Neiva, 28 de octubre de 2019

Señores

CENTRO DE INFORMACIÓN Y DOCUMENTACIÓN

UNIVERSIDAD SURCOLOMBIANA

Ciudad

El (Los) suscrito(s):

Luis Felipe Bustos Murcia, con C.C. No.1.075.293.947,

Nixon Raúl Gamboa, con C.C. No. 1.075.302.469

Autor(es) de la tesis y/o trabajo de grado o Proyecto de grado

Titulado Diseño de una aplicación móvil como marketing digital para la logística de celebraciones especiales presentado y aprobado en el año 2019 como requisito para optar al título de Ingeniero Electrónico

Autorizo (amos) al CENTRO DE INFORMACIÓN Y DOCUMENTACIÓN de la Universidad Surcolombiana para que, con fines académicos, muestre al país y el exterior la producción intelectual de la Universidad Surcolombiana, a través de la visibilidad de su contenido de la siguiente manera:

- Los usuarios puedan consultar el contenido de este trabajo de grado en los sitios web que administra la Universidad, en bases de datos, repositorio digital, catálogos y en otros sitios web, redes y sistemas de información nacionales e internacionales “open access” y en las redes de información con las cuales tenga convenio la Institución.

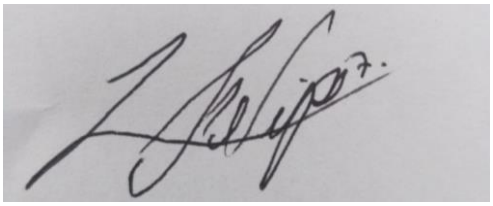
- Permita la consulta, la reproducción y préstamo a los usuarios interesados en el contenido de este trabajo, para todos los usos que tengan finalidad académica, ya sea en formato Cd-Rom o digital desde internet, intranet, etc., y en general para cualquier formato conocido o por conocer, dentro de los términos establecidos en la Ley 23 de 1982, Ley 44 de 1993, Decisión Andina 351 de 1993, Decreto 460 de 1995 y demás normas generales sobre la materia.

- Continúo conservando los correspondientes derechos sin modificación o restricción alguna; puesto que, de acuerdo con la legislación colombiana aplicable, el presente es un acuerdo jurídico que en ningún caso conlleva la enajenación del derecho de autor y sus conexos.

De conformidad con lo establecido en el artículo 30 de la Ley 23 de 1982 y el artículo 11 de la Decisión Andina 351 de 1993, “Los derechos morales sobre el trabajo son propiedad de los autores”, los cuales son irrenunciables, imprescriptibles, inembargables e inalienables.

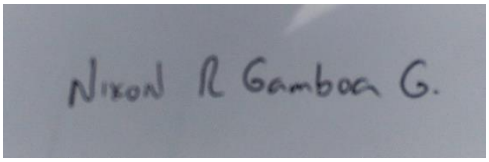
EL AUTOR/ESTUDIANTE: Luis Felipe Bustos

Firma:



EL AUTOR/ESTUDIANTE: Nixon Raúl Gamboa

Firma:



TÍTULO COMPLETO DEL TRABAJO: DISEÑO DE UNA APLICACIÓN MÓVIL COMO MARKETING DIGITAL PARA LA LOGÍSITCA DE CELEBRACIONES ESPECIALES

AUTOR O AUTORES:

Primero y Segundo Apellido	Primero y Segundo Nombre
Bustos Murcia Gamboa	Luis Felipe Nixon Raúl

DIRECTOR Y CODIRECTOR TESIS:

Primero y Segundo Apellido	Primero y Segundo Nombre
Molina Mosquera	Johan Julián

ASESOR (ES):

Primero y Segundo Apellido	Primero y Segundo Nombre

PARA OPTAR AL TÍTULO DE: Ingeniero Electrónico

FACULTAD: Facultad de Ingeniería

PROGRAMA O POSGRADO: Electrónica

CIUDAD: Neiva **AÑO DE PRESENTACIÓN:** 2019 **NÚMERO DE PÁGINAS:** 95

TIPO DE ILUSTRACIONES (Marcar con una **X**)**:**

Diagramas__X_ Fotografías_X__ Grabaciones en discos___ Ilustraciones en general___ Grabados___ Láminas___ Litografías___ Mapas___ Música impresa___ Planos___ Retratos___ Sin ilustraciones___ Tablas o Cuadros_X_

Vigilada Mineducación
La versión vigente y controlada de este documento, solo podrá ser consultada a través del sitio web Institucional www.usco.edu.co, link Sistema Gestión de Calidad. La copia o impresión diferente a la publicada, será considerada como documento no controlado y su uso indebido no es de responsabilidad de la Universidad Surcolombiana.

