2, se desglosa el flujo que sigue el usuario para desplegar las características del producto y/o servicio.

Tabla 4

Historia de caso de Uso RF1

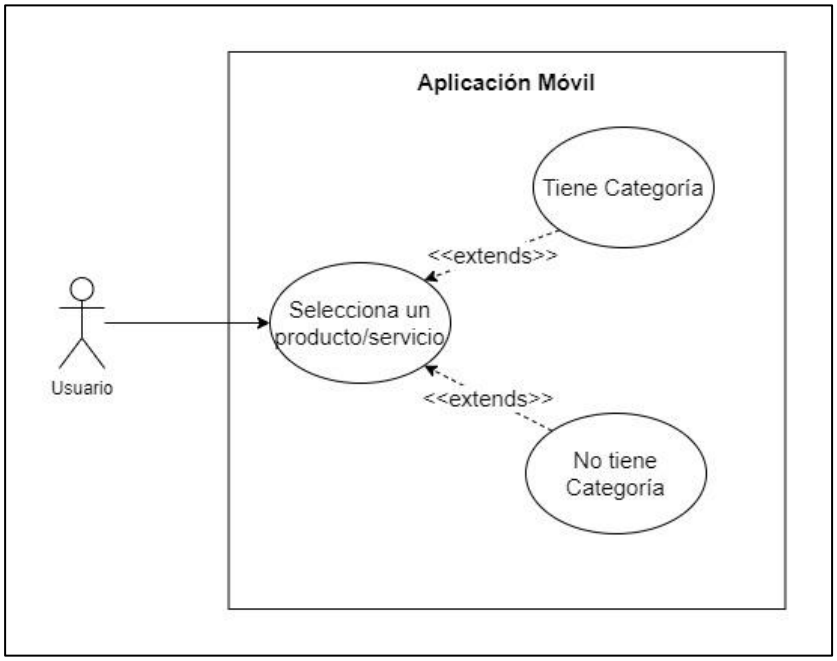
Actores	Usuario
Requisito Funcional	RF1
Flujo Normal	El flujo empieza cuando el usuario inicia la aplicación. La aplicación muestra una lista de productos y servicios. El actor selecciona un producto o servicio. La aplicación muestra una lista de categorías del producto seleccionado. El actor selecciona una categoría. La aplicación indica las características y ficha técnica.
Flujo Alternativo	4.a. Si el producto no tiene una categoría, la aplicación muestra las características y ficha técnica.

Nota. Flujo del usuario con la aplicación para observar los productos de la aplicación.

Elaborado por: El autor.

Figura 2

Diagrama Caso de Uso RF1



Nota. Caso de uso para el Flujo de visualización del producto.

Elaborado por: El autor.

Para la tabla 5 y figura 3, se expone el flujo del usuario para registro y autenticación de la cuenta dentro de la aplicación.

Tabla 5

Historia de Caso de Uso RF2

Actores	Usuario
Requisito Funcional	RF2
Flujo Normal	1. El flujo inicia cuando el usuario se dirige a la pantalla de registro y autenticación de cuentas. 2. La aplicación muestra las opciones de registro y autenticación.

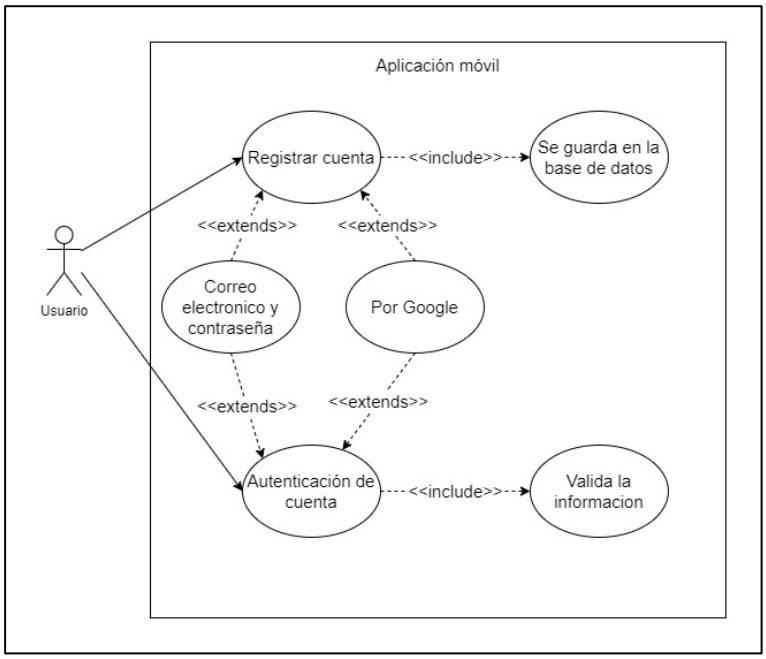
	3. El usuario selecciona registro e ingresa los datos del correo con una contraseña o por una cuenta de Google. 4. La aplicación ingresa la información en la base de datos y se autentica dentro de la aplicación.
Flujo Alternativo	3.a. El usuario selecciona autenticación e ingresa los datos de correo con la contraseña o por una cuenta de Google. 4.a. La aplicación valida la información ingresa y se autentica dentro de la aplicación.

Nota. Flujo de usuario para registrarse y autenticarse en la aplicación.

Elaborado por: El autor.

Figura 3

Diagrama Caso de Uso RF2



Nota: Caso de uso para el registro y autenticación de usuario por correo y/o cuenta de Google.

Elaborado por: El autor.

Para la tabla 6 y figura 4, se describe el flujo cuando el usuario desea observar el estado de su pedido.

Tabla 6

Historia de Caso de Uso RF3

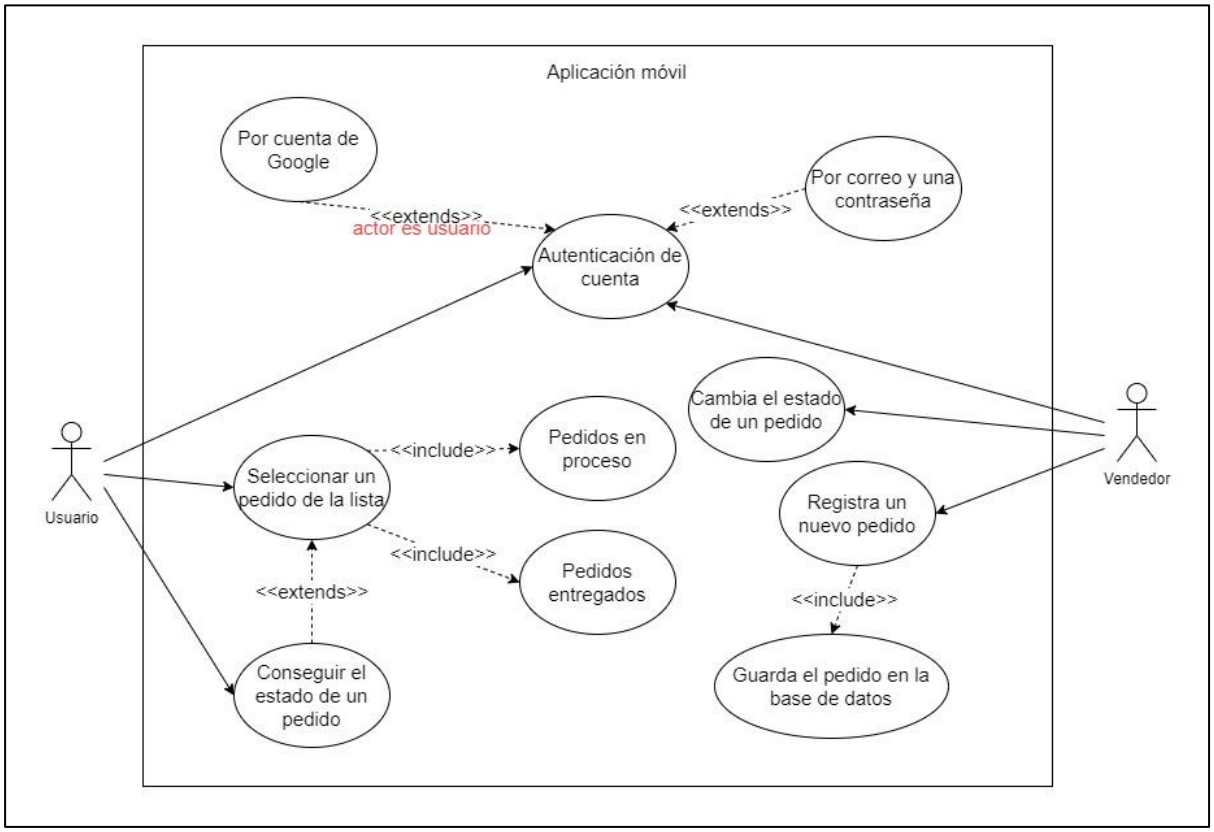
Actores	Usuario - Vendedor
Requisito Funcional	RF3
Flujo Normal	1. El flujo empieza cuando el usuario se autentica en la aplicación por medio de un correo electrónico y una contraseña. 2. La aplicación muestra al usuario una lista de pedidos. (Activos o inactivos). 3. El usuario selecciona un pedido. 4. La aplicación indica el estado del pedido. La información que se expone es: Nombre cliente, nombre vendedor, fecha ingresada y descripción de lo adquirido. Vendedor 1. El flujo empieza cuando el vendedor se autentica en la aplicación por medio de un correo electrónico y una contraseña. 2. La aplicación indica el estado de pedidos y la opción de ingresar un nuevo pedido. 3. El vendedor presiona escoge la opción de un nuevo pedido y registra el nuevo pedido con un cliente que se encuentre almacenado en la base de datos. El sistema guarda el nuevo pedido y lo asigna al cliente.
Flujo Alternativo	Usuario 1.a. El usuario se autentica por medio de una cuenta de Google (como responde la aplicación).

Nota: Flujo para el caso de uso del RF3.

Elaborado por: El autor.

Figura 4

Diagrama de caso de Uso RF3



Nota: Caso de uso para el usuario pueda observar el estado de sus pedidos.

Elaborado por: El autor.

Para la tabla 7 y figura 5, se expone el flujo de usuario para realizar una cotización.

Tabla 7

Historia de Caso de Uso RF4

Actores	Usuario
Requisito Funcional	RF4
Flujo Normal	1. El flujo empieza cuando el usuario selecciona un producto y/o servicio. 2. La aplicación muestra una lista de categorías del producto seleccionado.

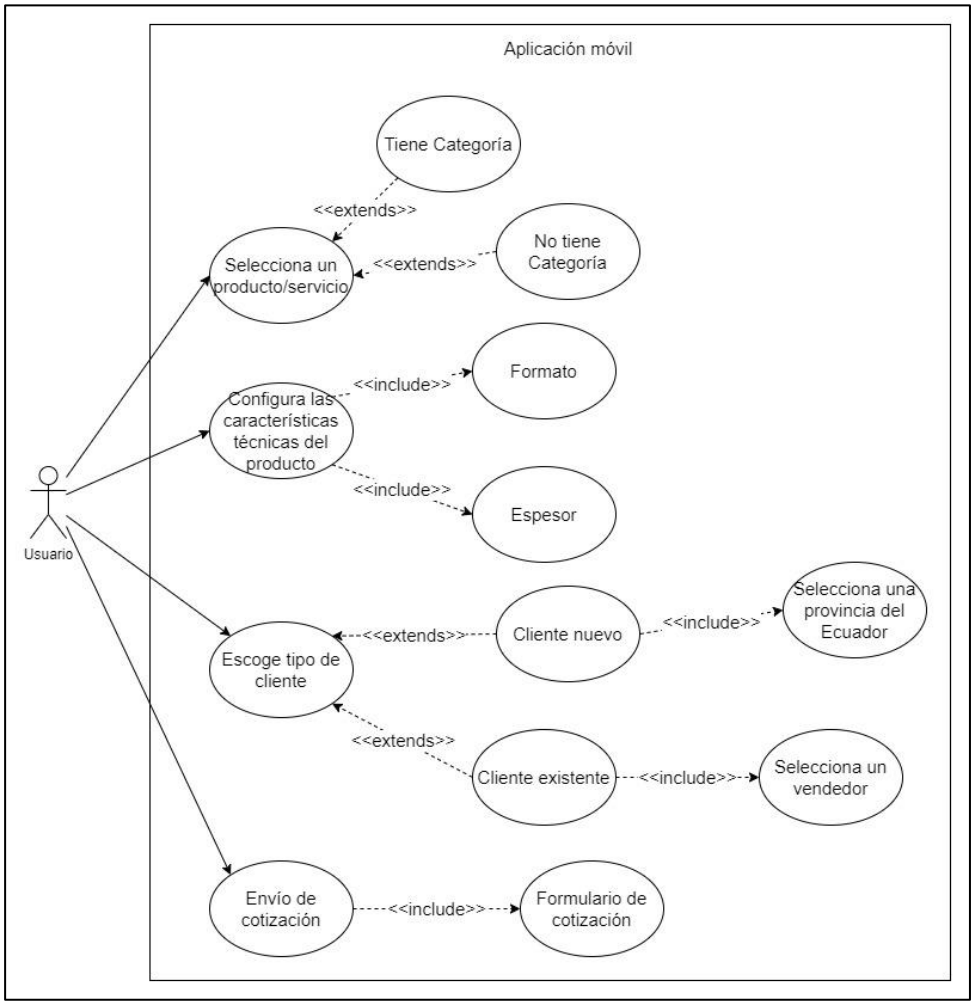
	3. El usuario selecciona una categoría. 4. La aplicación indica las características y la ficha técnica. 5. El usuario selecciona el formato y espesor de la ficha técnica. 6. La aplicación indica dos opciones sobre el tipo de cliente, nuevo o existente. 7. El usuario escoge cliente nuevo. 8. La aplicación muestra una lista de provincias del Ecuador. 9. El usuario selecciona o busca una provincia en la que se encuentre. 10. La aplicación muestra un formulario de registro para enviar cotización. 11. El usuario llena la información del registro y se envía por correo.
Flujo Alternativo	7.a. El usuario escoge el cliente existente. 8.a. La aplicación muestra un listado de vendedores por algunas ciudades. 9.a. El usuario selecciona o busca un vendedor que conoce.