SOFTWARE requerido y/o especializado para la lectura del documento:

MATERIAL ANEXO:

PREMIO O DISTINCIÓN *(En caso de ser LAUREADAS o Meritoria)*:

PALABRAS CLAVES EN ESPAÑOL E INGLÉS:

Español	Inglés	Español	Inglés
1. Diseño	Design	**4.** Frameworks	Frameworks
2. Interfaz	Interface	**5.** Firebase	Firebase
3. React native	React native	**6.** Tendencias	Trends

RESUMEN DEL CONTENIDO: (Máximo 250 palabras)

El presente proyecto de grado es el diseño de una aplicación móvil como marketing digital para la logística de celebraciones especiales, desarrollada para facilitar el contacto con los clientes, donde éstos tendrán acceso al amplio portafolio de productos y servicios ofrecidos, así como sus promociones, descuentos, etc., por ´ultimo realizar su respectivo registro de compra y obtener su producto de una manera fácil y sencilla. Para ésto se tuvieron en cuenta dos aspectos importantes de los usuarios a la hora de realizar compras: Tendencias y Precios, los cuales se constituyen en los puntos básicos para el diseño de la App.

La ejecución del proyecto se ha realizado sobre la herramienta de software Android Studio. La aplicación está compuesta de dos partes el frontend y el backend: para el desarrollo del frontend o interfaz se ha utilizado el framework de aplicaciones reactivas web pero en el mundo nativo conocido como react-native y para el backend se ha hecho uso de la nueva y mejorada plataforma de desarrollo móvil en la nube de Google conocida como Firebase.

La aplicación proporcionará información a los usuarios para que puedan filtrar sus compras según sea la tendencia por la que otros clientes se han inclinado, también desplegará una lista de categorías en donde en cada una de ellas se encontrarán los productos y servicios ofrecidos, el cliente escogerá el producto de su interés y en la parte de notas adicionales,

Vigilada Mineducación
La versión vigente y controlada de este documento, solo podrá ser consultada a través del sitio web Institucional www.usco.edu.co, link Sistema Gestión de Calidad. La copia o impresión diferente a la publicada, será considerada como documento no controlado y su uso indebido no es de responsabilidad de la Universidad Surcolombiana.

indicará como quiere el producto, por último, seleccionará la fecha que lo desea y el método de pago.

ABSTRACT: (Máximo 250 palabras)

This degree project is the design of a mobile application as digital marketing for the logistics of special celebrations, developed to facilitate contact with customers, where they will have access to the wide portfolio of products and services offered, as well as their promotions, discounts, etc.., finally make their respective purchase registration and obtain their product in an easy and simple. Two important aspects of users were taken into account when making purchases: Trends and Prices, which are the basic points for the design of the App.

The project was executed using the Android Studio software tool. The application is composed of two parts the frontend and the backend: for the development of the frontend or interface has been used the framework of reactive web applications but in the native world known as react-native and for the backend has made use of the new and improved mobile development platform in the Google cloud known as Firebase.

Detallitos App will provide information to users so that they can filter their purchases according to the trend that other customers have leaned towards. It will also display a list of categories in which each of them will find the products and services offered, the customer will choose the product of their interest and in the part of additional notes will tell us how the product wants, finally select the date you want and the method of payment.

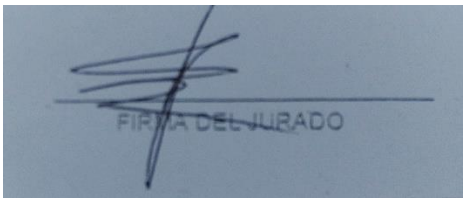
APROBACION DE LA TESIS

Nombre Jurado: Martín Bravo Obando

Firma:

Nombre Jurado: Jesús David Quintero

Firma:



Vigilada Mineducación
La versión vigente y controlada de este documento, solo podrá ser consultada a través del sitio web Institucional www.usco.edu.co, link Sistema Gestión de Calidad. La copia o impresión diferente a la publicada, será considerada como documento no controlado y su uso indebido no es de responsabilidad de la Universidad Surcolombiana.