Nota: Flujo de Usuario para cotizar un producto y/o servicio.

Elaborado por: El autor.

Figura 5

Diagrama de caso de uso RF4



Nota: Caso de uso para realizar una cotización con las características del producto.

Elaborado por: El autor.

Para la tabla 8 y figura 6, se indica el flujo de usuario cuando el cliente requiera descargar los catálogos de la aplicación móvil.

Tabla 8

Historia de caso de uso RF5

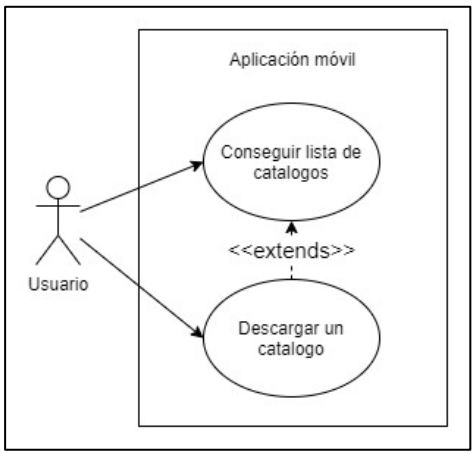
Actores	Usuario
Requisito Funcional	RF5
Flujo Normal	1. El flujo empieza cuando el usuario ingresa a la pantalla principal. 2. El sistema muestra los productos, opción de contacto y catálogo. 3. El usuario selecciona la opción de catálogo. 4. La aplicación indica una lista de los catálogos disponibles. 5. El usuario selecciona un catálogo. 6. La aplicación procede a descargar el archivo en formato PDF a través de un navegador web.
Flujo Alternativo	6.a. La aplicación no puede descargar el archivo y muestra un mensaje al usuario, sobre el archivo no existente.

Nota: Flujo de usuario cuando el cliente descarga un catálogo.

Elaborado por: El autor.

Figura 6

Diagrama de caso de uso RF5



Nota: Caso de uso para la descarga de un catálogo por parte del usuario.

Elaborado por: el autor.

Para la tabla 9 y figura 7, se indica el flujo de usuario para ejecutar un contacto, a través de diferentes medios, con la empresa.

Tabla 9

Historia de caso de uso RF6

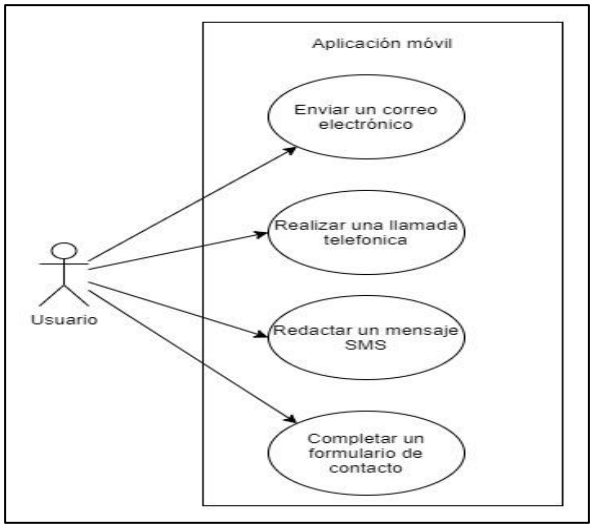
Actores	Usuario
Requisito Funcional	RF6
Flujo Normal	1. La aplicación indica la pantalla principal de contacto o medios de comunicación. 2. La aplicación indica varios medios de comunicación. Se muestran los siguientes: Correo electrónico, llamadas, mensajes SMS y formulario de contacto.
Flujo Alternativo	2.a. Al no tener internet. Para los medios de comunicación como correo electrónico y formulario, muestra al usuario un mensaje de no conexión a Internet.

Nota: Flujo de usuario para los diferentes medios de comunicación entre el cliente y la empresa.

Elaborado por: El autor.

Figura 7

Diagrama de caso de uso RF6



Nota: Caso de uso para los medios de comunicación que dispone el usuario para comunicarse con la empresa.

Elaborado por: El autor.

Para la tabla 10 y figura 8, se expone el flujo de usuario para agregar y observar productos en el carrito.

Tabla 10

Historia de caso de uso RF7

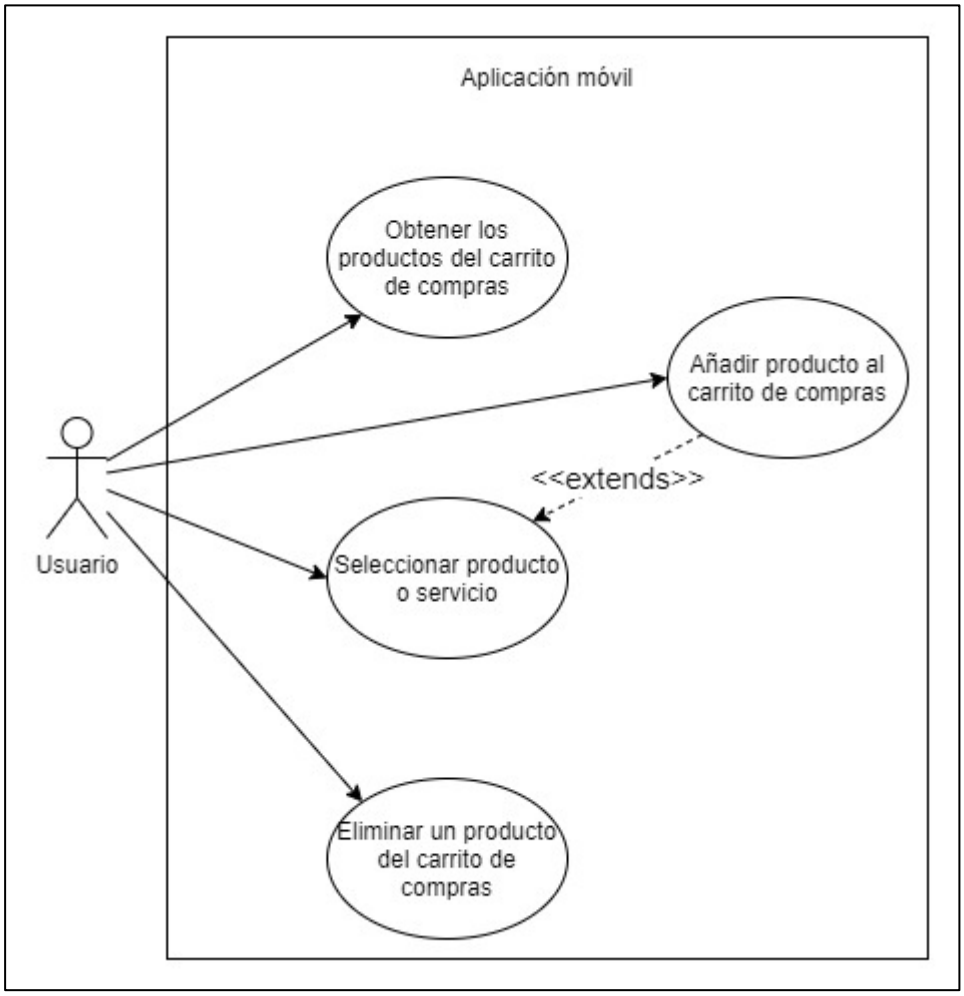
Actores	Usuario
Requisito Funcional	RF7
Flujo Normal	1. El flujo empieza cuando el usuario selecciono un producto o servicio con sus características técnicas. 2. La aplicación muestra la opción de cotización o de añadir al carrito de compras. 3. El usuario escoge añadir al carrito de compras. 4. La aplicación retorna a la pantalla principal donde se encuentran los productos y el carrito de compras. 5. El usuario selecciona el carrito de compras. 6. La aplicación muestra los productos añadidos al carrito. Por cada producto se muestra las siguientes características: Nombre, Cantidad, eliminar, formato y espesor (solo si es producto).
Flujo Alternativo	6.a. El carrito de compras está vacío. La aplicación muestra una imagen y un texto representativo donde se indica que el carrito de compras está vacío.

Nota: Flujo de usuario para añadir y observar los productos del carrito de compras.

Elaborado por: El autor.

Figura 8

Diagrama de caso de uso RF7



Nota: Caso de uso para añadir, eliminar y observar productos del carrito de compras.

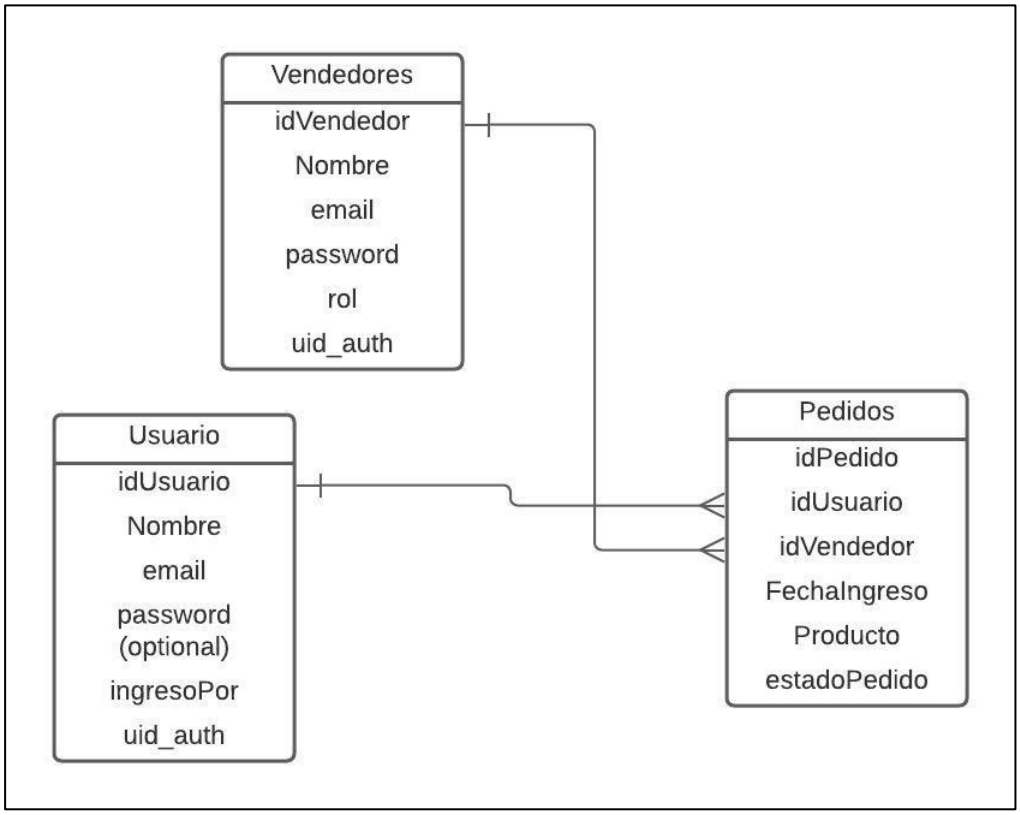
Elaborado por: El autor.

2.3.2. Diagrama de base de datos

Para la base de datos se lo realizó a partir de la interacción de registro como de autenticación por parte del usuario, el estado los pedidos y los vendedores (clientes internos de la empresa Ferro Torre).

Figura 9

Diagrama de Base datos



Nota: Tablas de Vendedores, Usuario y pedidos; para gestionar el estado de los pedidos.

Elaborado por: El autor.

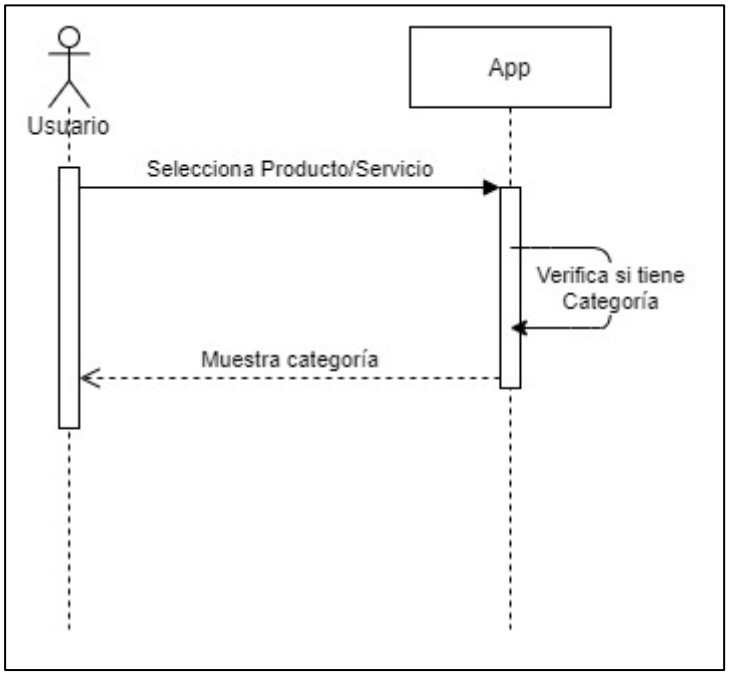
2.3.3. Diagrama de Secuencia

El diagrama de secuencia es una representación gráfica de la comunicación con la diferente información manejada con el usuario y el aplicativo móvil en base a los casos de uso descritos anteriormente.

La figura 10, muestra la interacción que realiza el usuario para mostrar los productos dentro de la aplicación.