DISEÑO DE UNA APLICACIÓN MÓVIL COMO MARKETING DIGITAL PARA LA LOGÍSITCA DE CELEBRACIONES ESPECIALES

NIXON RAÚL GAMBOA GARCÍA
LUIS FELIPE BUSTOS MURCIA

UNIVERSIDAD SURCOLOMBIANA
FACULTAD DE INGENIERÍA
PROGRAMA DE INGENIERÍA ELECTRÓNICA
NEIVA
2019

DISEÑO DE UNA APLICACIÓN MÓVIL COMO MARKETING DIGITAL PARA LA LOGÍSITCA DE CELEBRACIONES ESPECIALES

NIXON RAÚL GAMBOA GARCÍA
LUIS FELIPE BUSTOS MURCIA

Trabajo De Grado Para Optar Al Título De Ingeniero Electrónico

Director
Johan Julián Molina Mosquera
Mag.

UNIVERSIDAD SURCOLOMBIANA
FACULTAD DE INGENIERÍA
PROGRAMA DE INGENIERÍA ELECTRÓNICA
NEIVA
2019

Nota de aceptación:

_____

_____

_____

_____

_____

_____

_____

_____

Firma del presidente del Jurado

_____

Firma del Jurado

_____

Firma del Jurado

Neiva, Octubre 19 de 2019

DEDICATORIA

Mi tesis la dedico a mis padres Henry Gamboa Tovar y Maria Antonia Garcia por creer en mis capacidades, ellos por su sacrificio y esfuerzo merecen este logro tanto como yo, a mis abuelos Reinaldo Gamboa y Lazaro Garcia quienes siempre me acompañan desde el cielo, a mis abuelas Antonia Rodriguez y Florentina Tovar que por la gracia de Dios hoy me ven culminar esta etapa. A mi hermano Cristian Javier Peña Garcia con quien espero compartir los frutos de mi vida. A mis primos y amigos les dedico este ejemplo para sus vidas. A toda mi familia que en un momento u otro han estado junto a mi para hacer esto posible. A mis amigos de universidad y futuros colegas para quienes deseo éxitos y buenaventura en sus vidas.

Nixon Raúl Gamboa García

A Dios, por haberme permitido llegar hasta este punto y haberme dado el conocimiento necesario para lograr cada objetivo propuesto en mi vida. A mi mamá María Gladys Murcia por su amor, enseñanza, aprecio, perseverancia y por creer en mi. A mi padre Luis Eduardo Bustos Díaz por los ejemplos de lealtad, perseverancia, constancia y entrega. A mi abuelo Damían Bustos y abuela Nelly Murcia por su agran precio y cariño que me han brindado. Al ingeniero Julian Molina por su amabilidad, su conocimiento y su guía durante el proyecto. A los maestros, ingenieros y trabajadores de la facultad por su acompañamiento en este gran camino. A mis AMIGOS, que nos apoyamos mutuamente en nuestra formación profesional y personal

Luis Felipe Bustos Murcia

AGRADECIMIENTOS

Quiero agradecer a Dios. A mi padre, Henry Gamboa Tovar por su infinito apoyo, su increíble manera de motivarme día a día, porque gracias a él empece este camino y pude culminarlo. A mi madre, Maria Antonia Garcia Rodriguez, por su amor y cariño, porque me vio caer muchas veces y siempre estuvo ahi para levantarme, porque cada fin de semestre me recibía en casa para renovar mis fuerzas como solo una madre lo puede hacer, agradezco a ellos por que incondicionalmente han estado en mi vida y han dado su máximo para hacer este logro también suyo. A mis primas Lucy Tovar y Fabiola Acosta quienes me apoyaron en momentos claves de la carrera. Al Ingeniero Julian Molina por sus enseñanzas y por aceptar dirigirnos a lo largo de este proyecto, a mi compañero de trabajo Luis Felipe Bustos por su grandiosa amistad, con quien compartí grandes momentos dentro y fuera del plantel educativo, a la plantilla de ingenieros del programa de Ingeniería Electrónica y a los docentes de la Universidad Surcolombiana por compartir sus conocimientos e intentar hacer de este un mundo mejor.