Figura 10

Diagrama de secuencia RF1



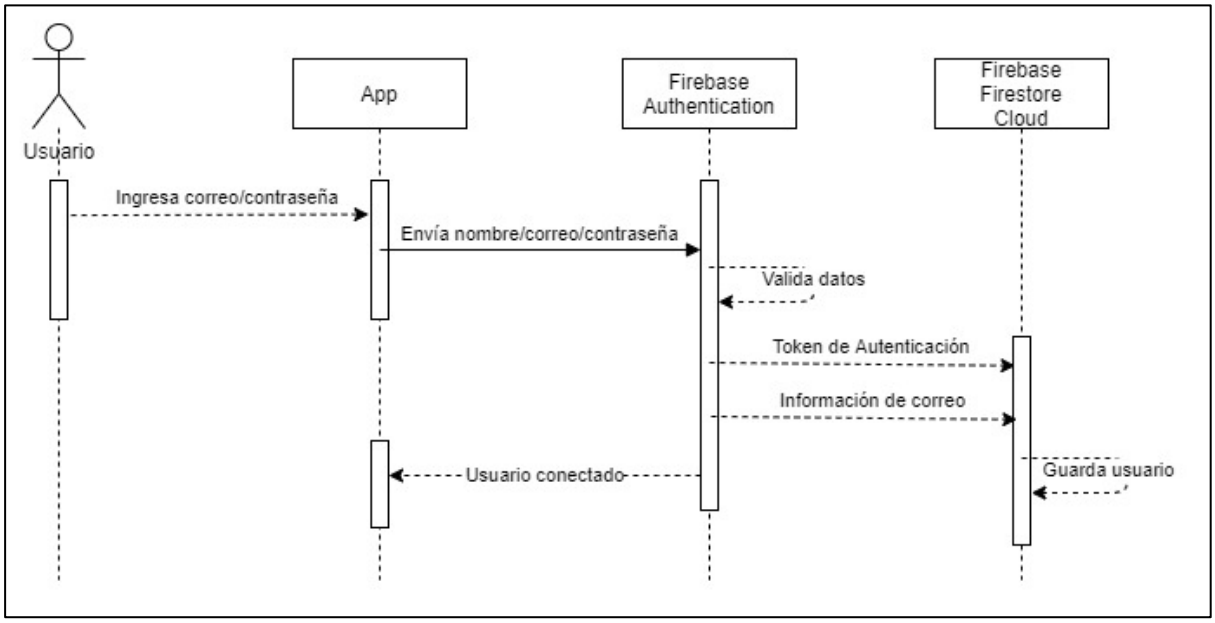
Nota: Interacción del usuario con la aplicación para mostrar los productos y categorías.

Elaborado por: El autor.

El registro y autenticación de usuario se realiza con Firebase Authenticación y FireStore (La configuración de los productos de Firebase, se encuentran en el **Anexo 1** de este documento) que permite guardar la información de los usuarios, para luego, utilizar esta información para gestionar los diferentes pedidos de cada cliente o usuario, dentro de la aplicación. En la figura 11, se puede observar el registro de un usuario utilizando una cuenta de correo electrónico y una contraseña (mismo procedimiento para registro con una cuenta de Google), además, la información obtenida de Firebase Authenticación, como el token, se guarda en la Firebase Firestore Cloud para mantener la información actualizada ante cualquier cambio de estado.

Figura 11

Diagrama de secuencia sobre registro de usuario



Nota: Registro de usuario en los productos de Firebase, simplifica el trabajo de validación.

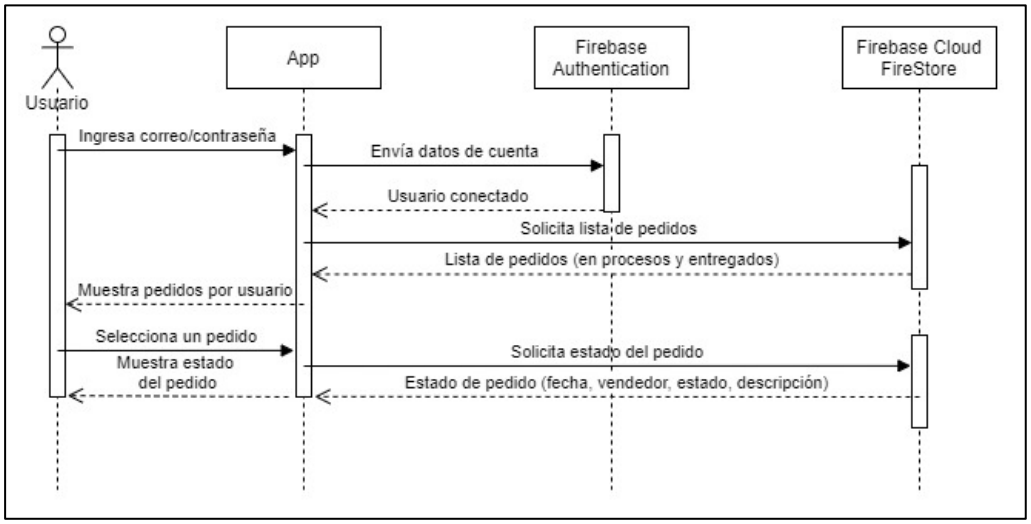
Elaborado por: El autor.

Este procedimiento es simplificado, debido a que, Firebase Authentication verifica la información en su base de datos interna o con los proveedores de correo electrónico y el almacenamiento de la contraseña se lo valida de la misma forma. Este producto genera un token único de autenticación para identificar al cliente dentro del entorno de Firebase y esta información es almacenada en el Firebase Firestore Cloud en el documento de usuarios.

En la figura 12, se puede observar el diagrama de secuencia para, cuando el usuario se autentica para ver el estado(s) de los pedidos que tiene.

Figura 12

Diagrama de secuencia RF3 (Usuario)



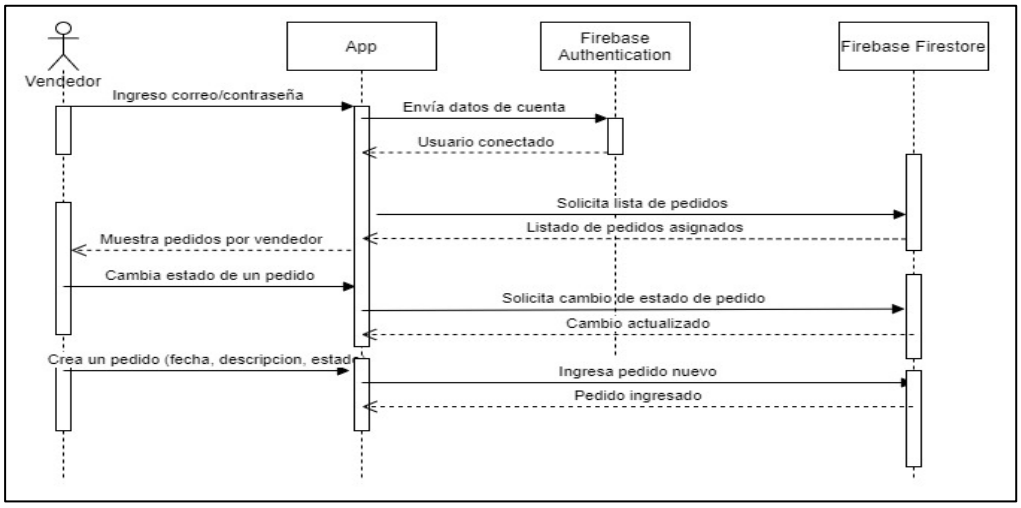
Nota: Flujo de usuario para autenticarse en la aplicación y conseguir los pedidos realizados.

Elaborado por: El autor.

Para el complemento de la figura anterior, se utiliza la figura 13 para describir el flujo de información o mensajes para el lado del vendedor con la interacción de administración de pedidos.

Figura 13

Diagrama de secuencia RF3 (Vendedor)



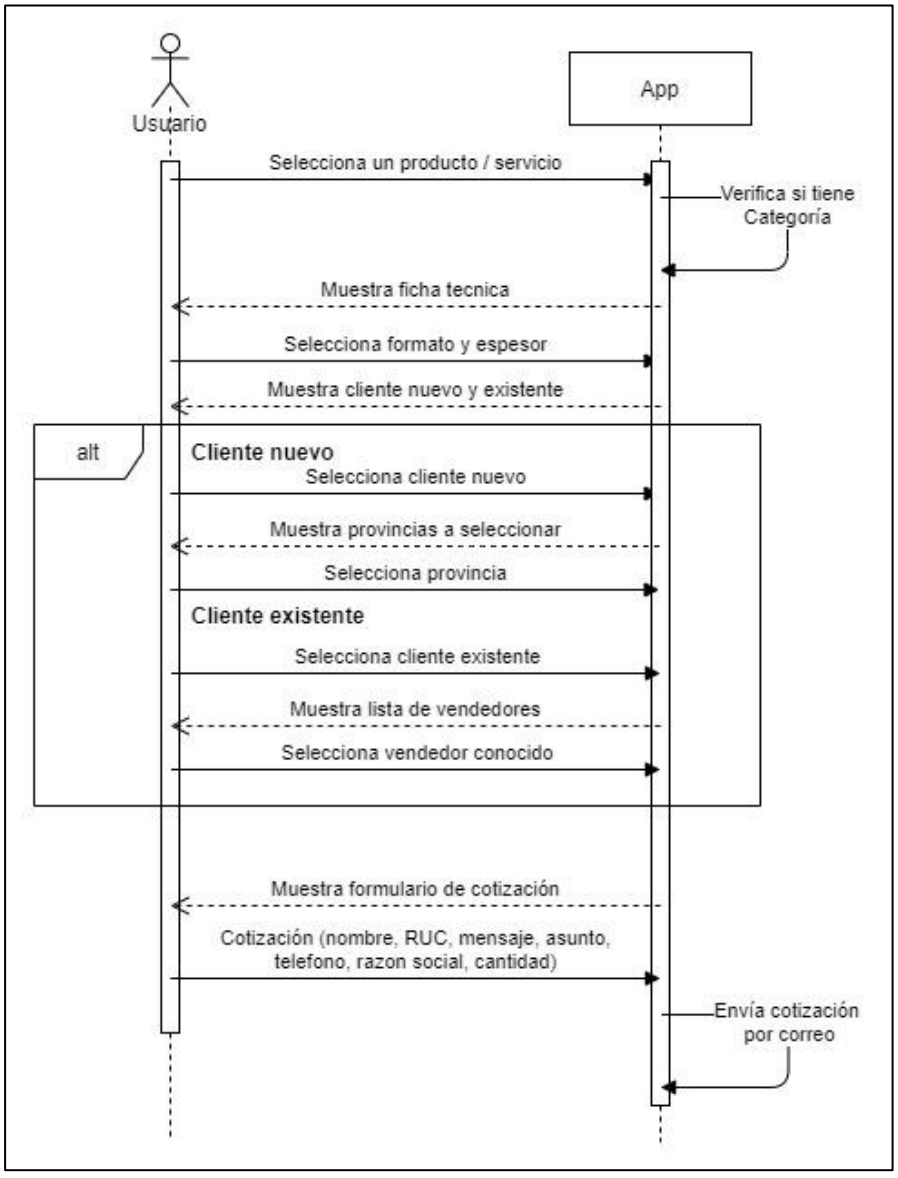
Nota: Flujo de mensajes entre la aplicación y el vendedor para administrar pedidos.

Elaborado por: El autor.

Para el requerimiento funcional RF4, en la figura 14, sobre el diagrama de secuencia basado en el caso de uso de la figura 5, por lo cual, se considera las opciones de usuarios para identificarse sobre el cliente nuevo y existente, además de la ficha técnica intuitiva que el usuario selecciona valores de formato y espesor para definir las características del producto deseado.

Figura 14

Diagrama de secuencia RF4



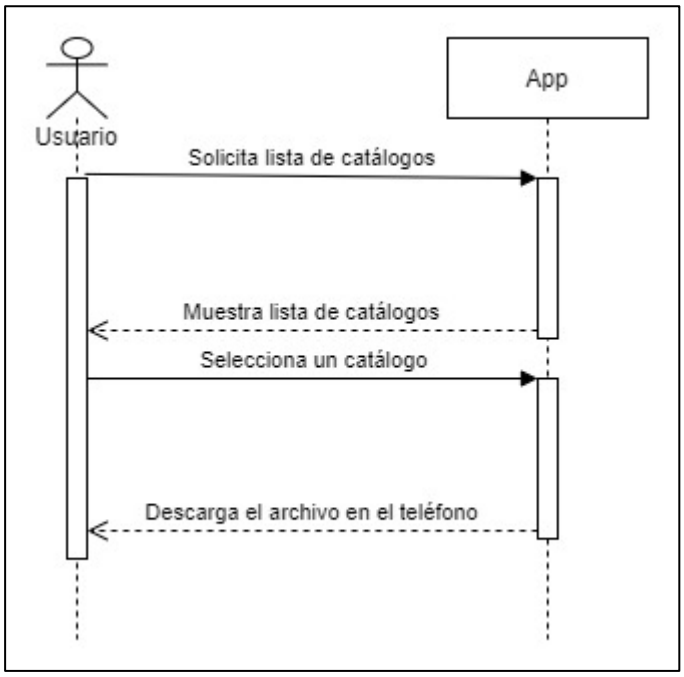
Nota: Flujo de los mensajes entre el usuario y la aplicación para realizar una cotización.

Elaborado por: el autor.

Para el diagrama de secuencia del requisito funcional con identificador RF5, se expresa en la figura 15, para demostrar la información que fluye a través de la información para la descarga de los catálogos de productos que tiene la empresa.

Figura 15

Diagrama de secuencia RF5



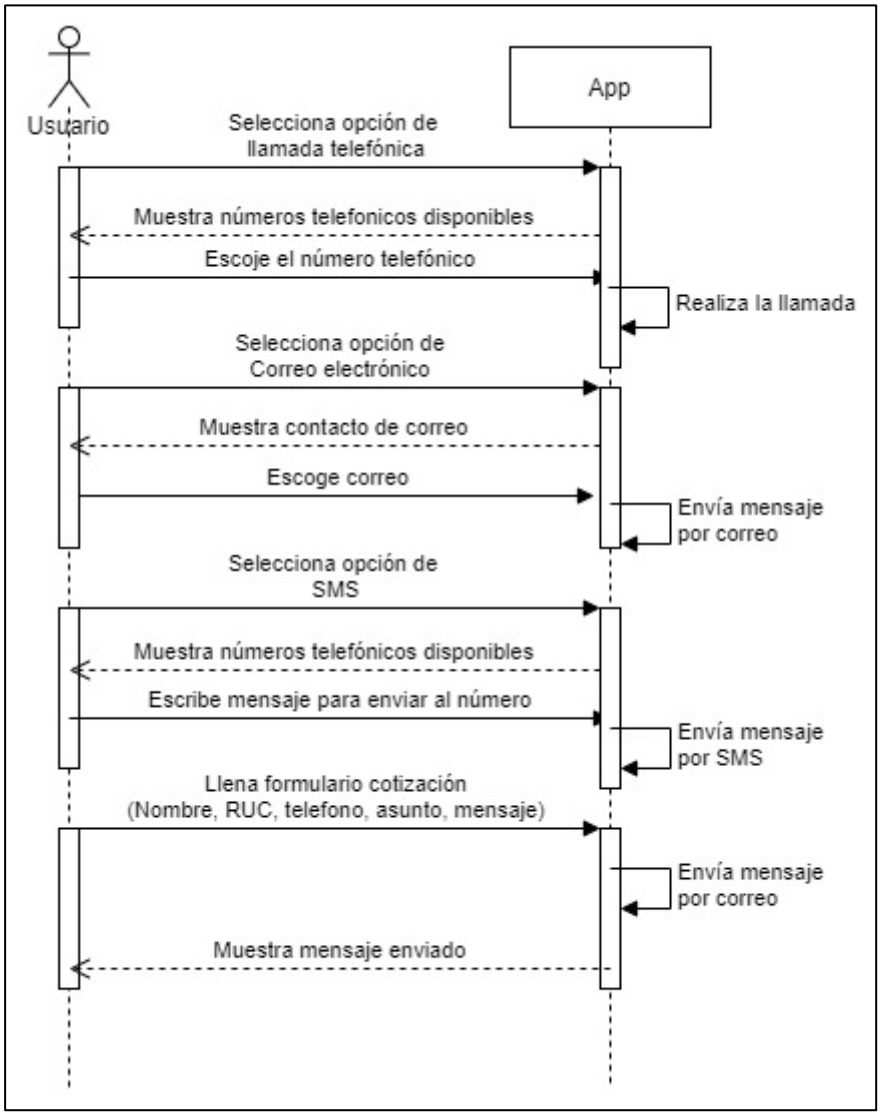
Nota: Flujo de la interacción entre el usuario y la aplicación para descargar un catálogo.

Elaborado por: El autor.

La figura 16, indica el diagrama de secuencia para el requisito funcional con identificador RF6.

Figura 16

Diagrama de secuencia RF6



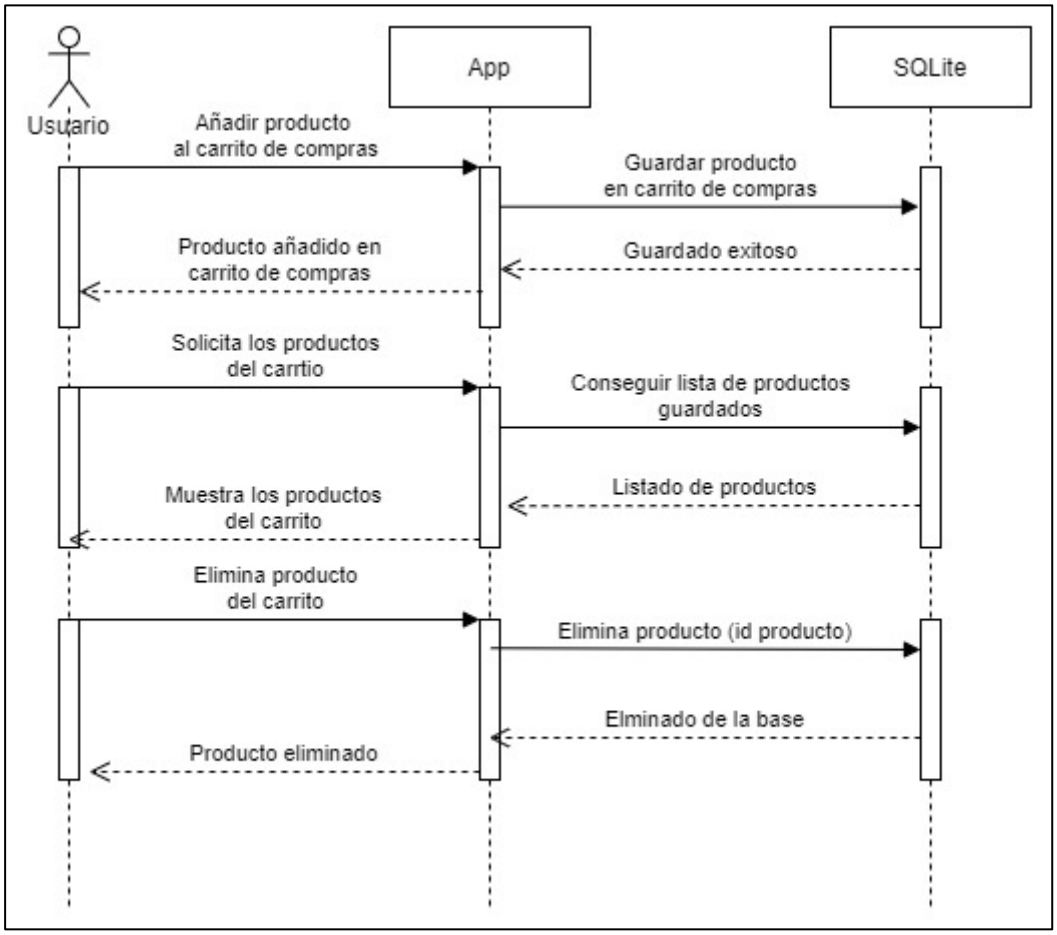
Nota: Uso de la aplicación para los diferentes medios de comunicación.

Elaborado por: El autor.

La figura 17, se muestra el diagrama de secuencia para el requisito funcional con identificador RF7, que permite al usuario realizar acciones como añadir o eliminar productos del carrito de compras. Este proceso se lo realiza con la interacción de la base de datos SQLite para almacenar los datos de forma persistente en la aplicación, sin la necesidad de una conexión a Internet, por tal motivo, no se almacena en el Firebase firestore.

Figura 17

Diagrama de secuencia RF7



Nota: Gestión por parte del usuario para el carrito de compras.

Elaborado por: El autor.

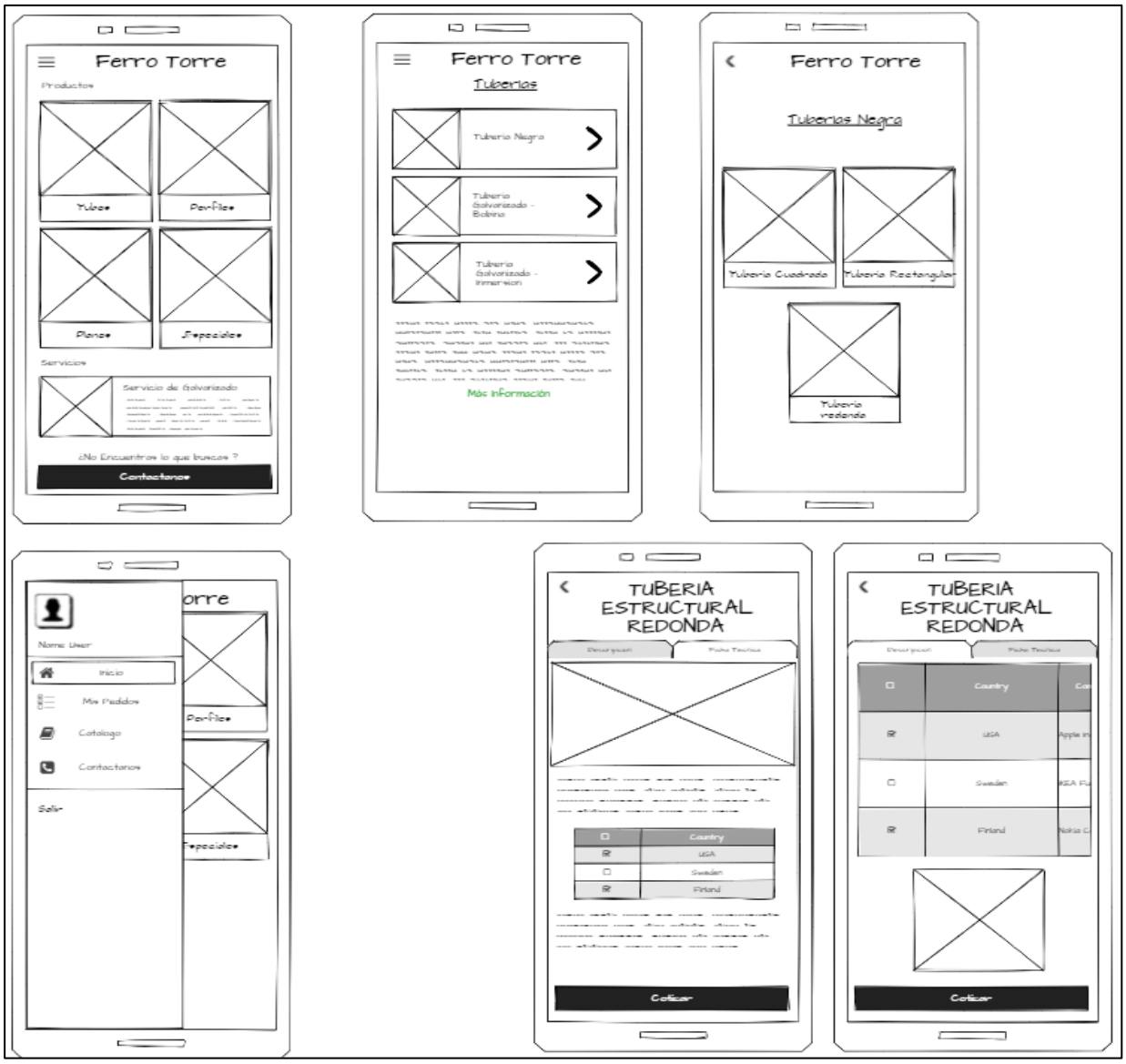
2.4. PROTOTIPO DE LA INTERFAZ DE USUARIO

Para facilitar el desarrollo de una aplicación móvil, se ha optado por realizar un prototipo o mockup para tener las nociones básicas de la interacción que tendrá el usuario con la aplicación. En este proceso se definen los componentes más básicos de una interfaz de usuario, como los botones, imágenes, iconos y texto. Por tal motivo, al tener este prototipo en una versión preliminar, acelera el proceso de diseño y desarrollo porque da una base fundamentada sobre las pantallas principales que se van a tener dentro del aplicativo móvil.

A continuación, se presenta en la figura 8, el prototipo realizado en la herramienta gratuita de nombre Mockflow; de forma que se valida las pantallas principales, con gráficos al estilo de dibujo a mano porque esta clase de prototipo tiene como objetivo ser de rápido desarrollo para validarlo con mayor rapidez.

Figura 18

Mockup de la aplicación Ferro Torre App realizado en Mockflow



Nota: Prototipo de las pantallas principales de la aplicación realizado en Mockflow.

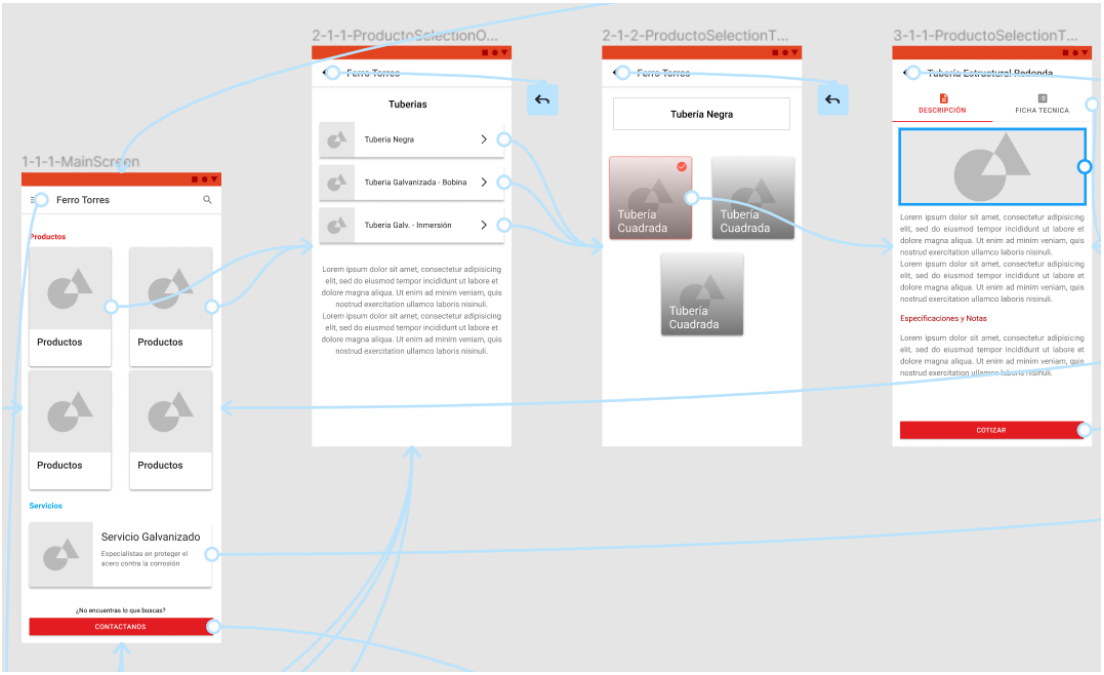
Elaborado por: El autor.