Nixon Raúl Gamboa García

A mis padres, hermanos, sobrinos y mis abuelos por apoyarme en todo momento, por lo que me han inculcado, por creer en mí, por ser parte fundamental en mi vida y representar la unidad familiar. Al ingeniero Julian Molina por creer en nosotros. A la Universidad Surcolombiana por el apoyo y facilidades otorgadas. A mi compañero de tesis Nixon Gamboa con quien he recorrido a lo largo de la carrera y por el trabajo realizado en conjunto. A los ingenieros de la facultad por haberme brindado sus conocimientos en un mundo de constantes cambios. A mis amigos y futuros colegas. A todas y cada una de las personas con quienes he compartido a lo largo de mi carrera.

Luis Felipe Bustos Murcia

TABLA DE CONTENIDOS

Pág

LISTA DE FIGURAS

Pág

LISTA DE ANEXOS

Pág

GLOSARIO

Android: Sistema operativo creado por Google para celulares.
Android Studio: Entorno de desarrollo para crear aplicaciones de Android
API: Aplicación que funciona como interfaz de programación, la cual es usada como biblioteca y proporciona interoperabilidad entre sistemas.
APK: Application Package, archivo ejecutable de Android generado después de la compilación.
App: Aplicaciones móviles desarrolladas con los lenguajes oficiales de un sistema operativo y soportados por el propietario. Corren en dispositivos móviles como celulares o tablets.
AVD: Android Virtual Device, emula un dispositivo Android.
Backend: Parte del software que actúa fuera de los ojos del usuario, en el desarrollo web es la capa que se encarga de manejar la información y las reglas del negocio.
Base de Datos: Son contenedores o repositorios digitales en los que se almacenan datos de manera estructurada y organizada para su posterior procesamiento y consulta.
CLI: Command Line Interface o interfaz de linea de comandos, permite al usuario interactuar con un programa enviando comandos de texto haciendo uso de la terminal o consola del sistema operativo.
ES6: ECMAScript v6 es el estándar que sigue JavaScript desde Junio de 2015.
Firebase: Es el servicio de backend de Google utilizado para el desarrollo Web y móvil.
Frontend: Conjunto de tecnologías que dan estructura, apariencia y comportamientos de interactividad a los sitios y aplicaciones web modernas de cara a los clientes o usuarios finales en un navegador.
Framework: Serie de herramientas/librerías de programación con un proposito especifico
JDK: Son las siglas del Java Development Kit este es el kit de desarrollo de Java que contiene todas las clases y librerías necesarias para crear propios programas en Java.
Json: JavaScript Object Notation es un formato ligero de intercambio de datos.
Método: es un conjunto de instrucciones definidas dentro de una clase, que realizan una determinada tarea y a las que podemos invocar mediante un nombre.
NodeJS: Es el entorno de ejecución de Javascript en el lado del servidor, está basado en el motor V8 de Chrome para ejecutar el código.
NoSQL: Los datos almacenados no requieren estructuras fijas como tablas y utilizan una variedad de modelos de datos para acceder y administrar datos, como documentos, gráficos, clave-valor, en-memoria y búsqueda.
npm: Node Package Manager o simplemente npm es un gestor de paquetes, el cual hará más fácil trabajar con Node, porque se puede tener cualquier librería disponible con solo

una línea de código, npm también ayudará a administrar los módulos, distribuir paquetes y agregar dependencias de una manera sencilla.

Objeto: Instancia de la clase, forma general del objeto con datos particulares.

Usabilidad: Se refiere a la facilidad de uso de un sitio web para todos los visitantes, no solo para las personas discapacitadas. Por ejemplo, un sitio que requiere un registro prolongado para ver su contenido tiene mala usabilidad.

Usuario: Persona que está en la facultad de usar un producto o un servicio, puede o no pagar por él, y se encarga de darle permanencia mediante su uso y así suplir una necesidad.

UI: Interfaz de usuario, es la vista que permite a un usuario interáctuar de manera efectiva con un sistema.

RESUMEN

El escenario tecnológico actual posiciona a los desarrolladores de servicios como la nueva fuerza de negocios en las redes de próxima generación Application Stores, debido al crecimiento exponencial de las tecnologías de comunicación. Allí radica la importancia de la mejora en las nuevas aplicaciones de compra/entrega de productos y servicios.

El presente proyecto de grado es el diseño de una aplicación móvil como marketing digital para la logística de celebraciones especiales, desarrollada para facilitar el contacto con los clientes, donde éstos tendrán acceso al amplio portafolio de productos y servicios ofrecidos, así como sus promociones, descuentos, etc., por último realizar su respectivo registro de compra y obtener su producto de una manera fácil y sencilla. Para ésto se tuvieron en cuenta dos aspectos importantes de los usuarios a la hora de realizar compras: Tendencias y Precios, los cuales se constituyen en los puntos básicos para el diseño de la App.

La ejecución del proyecto se ha realizado sobre la herramienta de software Android Studio. La aplicación esta compuesta de dos partes el frontend y el backend: para el desarrollo del frontend o interfaz se ha utilizado el framework de aplicaciones reactivas web pero en el mundo nativo conocido como react-native y para el backend se ha hecho uso de la nueva y mejorada plataforma de desarrollo móvil en la nube de Google conocida como Firebase.

Detallitos App proporcionará información a los usuarios para que puedan filtrar sus compras según sea la tendencia por la que otros clientes se han inclinado, también desplegará una lista de categorías en donde en cada una de ellas se encontrarán los productos y servicios ofrecidos, el cliente escogerá el producto de su interés y en la parte de notas adicionales nos indicará como quiere el producto, por ultimo seleccionará la fecha que lo desea y el método de pago.

PALABRAS CLAVES

Aplicación móvil, diseño, interfaz, react native, Frameworks, firebase, tendencias, internet.

ABSTRACT

The current technology scenario positions service developers as the new business force in next-generation Application Stores networks due to the exponential growth of communication technologies. Therein lies the importance of the improvement in the new applications of purchase/delivery of products and services.

This degree project is the design of a mobile application as digital marketing for the logistics of special celebrations, developed to facilitate contact with customers, where they will have access to the wide portfolio of products and services offered, as well as their promotions, discounts, etc.., finally make their respective purchase registration and obtain their product in an easy and simple. Two important aspects of users were taken into account when making purchases: Trends and Prices, which are the basic points for the design of the App.

The project was executed using the Android Studio software tool. The application is composed of two parts the frontend and the backend: for the development of the frontend or interface has been used the framework of reactive web applications but in the native world known as react-native and for the backend has made use of the new and improved mobile development platform in the Google cloud known as Firebase.

Detallitos App will provide information to users so that they can filter their purchases according to the trend that other customers have leaned towards. It will also display a list of categories in which each of them will find the products and services offered, the customer will choose the product of their interest and in the part of additional notes will tell us how the product wants, finally select the date you want and the method of payment.

KEYWORDS

Mobile application, design, interface, react native, Frameworks, firebase, trends, internet.

1. INTRODUCCIÓN

En la última década, el uso de la telefonía móvil en Latinoamérica ha tenido un creciente auge; la disminución de los costos en los equipos y la evolución de las tecnologías móviles han permitido aumentar el número de usuarios y las velocidades de transferencia de información. La disponibilidad de las mejoras tecnológicas, tanto en las redes de los operadores como en los dispositivos móviles, están creando en las personas la demanda del uso de nuevos servicios. Servicios que se basan en estas tecnologías para solucionar problemas específicos[1]. Es gracias al aumento en el uso de dispositivos móviles que los expertos y entusiastas de la tecnología vieron el potencial de dicha tecnología en el desarrollo de soluciones orientadas a la movilidad; naciendo de esta forma el nuevo campo del desarrollo de aplicaciones para dispositivos móviles[2].

Actualmente en el mercado local no se ofrecen servicios en aplicaciones móviles sobre logística de celebraciones especiales, sino que utilizan muchos métodos ortodoxos o poco prácticos y a veces resulta difícil para el cliente seleccionar un producto de su interés. De acuerdo a esto, se estudiaron distintas metodologías para fomentar el desarrollo de la aplicación, abarcando una línea de actividades, la cual comprende: toma de requisitos, especificación, diseño, pruebas e implementación.