2.5. DISEÑO DE LA INTERFAZ DE USUARIO

Con un prototipo validado por el usuario, la construcción de un diseño con mayores elementos gráficos se vuelve más fácil de realizar, tal que, los principales componentes visuales de la aplicación son validados. El diseño para un acercamiento a un producto final, que después se desarrollará en código, se convierte en una tarea de creatividad porque se implementan elementos visuales más completos, como lo son, los colores, animaciones y la navegación. Para transformar el prototipo en un diseño fiable, se utiliza el software gratuito Figma, que provee de las herramientas necesarias para construir un diseño más agradable para el usuario y pueda ser validado con una navegación decente. En la figura 9, se presentan algunas de las pantallas del diseño con mayor fidelidad donde se muestran conexiones, que indican la navegabilidad de la aplicación entre las diferentes pantallas.

Figura 19

Diseño de la aplicación con navegación



Nota: Las flechas de color celeste son elementos, donde el usuario interactúa con los componentes, que Figma los interpreta para la navegación.

Elaborado por: El autor.

En la figura 19, se puede observar, que los colores que la empresa tiene en el logo están correlacionados con la interfaz de usuario y su relación con los componentes de la aplicación indican que la aplicación pertenece a la empresa. Cada componente observado está en base a la guía de diseño de Material Design provisto por las librerías abiertas que provee el software Figma para el diseño; por tal motivo, esta interfaz de usuario tiende acercarse a la realidad de la aplicación porque mantiene el esquema utilizado con los mismos componentes que se utilizan en las librerías internas del SDK Flutter.

CAPÍTULO III
IMPLEMENTACIÓN Y PRUEBAS

3.1. DESARROLLO E IMPLEMENTACIÓN

El desarrollo de la aplicación se realizó en base al SDK de Flutter con el editor de código denominado Android Studio, que permite la integración de los códigos con mayor rapidez y la pruebas en los emuladores correspondientes. Esta aplicación está dirigida para los sistemas operativos de Android y iOS, para cada una de estas plataformas, tienen las tiendas oficiales que son Google Play y Apple Store, respectivamente.

Versión mínima de sistema operativo. Para cada sistema operativo, existen diferentes versiones disponibles en los teléfonos móviles distribuidos de forma no equitativa, es decir, que un teléfono no pueda tener la misma versión del sistema operativo que en otro teléfono con la misma plataforma; por tal motivo, la aplicación soporta desde una versión mínima de sistema operativo, cuando cualquier teléfono de un usuario se encuentre en el rango superior de la versión mínima propuesta por la aplicación, pueda encontrar a la aplicación y descargarla. En la tabla 11, se indica las versiones mínimas para cada sistema operativo, que la aplicación desarrollada puede soportar para su correcto funcionamiento.

Tabla 11

Versión mínima de cada sistema operativo

Sistema operativo	Versión mínima soportada por la aplicación
Android	API 21: Android 5.0 Lollipop
iOS	iOS 9.0

Nota: Versión mínima que la aplicación puede ejecutarse en cada sistema operativo.

Elaborado por: El autor.

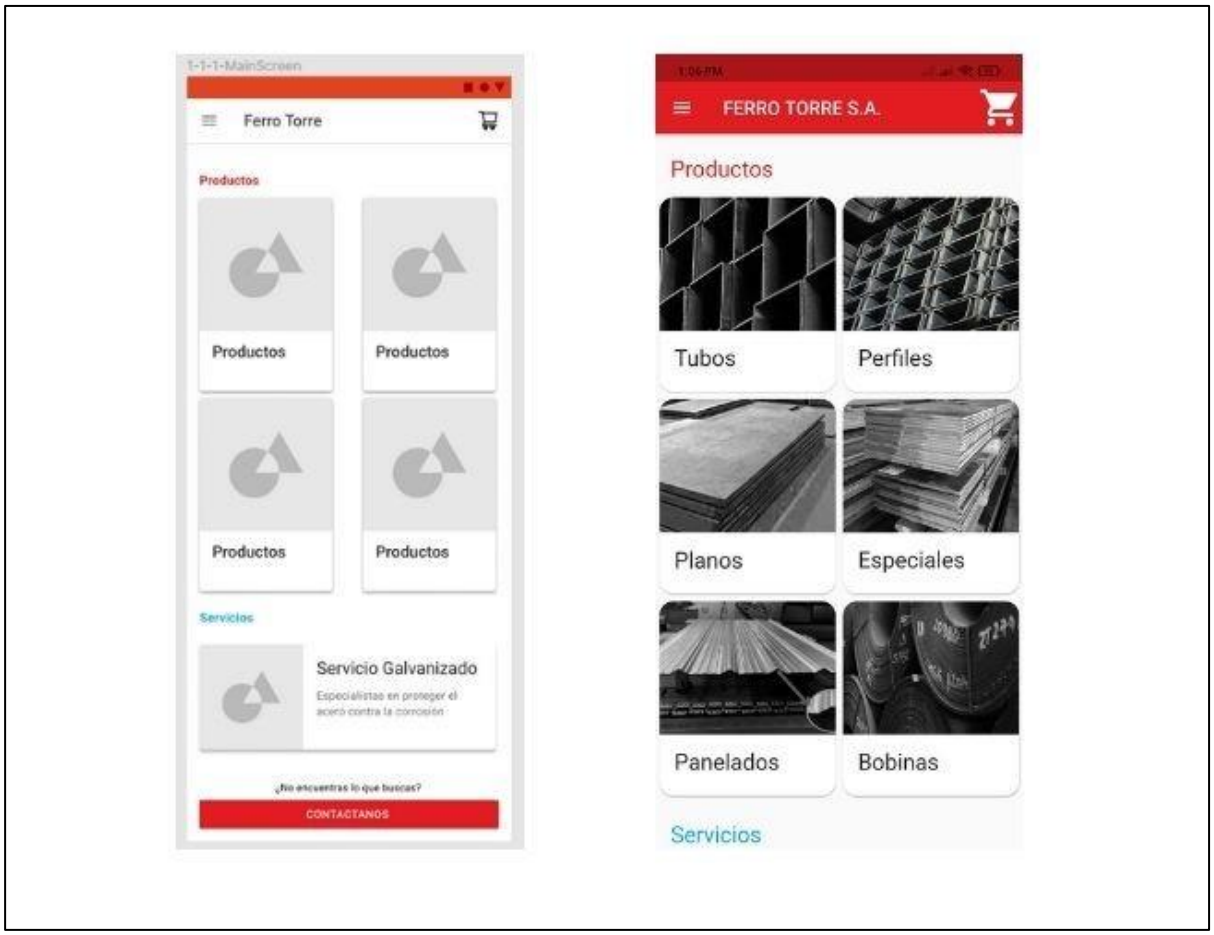
Esta discriminación para las versiones de cada sistema operativo es debido, principalmente, por la compatibilidad de componentes o algoritmos que no son soportados para algunas versiones muy antiguas de cada plataforma.

3.1.1. Desarrollo

Con el SDK de Flutter se puede construir las pantallas de usuario con los elementos provistos por este kit de desarrollo, a partir de la guía de diseño de Material Design. Las interfaces de usuario diseñadas en el software de Figma son de utilidad para convertir los elementos gráficos en componentes codificables para así facilitar el análisis de un diseño en momento de codificación. En la figura 20, se muestra una captura de las interfaces creadas tanto en Figma como en el SDK de Flutter.

Figura 20

Captura de la Pantalla principal de la aplicación (Figma y SDK Flutter)



Nota: Capturas de la pantalla realizadas en Figma (izquierda) y SDK Flutter (Derecha).

Elaborado por: El autor.

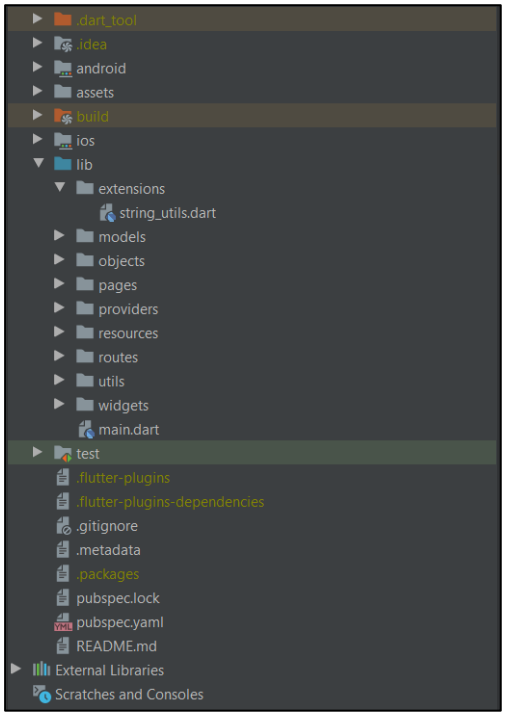
En la figura 20, se puede observar la similitud entre pantallas, debido a que, Figma tiene como objetivo validar un diseño previo de la aplicación para facilitar el trabajo del desarrollador

y así entregar un producto más rápido al cliente. Con el SDK Flutter, esta transición de elementos de un diseño estático a componentes interactivos se lo realiza con los propios componentes visuales que contiene el kit de desarrollo, por tal motivo, el diseño tiene gran similitud con la aplicación puesta en marcha sobre un dispositivo móvil.

Estructura del proyecto de Flutter. Para el SDK de Flutter, la estructura del proyecto se encuentra incrustado con parte nativa de los sistemas operativos, que este proyecto necesita, Android y iOS. A continuación, se muestra en la figura 21, todas las carpetas del proyecto, en donde se puede destacar las carpetas, Android y iOS (cada uno contiene archivos para construir una aplicación nativa), la carpeta lib, que contiene los archivos con extensión dart y la carpeta de recursos, que contiene las imágenes junto algunos archivos de información que puede utilizar la aplicación.

Figura 21

Estructura del proyecto de Flutter para la aplicación



Nota: Captura de pantalla del proyecto de la aplicación, en código.

Elaborado por: El autor.

En el proyecto de Flutter, todos los archivos que forman parte de la aplicación para la codificación de funcionalidades o de interfaz de usuario, son de extensión dart lo que facilita la gestión y comunicación entre los archivos sin la necesidad de configuraciones externas. Cada archivo del proyecto puede ser utilizado para diferentes objetivos de la aplicación, es decir, un archivo que contenga la interfaz de usuario tiene la misma configuración para un archivo de codificación de funcionalidad.

3.1.2. Algoritmos importantes

En las aplicaciones móviles deben existir una diferenciación en base a la interfaz de usuario y como la información es mostrada para el usuario. Para ello, se utilizan los temas, que permite a la aplicación utilizas los colores o la tipografía creada por la etapa del diseño, para adaptarse al sistema y se pueda diferenciar entre otras aplicaciones de la misma categoría. Para la figura 22, se puede observar el archivo de colores que tiene la aplicación y se aplican a todos los componentes visuales (botones, texto, barra de estado, etc.).

Figura 22

Archivo Dart sobre los colores de la aplicación

colors.dart

```dart
import 'package:flutter/widgets.dart';

const Color primaryColor = Color(0xFFE21C21);
const Color primaryLightColor = Color(0xFFFF5D4B);
const Color primaryDarkColor = Color(0xFFA70000);

const Color secondaryColor = Color(0xFF03A9F4);
const Color secondaryLightColor = Color(0xFF67DAFF);
const Color secondaryDarkColor = Color(0xFF007AC1);

```

Nota: Colores basados en el logo de la empresa.

Elaborado por: El autor

En la figura 22, es posible diferenciar los colores primarios, basados en la marca de la empresa y los colores secundarios, para ofrecer una identidad sobre los componentes de la aplicación; esto para lograr una mejor experiencia de usuario con la aplicación.

Creación de Widget personalizados. En el SDK Flutter, los componentes visuales se adaptan con otros creados por parte del desarrollador, pero algunos Widgets creados por el kit de desarrollo no pueden ser suficientes para la exigencia visual provista por el diseño realizado en Figma. Como un ejemplo, se desarrolló un Widget, que permite dar una altura y un ancho a un Widget para dibujar archivos con formato SVG (imágenes realizadas en vector); que el SDK Flutter no proveía.

Figura 23

Widget personalizado

```dart
class KSvgPictureAsset extends StatelessWidget {
  final double width, height;
  final String path;
  final EdgeInsetsGeometry padding, margin;
  final BoxFit boxFit;

  // Constructor
  const KSvgPictureAsset({
    @required this.path,
    @required this.height,
    @required this.width,
    this.padding,
    this.margin,
    this.boxFit
  });

  @override
  Widget build(BuildContext context) {
    return Container(
      width: width,
      height: height,
      padding: padding,
      margin: margin,
      child: SvgPicture.asset(path,
        fit: (boxFit != null) ? boxFit : BoxFit.contain,
        placeholderBuilder: (context) => Center(child: CircularProgressIndicator()),
      ),  // SvgPicture.asset
    );  // Container
  }
}
```

Nota: Widget personalizado creado en la aplicación, para dar una altura y ancho a una clase.

Elaborado por: El autor.

Añadir métodos personalizados a clases establecidas. Dentro de los lenguajes de programación, existen clases que no pueden ser modificadas por los desarrolladores, lo que limita a utilizar métodos personalizados para ciertos casos. Por tal motivo, SDK Flutter provee métodos para añadir nuevos métodos propios a una clase establecida. Por ejemplo, la clase String contiene métodos generalizados para transformar una cadena de caracteres y el añadir un método que permite añadir mayúsculas en cada palabra de una frase. Este ejemplo se encuentra representado en la figura 24, para que la clase String pueda tener este método y utilizarlo en cada invocación.

Figura 24

Método personalizado para la clase String

string_utils.dart

```
extension StringUtils on String {
  String capitalize() {
    return "${this[0].toUpperCase()}${this.substring(1).toLowerCase()}";
  }
}

```

Nota: El método con nombre "capitalize" añade una mayúscula en la primera palabra de la oración.

Elaborado por: el autor.

Esta configuración es posible realizarla por los métodos provisto de SDK Flutter que permite el agregado de métodos propios para unas clases que solo puede ser modificado por el equipo de desarrollo de Flutter. Por lo que, aporta un gran valor para no utilizar nuevos algoritmos complejos para llegar a la misma funcionalidad del método.

3.1.3. Integración de Firebase con SDK Flutter

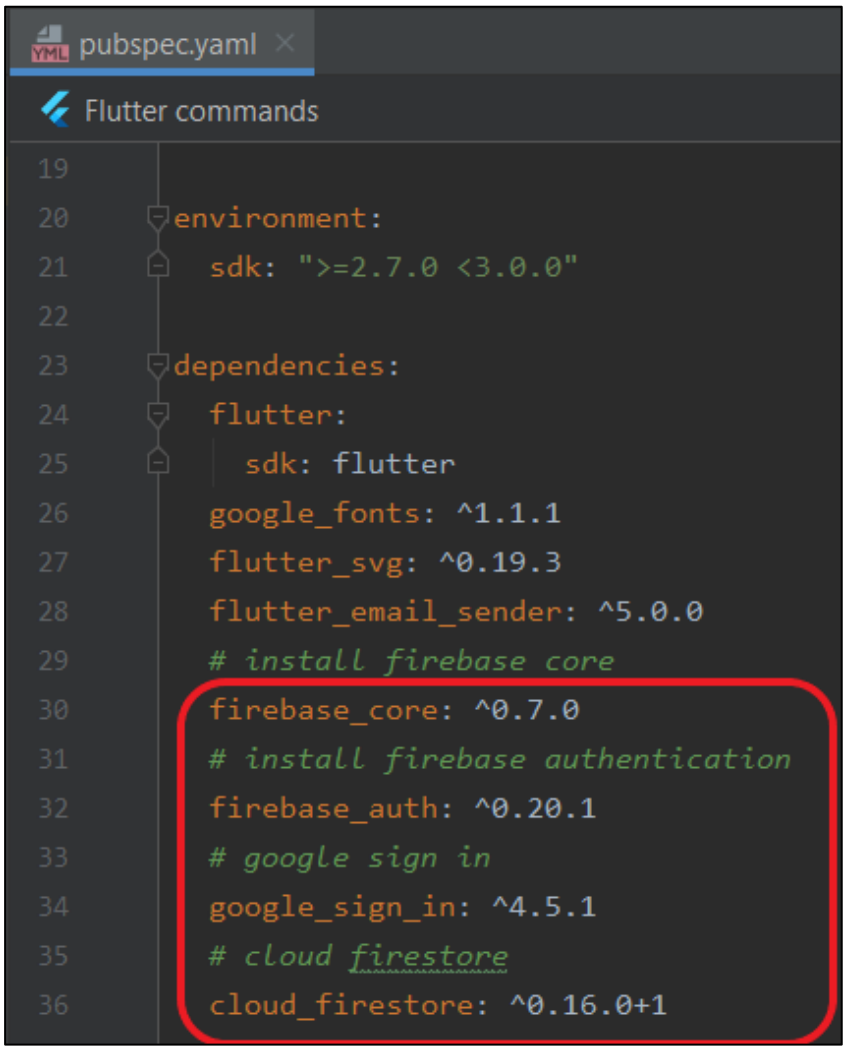
El servicio de Firebase es un servicio de la nube que está basada en la gestión de aplicaciones informáticas de web o móvil, por tal motivo, su integración con los diferentes lenguajes de programación se encuentra diversificado para su implementación con librerías

desarrolladas por cada kit de desarrollo. El equipo de desarrollo del SDK Flutter, se encuentran librerías de Firebase para la integración con este kit de desarrollo, con diferentes clases, como FirebaseAuth, FirebaseFunctios, FirebaseMessaging, etc. (FlutterFire, 2021)

En la figura 25, se indica la integración de las librerías de Firebase para añadir las librerías de cada servicio en el archivo de configuración del proyecto de Flutter, denominado pubspec.yaml, que permite la instancia y las llamadas a estos servicios sin realizar otras configuraciones necesarias.

Figura 25

Librerías de Firebase en el proyecto de Flutter



Nota: Librerías utilizadas para los servicios de Firebase dentro del proyecto de Flutter.

Elaborado por: El autor.

Para la aplicación de Ferro Torre, se utilizan los servicios de Firebase Authentication y Firebase Cloud Firestore, que tiene como base la clase FirebaseApp para crear una instancia de Firebase dentro de la aplicación de Flutter y después instanciarlo a cada servicio de Firebase. En la figura 26, se indica el objeto que contiene las clases para instanciar los diferentes servicios de Firebase utilizados.

Figura 26

Clase que instancia los servicios de Firebase

firebase_factory.dart

```dart
class KFirebaseFactory {
  bool _isConnected;
  KFirebaseFactory() {
    _isConnected = false;
  }

  Future<FirebaseApp> _init() async {
    return await Firebase.initializeApp();
  }

  Future<KFirebaseAuthentication> getAuthentication() {
    return _init()
        .then((value) {
          _isConnected = true;
          return KFirebaseAuthentication();
    }
    ).catchError(
        (error) {
          _isConnected = false;
          return null;
        }
    );
  }

  Future<KFirebaseCloudFireStore> getCloudFireStore() {
    return _init().then((value) {
      _isConnected = true;
      return KFirebaseCloudFireStore();
    }).catchError((error) {
      _isConnected = false;
      return null;
    });
  }
```

Nota: Clase que contiene los servicios de Firebase Authentication y Firebase Cloud Firestore.

Elaborado por: El autor.

Para cada servicio se tiene una clase que realiza las acciones sobre el servicio, por consiguiente, puede ser utilizado por las diferentes pantallas de la aplicación, entre algunos de los métodos importantes para añadir usuarios, vendedores y pedidos. Los métodos mencionados se pueden observar en la figura 27.

Figura 27

Métodos principales para la Firebase Cloud Firestore

```
Future<DocumentReference> addUser(UserFireStore userFireStore) async {
  CollectionReference collectionUser = _firestore.collection(_USER_COLLECTION_NAME);
  return collectionUser.add({...});
}

Future<DocumentReference> addOrder(OrderFireStore orderFireStore) async {
  CollectionReference collectionOrder = _firestore.collection(_ORDER_COLLECTION_NAME);
  return collectionOrder.add({...});
}

Future<DocumentReference> addSeller(SellerFireStore sellerFireStore) async {
  CollectionReference collectionSeller = _firestore.collection(_SELLER_COLLECTION_NAME);
  return collectionSeller.add({...});
}
```

Nota: Métodos para añadir un usuario, vendedor y pedidos.

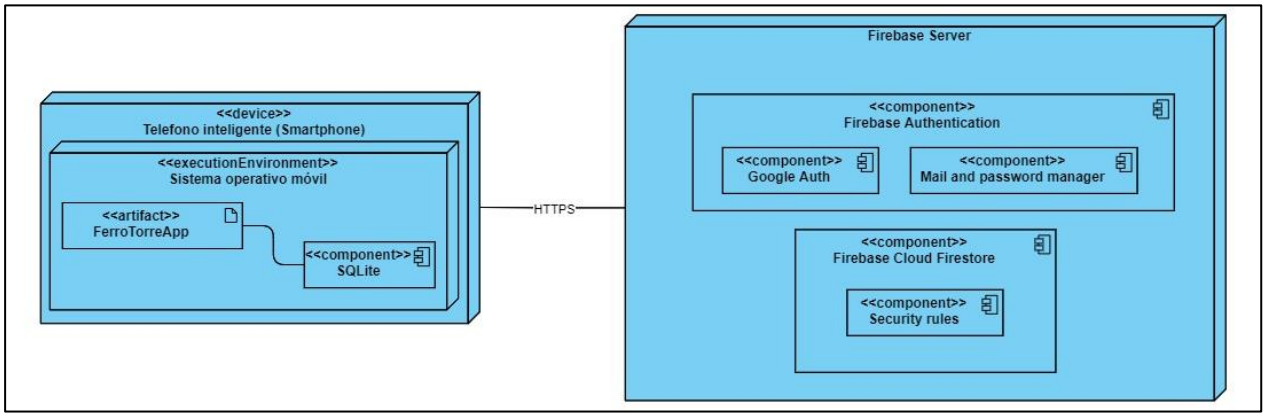
Elaborado por: El autor.

3.1.4. Diagrama de Despliegue

En la figura 20, se indica los componentes que integran en conjunto toda la solución de la aplicación móvil, desde los componentes internos hasta su parte externa o servicios.

Figura 28

Diagrama de Despliegue



Nota: Diagrama que contiene los artefactos internos y externos, al mismo modo su comunicación.

Elaborado por: El autor.

Descripción de los componentes.

Teléfono inteligente (Smartphone). Es el dispositivo móvil que tienen los usuarios y que albergan el sistema operativo, Android o iOS.

Sistema operativo móvil. Es la plataforma que interactúa con el hardware del dispositivo para facilitar la interacción de los componentes con el usuario final y las aplicaciones albergadas.

FerroTorreApp. Es la aplicación desarrollada por este proyecto técnico, que se encuentra dentro del sistema operativo móvil para interactuar con el mismo.

SQLite. Es la base de datos utilizada por las aplicaciones móviles para persistir información de la aplicación.

HTTPS. Es el protocolo de comunicación entre el teléfono celular y el servidor de la nube de Firebase.

Firebase Server. Es la nube de productos para el servicio de Firebase, que contiene los productos a utilizar, según la necesidad de la aplicación.

Firebase Authentication. Es un servicio de Firebase que provee mayor facilidad para la autenticación de usuarios y manejo de cuentas dentro de la aplicación. Dentro de este servicio, se utiliza la autenticación por Google Auth y por correo con contraseña; para el registro de cuentas de usuario.

Firebase Cloud Firestore. Es un servicio de Firebase que permite el almacenamiento de información, de forma escalable, en un formato tipo JSON; en su estructura interna, se encuentra las reglas de seguridad, que permiten gestionar el acceso a la información almacenada según ciertas configuraciones y limitaciones.

3.2. PRUEBAS

El procesamiento de las pruebas sobre la aplicación permite la validación de los componentes internos (código, lógica, datos) y externos (interfaz de usuario, navegación), para ello, se utilizan las pruebas de caja negra y caja blanca que validan ambas estructuras de la aplicación móvil, para la entrega de un producto con mayor calidad a los usuarios. Además de utilizar los emuladores de teléfonos inteligentes para comprar el funcionamiento correcto en cada sistema operativo del dispositivo móvil

3.2.1. Pruebas de Caja Negra

Para gestionar el funcionamiento correcto de la aplicación, se realizan las pruebas de caja negra, que permiten validar el funcionamiento sin la necesidad de conocimiento interno de los componentes de la aplicación.

Por consiguiente, la construcción de las pruebas de caja negra se realiza en base a casos de uso de los requerimientos funcionales de la aplicación. A continuación, se describe un conjunto de casos de pruebas para el requisito funcional con contador 2 (RF2).

Tabla 12

Prueba de caja negra RF2

Requisito funcional	RF2		
Caso de Pruebas			
Caso	Descripción	Entrada	Salida
1	Autenticación de usuario con un correo/contraseña	El usuario ingresa un correo invalido	La aplicación muestra un mensaje de error.
2		El usuario ingresa la contraseña incorrecta	La aplicación muestra un mensaje de error sobre la contraseña incorrecta.
3		El usuario no tiene conexión a Internet	La aplicación indica un mensaje error en el servicio, para reintentar en otra ocasión.
4		El usuario ingresa un correo no registrado previamente	La aplicación muestra un mensaje de usuario no encontrado.
5	Registro de un nuevo usuario con correo/contraseña.	El usuario ingresa un correo ya registrado en la aplicación.	La aplicación indica un mensaje sobre el correo ya fue utilizado por otro usuario para registro.

Nota: Casos de prueba para el RF2.

Elaborado por: El autor.

Se utiliza la tabla 13, para describir los casos de prueba para el RF4, formulario de cotización.

Tabla 13

Prueba de caja negra RF4

Requisito funcional	RF4		
Caso de Pruebas			
Caso	Descripción	Entrada	Salida
1	Formulario de cotización	El usuario no ingresa cualquier campo del formulario	La aplicación indica con un resaltado en rojo, sobre los campos que no están completos.
2		El usuario intenta ingresar letras en el campo de teléfono	La aplicación le muestra un teclado numérico para el campo telefónico.
3		El usuario presiona sobre el botón con signo "-" simultáneamente sin detenerse, para disminuir la cantidad de producto.	La aplicación detecta que se encuentra en el número mínimo de cantidad, y no le permite restar más cantidad de productos, solo hasta el 1.

Nota: Casos de pruebas sobre el formulario de cotización.

Elaborado por: El autor.

Para la tabla 14, se indica el caso de prueba sobre la conexión a internet al intentar descargar un catálogo de productos (RF5).

Tabla 14

Prueba de caja negra RF5

Requisito funcional	RF5		
Caso de Pruebas			
Caso	Descripción	Entrada	Salida
1	Catálogo de productos	El usuario intenta descargar un catálogo de la lista.	La aplicación muestra un pequeño mensaje, sobre el error en la descarga.

Nota: Caso de prueba sobre la falta de conexión a Internet.

Elaborado por: El autor.

Para la tabla 15, se indican los casos de pruebas del requisito funcional con contador 7 (RF7), aplicado al carrito de compras de la aplicación móvil.

Tabla 15

Prueba de caja negra RF7

Requisito funcional	RF7		
Caso de Pruebas			
Caso	Descripción	Entrada	Salida
1	Carrito de compras	El usuario presiona varias veces en el botón con signo "-" para disminuir cantidad	La aplicación no permite disminuir la cantidad del producto a números negativos,

			y lo preserva en la cantidad mínima de 1.
2		El usuario intenta eliminar un producto por equivocación	La aplicación muestra un mensaje de confirmación, sobre el deseo de eliminar el producto, antes de ejecutar la acción.
3		El usuario intenta limpiar todos los productos.	La aplicación muestra un mensaje de confirmación, sobre eliminar todos los productos del carrito de compras.