El propósito de este proyecto, tiene como fin implementar una aplicación móvil para dispositivos Android, en la que se ofrece el servicio de logística de celebraciones especiales, con una interfaz muy dinámica y con una base de datos hecha en Firebase para el almacenamiento y la gestión.

[1]Mantilla, Ariza y Delgado, "Metodología para el desarrollo de aplicaciones móviles".

[2]Sequeira Plama, "Aplicacion movil bajo la plataforma Android para la suscripcion y notificaciones de la Asociacion de Emprendedores de la comunidad de renovacion familiar" HOSANNA", año 2014".

2. OBJETIVOS

2.1. OBJETIVO GENERAL

Diseñar una aplicación móvil con el fin de impulsar el comercio electrónico en la compra y entrega de detalles ofreciendo un amplio portafolio de productos y servicios al cliente para la logística de festividades sociales en la ciudad de Neiva.

2.2. OBJETIVOS ESPECÍFICOS

- Analizar las características y los requerimientos que debe tener la aplicación móvil, según las tecnologías de desarrollo nativo que hay en el mercado.
- Diseñar la arquitectura de la aplicación móvil mediante diagramas o esquemas teniendo en cuenta los aspectos técnicos y funcionales.
- Crear la base datos de la aplicación móvil estudiando los recursos de software necesarios para su desarrollo qué supla los requerimientos de almacenamiento y gestión de datos.
- Poner en funcionamiento una aplicación móvil que contenga el módulo de catálogo de los productos ofrecidos y descuentos.

3. MARCO TEÓRICO

3.1. ANTECEDENTES

Las primeras aplicaciones móviles datan de finales de los 90s estas eran lo que conocemos como la agenda, árcade games, Los editores de ringtone, etc. cumplían funciones muy elementales y su diseño era bastante simple[3].

No era preciso que las aplicaciones tuvieran una evolución tan rápida. La historia comienza cuando Apple saca al mercado su teléfono IPhone y una vez la competencia ve el producto en el mercado, comienzan a aparecer otras ofertas de Smartphones intentado simular la calidad, es así como aparece con fuerza Android, ganando terreno rápidamente hasta constituirse en la principal competencia de IOS. A partir de este punto comienza a desarrollarse el universo de las apps, aparecen los juegos, diseños, noticias, actualidad, la posibilidad de tener todo y mucha información a la mano gracias al creciente desarrollo de las aplicaciones móviles.[4].

Actualmente en el área de logística de celebraciones especiales no se cuenta con una aplicación específica, para el desarrollo de dicha App se tuvieron en cuenta otras del área del sector comercial y para lograr su desarrollo se detallaron las que tienen características similares:

Mezly Brito, Ángel Pinzón de la Universidad Libre de Bogotá, Colombia Desarrollaron una App móvil que permite armar un atuendo (outfit), bien sea en vestuario, calzado y accesorios, buscando las prendas en un motor de búsqueda propio, donde están indexadas imágenes de las colecciones de todas las marcas presentes de acuerdo a tu posición geográfica. Es decir, la búsqueda se hace con imágenes y la aplicación traza un mapa para llegar al lugar donde se encuentra la prenda, aparecen dirección teléfono y horario de la atención de la tienda, estilos similares y los precios de dicho artículo[5].

Diana Lemus, Luis Poveda de la Universidad Piloto,Bogotá-Colombia Detectaron una oportunidad de mejora para eWorld, fortaleciendo línea de turismo digital, para captar un mayor mercado y mejorar el posicionamiento de la compañía. Diseñaron e implementaron una aplicación móvil dirigida a cualquier agencia de viajes, creada para facilitar el contacto con los clientes, donde estos tendrán acceso a todos los planes ofrecidos por la agencia, así como sus promociones, descuentos, etc. La apuesta fue a que en cuestión de minutos los clientes pudieran encontrar el plan adecuado a su necesidad y realizar el contacto con los asesores de manera rápida, concluyeron que con el constante desarrollo e innovación

[3]Mantilla, Ariza y Delgado, "Metodología para el desarrollo de aplicaciones móviles".