Nota: Casos de pruebas para el RF7.

Elaborado por: El autor.

3.2.2. Pruebas de Caja Blanca

En el caso de pruebas de caja blanca, se evalúan los componentes internos de la aplicación, es decir, conocimiento del funcionamiento en codificación de los módulos principales que contiene la aplicación. Dado que Flutter SDK y Firebase ofrecen librerías con un control interno completo, la ejecución de pruebas de caja blanca se válida para la lógica que el desarrollador haya utilizado, por tal motivo, se realizo la prueba de caja blanca sobre un control de autenticación dado que sus otros controles se encuentran dentro de las librerías de Flutter SDK y Firebase.

A continuación, se muestra el caso de prueba para crear un usuario con correo y contraseña. En la figura 29, se indica el código sobre un botón con nombre “Crear cuenta”, que permite a la aplicación la creación de un usuario a partir del ingreso de un correo y una contraseña, validando que exista internet y los campos correspondientes estén completos.

Figura 29

Código para registro de cuenta de usuario

```
child: FactoryButtons(name: 'Crear cuenta', type: 2, onPress: () {
  if (firebaseFactory.isConnected()) {
    if (_name.isEmpty || _mail.isEmpty || _pwd.isEmpty) {
      setState(() { this._isInitPage = false; });
    } else {
      firebaseAuth.registerSignInWithEmail(_mail, _pwd, _name).then((value) =>
          _closePageAndReturnToMainPage(context)
      ).catchError((error) =>
          KFirebaseAuthenticationErrorCodes.showDialogMessageError(context, error)
      );
    }
  } else {
    KFirebaseAuthenticationErrorCodes.showDialogMessageError(context,
        KFirebaseAuthenticationErrorCodes.CODE_OPERATION_NOT_ALLOWED);
  }
})  // FactoryButtons
```

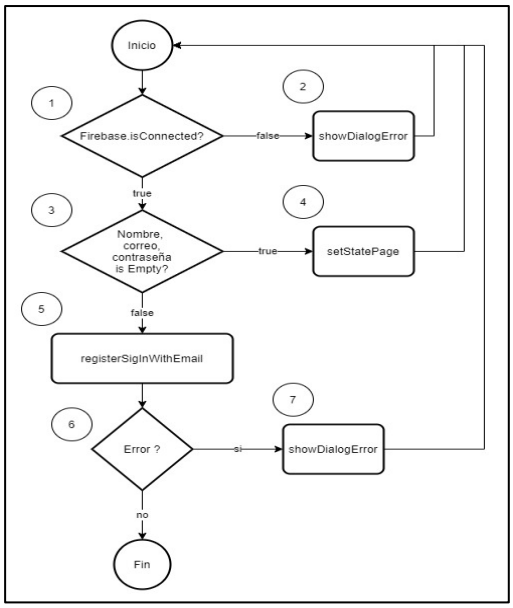
Nota: Código de la aplicación para registro de cuenta con un nombre, correo y contraseña.

Elaborado por: El autor.

Para crear los casos de prueba, es necesario la construcción de un diagrama de flujo para el seguimiento de cada una de las condiciones presentes. En la figura 30, se presenta el diagrama con literales que identifican a cada parte de las funciones.

Figura 30

Diagrama de flujo de código de creación de usuario



Nota: Diagrama con identificadores para cada función del código.

Elaborado por: El autor.

En base a la figura 30, se pueden extraer el conjunto de caminos a seguir para los diferentes que se describen a continuación con el identificador de la letra C y un contador siguiente:

- C1: 0-1-2-0
- C2: 0-1-3-4-0
- C3: 0-1-3-5-6
- C4: 0-1-3-5-6-7

En la tabla 16, se indica los casos para cada uno de los caminos que tiene el diagrama de flujo de la figura 30.

Tabla 16

Casos de pruebas de los caminos de caja blanca

Camino	**Ingreso**	**Resultado Esperado**	**Resultado**
C1	isConnected is false	Mensaje de error	Correcto
C2	isConnected is true y nombre, correo, contraseña estan en vacío	Resalta los inputs con color rojo.	Correcto
C3	isConnected es true, campos completos, registra la función con un error	Mensaje de error	Correcto
C4	isConnected es true, campos completos, registra la función es completada con éxito	Mensaje ingreso correcto a la aplicación	Correcto

Nota: Descripción y resultados de los casos de pruebas.

Elaborado por: El autor

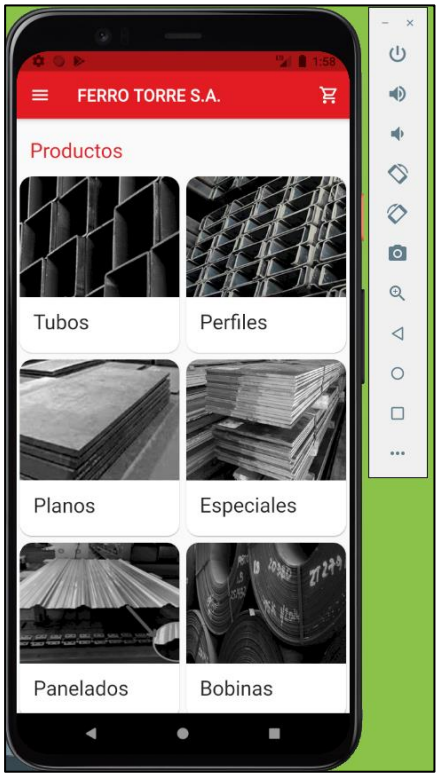
3.2.3. Ejecución en emuladores de Android y iOS

Con los casos de pruebas aprobados y la validación del cliente para la aplicación pueda ejecutarse en las tiendas oficiales de cada sistema operativo, Google Play y Apple Store, surge la necesidad de probar la aplicación en cada plataforma y su correcto funcionamiento. Para ello, se utilizan emuladores para comprobar el funcionamiento correcto de la aplicación en ambas plataformas. Para la ejecución de la aplicación en los emuladores de cada sistema operativo, se utiliza la herramienta de desarrollo denominado Android Studio.

Android. Con Android Studio, la ejecución en un teléfono con sistema operativo Android de versión mínima Lollipop o 5.0, solo es requerido una computadora que pueda ejecutar esta herramienta y su emulador que viene por defecto en el ambiente de desarrollo. En la figura 31, se realizó la captura de pantalla del emulador de Android.

Figura 31

Ejecución de la aplicación en un emulador con Android



Nota: Captura de la aplicación en un emulador Android 8.1.

Elaborado por: El autor.

Sistema operativo iOS. Para la ejecución de la aplicación en el entorno de iOS, es necesario una computadora con sistema operativo MacOS, porque los emuladores solo pueden ser ejecutados en esta plataforma; pero se utiliza la misma herramienta de Android Studio.

Figura 32

Ejecución de la aplicación en emulador con iOS

Nota: Captura de la aplicación en un emulador de un iPhone 12 Pro-Max.

Elaborado por: El autor.

3.3. IMPLEMENTACIÓN

3.3.1. Subida a la tienda de Google Play Store

Con la aprobación y revisión de pruebas para la subida completa de la aplicación, es necesario configurar parámetros importantes para la tienda de Google Play Store. Antes de generar el archivo de aplicación, se necesita la configuración del ID de la aplicación, versión mínima de Android y un código de versión que se debe ir aumentando conforme se realizan actualizaciones.

Figura 33

Archivo build.gradle de Android del proyecto de Flutter

```
defaultConfig {
    applicationId "com.ferrotorre.ferro_torre_app"
    minSdkVersion 21
    targetSdkVersion 29
    versionCode flutterVersionCode.toInteger()
    versionName flutterVersionName
    multiDexEnabled true
}
```

Nota: Parámetros de configuración que requiere la tienda de Google Play Store.

Elaborado por: El autor.

En la tienda ofrecida por Android, es necesario que la aplicación sea firmada por el desarrollador o la entidad que sube la aplicación, para evitar replicas o métodos de ingeniería inversa sobre actualizaciones de la aplicación. Por lo tanto, se crea una llave con un comando que se escribe en la terminal del proyecto, este comando se puede observar en la figura 34, su ejecución.

Figura 34

Comando para creación de la llave de subida

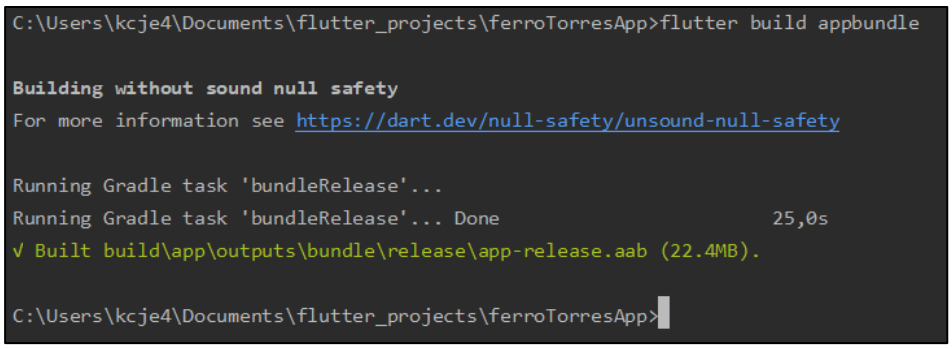
Nota: Se genera un archivo con formato jks para firmar la aplicación.

Elaborado por: El autor.

Esta llave se agrega en el archivo key.propierties (no se sube este archivo en la tienda) y se agregan las respectivas variables de configuración en el archivo con nombre build.gradle. Por último, se genera el archivo que se sube a la tienda de Google Play Store, con la ejecución de otro comando por medio de la consola que permite la creación del archivo Bundle (una mejora de los archivos APK) que se puede observar en la figura 35.

Figura 35

Creación del archivo aab para subir a la tienda



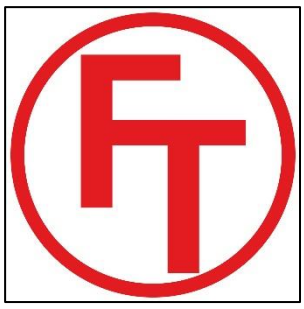
Nota: Comando propio del SDK Flutter, que compila la aplicación para su lanzamiento.

Elaborado por: El autor.

Después de realizadas las configuraciones internas de la aplicación se procede a realizar la parte de imágenes, iconos y marketing de la aplicación dentro de la tienda del Google Play Store. Empezando con el icono de la aplicación, se debe tener una imagen con dimensiones especificadas por la tienda (se utilizar el mismo icono para la tienda de Apple Store). En la figura 36, se puede observar el icono de la aplicación de Ferro Torre, representado por el logo de la empresa con un fondo blanco.

Figura 36

Icono de la aplicación



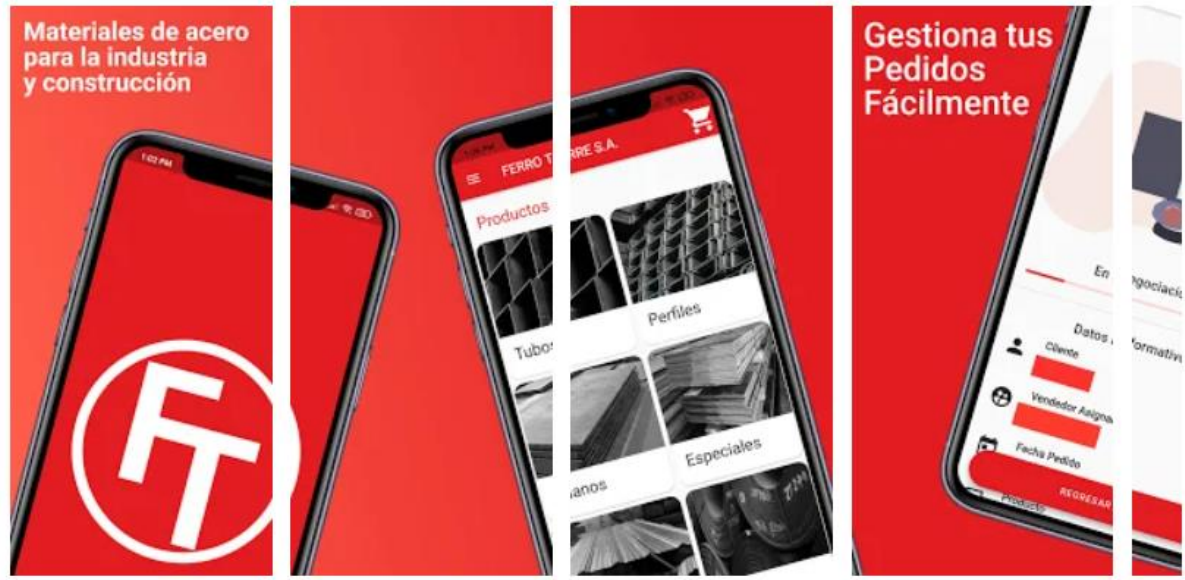
Nota: Icono de la aplicación que se sube en ambas tiendas (Google Play y Apple Store).

Elaborado por: El autor.

Para las tiendas de aplicaciones, se solicita un conjunto de capturas de pantalla que sean de agrado para el usuario que intenta descargar la aplicación. Por consiguiente, se utiliza la página web con nombre Previewed.app, para construir estas capturas de pantalla de forma visual, en la figura 37, se puede observar el resultado visual que se utilizara en ambas tiendas de aplicaciones.

Figura 37

Captura de pantalla para el marketing de la aplicación



Nota: Captura de pantalla que se colocan en la parte de ficha para incentivar la descarga por parte de los usuarios.

Elaborado por: El autor.

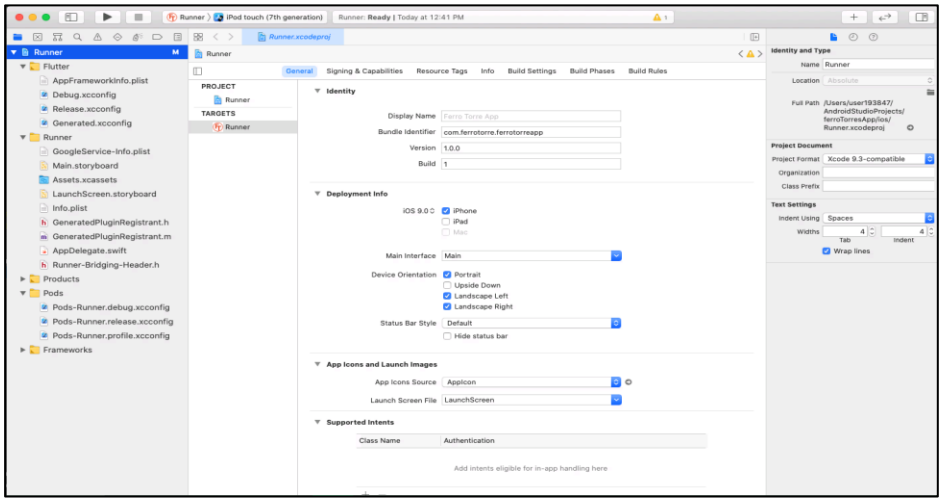
Para la configuración de la tienda y subida de archivos, se puede encontrar en el **Anexo 2 Configuración de Google Play Store.**

3.3.2. Subida a la tienda de Apple Store

Para el proceso de subir la aplicación a la tienda oficial de Apple Store, es necesario tener una computadora con sistema operativo MacOS, porque las configuraciones se realizan en el programa XCode (propio de este sistema operativo). Por consiguiente, en el proyecto de Flutter se debe abrir el programa para configurar los parámetros, como el nombre, la versión mínima del sistema operativo y el código de versión de la aplicación. En la figura 38, se puede observar la configuración realizada.

Figura 38

Configuración de información para la aplicación en iOS



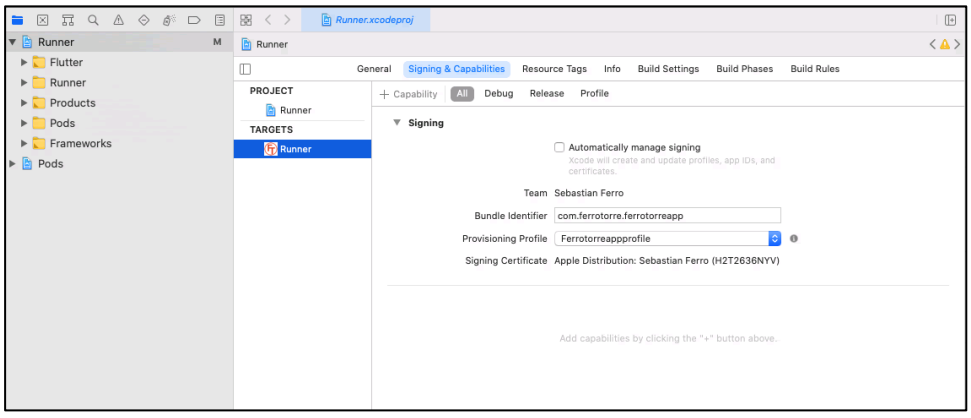
Nota: Configuración de datos importantes para la aplicación que se ejecuta en iOS.

Elaborado por: El autor.

Del mismo modo que en la tienda de Google Play Store de Android, es necesario que la aplicación sea firmada por la cuenta de desarrollador para subir a la tienda. Por lo tanto, se instancia la cuenta de desarrollador con el programa XCode y se utiliza un certificado creado en la página de Apple developer. En la figura 39, se puede observar la firma de la aplicación.

Figura 39

Firma de la aplicación por la cuenta de desarrollador



Nota: El certificado debe ser instalado en una computadora MacOS.

Elaborado por: El autor.

Al finalizar el proceso de firma, se procede a generar el archivo de la aplicación que utiliza XCode para subir la aplicación en el Apple Store. En la figura 40, se puede observar el comando ejecutado en el terminal para generar este archivo.

Figura 40

Creación de archivo para subir a la tienda de Apple Store

```
telehouse-moboware:ferroTorresApp user197914$ flutter build ios --release
Building com.ferrotorre.ferrotorreapp for device (ios-release)...
Automatically signing iOS for device deployment using specified development team in Xcode project: H2T2636NYV
Running pod install...                                          15.8s
Running Xcode build...
 └─Compiling, linking and signing...                         20.9s
Xcode build done.                                           652.3s
Built /Users/user197914/StudioProjects/ferroTorresApp/build/ios/iphoneos.
telehouse-moboware:ferroTorresApp user197914$
```

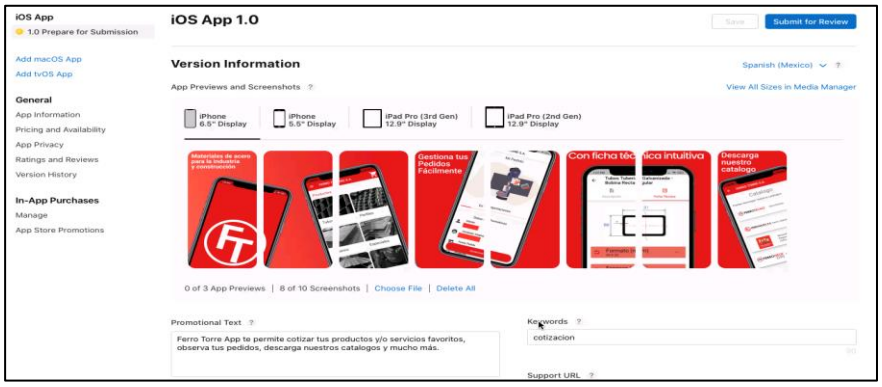
Nota: Ejecución de comando para generar la aplicación de producción.

Elaborado por: El autor.

El proceso para subir la aplicación en la página de desarrollador se puede encontrar en el **<u>Anexo 3 Configuración de tienda de Apple Store</u>**, del mismo modo, para realizar el marketing de la aplicación, se utiliza las capturas de pantalla provista en el punto 3.3.2. para la parte visual en la tienda del Apple Store. En la figura 41, se puede observar una captura de la página web de Apple Store developer para subir la aplicación.

Figura 41

Captura de pantalla para marketing en el Apple Store



Nota: Información de la aplicación, que el usuario observa en la tienda de Apple Store.

Elaborado por: El autor.

CONCLUSIONES

- Para la ejecución del proyecto en el SDK Flutter, es necesario solo ciertos archivos de configuración y los editores de código que soportan este kit de desarrollo permiten la creación del proyecto con sus archivos de manera muy sencilla, para empezar a desarrollar con facilidad.
- Conforme la tecnología va madurando contantemente, se vuelve aún más sencillo importar librerías de terceros sobre el lenguaje de programación y así facilitando su salida al mercado. Las aplicaciones multiplataforma permiten a las empresas utilizar el mismo código de programación con el fin de escalar hacía nuevos sistemas operativos.
- El SDK Flutter permite la creación de archivos con el lenguaje de programación Dart, que pueden realizar diferentes tareas, tales como la creación de la interfaz de usuario, manejo de recursos, colores, cadena de caracteres y proveedores de información para la aplicación. La conexión que existen entre ellos es sencilla para el desarrollador porque solo la importación es manejada por el SDK para un funcionamiento con normalidad de ambas partes.
- La subida de la aplicación en las tiendas oficiales, Google Play Store y Apple Store, resulta en una configuración del código nativo para ambas plataformas que soporta la aplicación, por tanto, se considera necesario el conocimiento en ambos sistemas operativos para realizar un despliegue eficaz en las tiendas de aplicaciones móviles. La ejecución de las pruebas en los diferentes sistemas operativos resulta de mayor facilidad porque se encuentra en el mismo código para ambas plataformas. El mismo SDK Flutter se encarga de interactuar con los componentes de hardware y software para que la aplicación se pueda utilizar en ambas plataformas sin código adicional.
- Con la integración de diferentes medios de comunicación, utilizados por la empresa, junto con las facilidades de interacción que ofrece la aplicación, resulta en un mayor aporte de

valor para los clientes porque mantienen al usuario activo dentro de la aplicación donde se provee la información que se necesita.

RECOMENDACIONES

- Para desarrollar en el SDK Flutter, es necesario conocer sobre la programación declarativa porque la construcción de la interfaz de usuario se basa completamente en este esquema de desarrollo.
- El servicio de Firebase promete una gran escalabilidad para las aplicaciones móviles, sin embargo, cuando la aplicación se convierte en un sistema completo (servicio web, aplicación de escritorio, etc.) se convierte en una tarea difícil, debido a que, el servicio de esta nube queda limitada por la falta de herramientas más específicas, que otros proveedores si poseen.
- La elaboración de un prototipo y un diseño previo resulta clave para reducir la incertidumbre de cambios que puede dar el cliente al producto, por tanto, es recomendable utilizar la construcción de diseño simples para validar los requisitos funcionales de la aplicación para cuando se proceda a construir una aplicación con mayor calidad en la interfaz de usuario.

LISTA DE REFERENCIAS

Bibliografía

Blanquer, E. B. (Madrid). *Desarrollo de iOS con Swift.* 2016: RA-MA Editorial.

Peter, D., Gabrielle, B., Craig, L., & Bas, V. (2009). Información básica de scrum (The scrum primer). *Scrum Training Institute*, 20.

Sitio Web

Android Developers. (03 de 05 de 2021). *Material Design para Android*. Obtenido de https://developer.android.com/guide/topics/ui/look-and-feel?hl=Es-419

Ferro torre. (06 de mar de 2021). *Página Oficial*. Recuperado el 06 de mar de 2021, de Quienes somos - Nuestra historia: https://www.ferrotorre.com/

Firebase. (03 de Mayo de 2021). Recuperado el 03 de Mayo de 2021, de https://firebase.google.com/

Firebase. (17 de 05 de 2021). *Firebase Cloud Firestore*. Recuperado el 17 de 05 de 2021, de https://firebase.google.com/products/firestore

Firebase Cloud Firestore. (17 de 05 de 2021). *Elige una base de datos: Cloud Firestore o Realtime Database*. Obtenido de https://firebase.google.com/docs/database/rtdb-vs-firestore

Flutter. (3 de Mayo de 2021). *Hot reload: Flutter.dev*. Recuperado el 3 de Mayo de 2021, de Flutter.dev: https://flutter.dev/docs/development/tools/hot-reload

FlutterFire. (06 de Junio de 2021). *FlutterFire Overview*. Obtenido de https://firebase.flutter.dev/

Mancuzo, G. (21 de ago de 2020). *Programación Extrema: Pros y Contras*. Obtenido de https://blog.comparasoftware.com/programacion-extrema-ventajas-desventajas/#:~:text=Fuerte%20dependencia%20de%20las%20personas,adelante%20hay%20que%20pedir%20explicaciones.

Tesis

Apolinario, C. A. (2020). *Diseño e implementación de un módulo de gestión de calidad de labores agrícolas de la empresa delindecsa ubicada en la ciudad de Guayaquil.* Guayaquil: Universidad Católica de Santiago de Guayaquil.

Benavides, D. A. (2017). *Sistema ERP basado en inteligencia de negocios para el proceso de administración y toma de decisiones de la distribuidora dimaco de la ciudad de Tulcán.* Tulcán: Universidad Regional Autónoma de los Andes Uniandes.

Coral, I. C., & Iza, C. A. (2018). *Análisis, diseño y construcción de un sistema E-Commerce para web y dispositivo Android.* Quito: Universidad Politécnica Salesiana.

Daniel, A. C. (2020). *Aplicación para crear un contacto entre emprendedores e inversores (VVENTURE).* Quito: Universidad San Francisco de Quito USFQ.

Guaman, G. E., & Orquera, F. S. (2020). *DESARROLLO DE UN BOTÓN DE SOCORRO EN UNA PLATAFORMA MÓVIL.* Quito: Universidad Politécnica Salesiana.

Lisandro, N. D. (2017). *Desarrollo de Aplicaciones móviles multiplataforma.* La Plata: Universidad Nacional de La Plata.

Morata, J. Q. (2011). *Desarrollo de una aplicación distribuida para dispositivos iOS.* Valencia: Universidad Politécnica de Valencia.

Mora, D. A., & Apolinar, D. A. (2015). *Desarrollo de un prototipo de una aplicación para dispositivos móviles para el acceso a información turística detallada de algunos puntos de interés de la ciudad. Caso de estudio: Edificio el Claustro de la Universidad Católica de Colombia (Beacon City).* Bogota D.C.: Universidad Católica de Colombia.

Pérez, N. A., & Torres, B. D. (2020). *Desarrollo de un prototipo para la geolocalización y monotorización de frecuencia cardiaca de mascotas en la ciudad de Quito.* Quito: Universidad Politécnica Salesiana.

Pérez, O. A. (2011). *Cuatro enfoques metodológicos para el desarrollo de Software RUP – MSF – XP - SCRUM.* Bogota: Facultad de Ingeniería UNIMINUTO.

Roche, J. P., & Suarez, J. M. (2009). *Análisis, diseño, e implementación de un software, para la administración de los proyectos de grado en el programa de ingeniería de sistemas, aplicando una metodología ágil.* Pereira: Universidad Tecnológica de Pereira.

Torres, J. P. (2021). *Aplicación móvil multiplataforma para la gestión de información georeferencial y servicio técnico comunitario de plomería, aplicando geolocalización offline, en la junta administradora de agua potable de los barrios occidentales de aloasí.* Ambato: Universidad Técnica de Ambato.

Villegas Salazar, K. A., & Loor Muñoz, K. A. (2020). *Desarrollo de una aplicación móvil utilizando flutter sdk de google para promocionar el arte de la ciudad de Guayaquil.* Guayaquil: Universidad de Guayaquil.