[4]Sequeira Plama, "Aplicacion movil bajo la plataforma Android para la suscripcion y notificaciones de la Asociacion de Emprendedores de la comunidad de renovacion familiar" HOSANNA", año 2014".

[5]Lubo, Beatriz y Pinzón Doncel, "Diseño de una aplicación móvil para la oferta de servicios de información (tendencias, precios y ubicación) enfocado a las prendas de vestir, accesorios y calzado en la ciudad de Bogotá DC".

de las tecnologías utilizadas en el mercado, es recomendable prepararse y adaptarse a este nuevo modelo de mercado, para lo cual la compañía eWorld deberá capacitar su personal profesional para atender las nuevas necesidades del mercado[6].

Oscar Higuera de la Universidad externado de Colombia desarrolló una aplicación móvil, dirigida a un nuevo enfoque de servicios ofrecidos por los tenderos en Colombia, permitiéndoles así ingresar al uso de la tecnología virtual y competir con grandes cadenas de servicio o franquicias reconocidas, que cuentan o poseen un servicio de domicilios. Permitiendo que los clientes accedan a los servicios de su barrio. Inicialmente, ofreciendo sus productos de forma efectiva desde plataformas web y móviles, ofreciendo un nuevo servicio de mercado al rango de direcciones donde se desea realmente prestar servicio de domicilios, permitiendo al tendero ser más eficiente en los tiempos de entrega, obtener una mejor calificación de sus clientes, control de sus pedidos y una mayor acogida de sus productos a sus vecinos más cercanos[7].

Santiago Arango, Danilo Escobar de la Universidad Pontificia Javeriana desarrollaron una aplicación móvil para suplir la necesidad de mejorar la forma en que los domicilios de las tiendas de barrio se realizan en la ciudad de Bogotá. La aplicación conecta al cliente directamente con la tienda, evitando tanto como sea posible la necesidad de un intermediario; en este caso el único intermediario es la aplicación. También es una gran ventaja para el dueño de tienda de barrio en la ciudad, en tanto que lo lleva un paso adelante en la modernización del sistema de domicilios, permitiendo a los pequeños negocios competir en el actual sistema económico[8].

3.2. SISTEMAS OPERATIVOS MÓVILES

Un sistema operativo móvil o (SO) móvil, es un sistema que controla un dispositivo móvil al igual que los PCs que utilizan Windows o Linux, los dispositivos móviles tienen sus sistemas operativos como Android, IOS entre otros. Los sistemas operativos móviles son mucho más simples y están más orientados a la conectividad inalámbrica. A medida que los teléfonos móviles crecen en popularidad, los sistemas operativos con los que funcionan adquieren mayor importancia[9].

3.2.1. Android

Android es una solución completa de software de código libre para teléfonos y dispositivos móviles. Es un paquete que engloba un sistema operativo, un "runtime" de ejecución basado en Java, un conjunto de librerías de bajo y medio nivel y un conjunto inicial de aplicaciones destinadas al usuario final (todas ellas desarrolladas en Java). Android

[6]Lemus Acosta y Poveda Poveda, "Diseño e implementación de una aplicación móvil para agencias de viajes".

[7]Higuera Benavides, "Veci-Domicilios".

[8]Arango Varón y Escobar Buitrago, "Mercapp: Aplicativo móvil para la recepción, solicitud y gestión de domicilios en las tiendas de barrio de Bogotá".

[9]DTyOC, *Sistemas operativos*.

se distribuye bajo una licencia libre permisiva (Apache) que permite la integración con soluciones de código propietario[10].

Android surge como resultado de la Open Handset Alliance[11] un grupo de 84 empresas de tecnología y dispositivos móviles distribuidas por todo el mundo con intereses diversos en la telefonía móvil y un compromiso de comercializar dispositivos móviles con este sistema operativo. El desarrollo viene de la avalado principalmente por Google (tras la compra de Android Inc. en 2005) y entre las compañías encontramos compañías de software (Ebay, LivingImage...), operadores (Telefónica, Vodafone, T-Mobile...), fabricantes de móviles (Motorola, Samsumg, acer, LG, HTC...) o fabricantes de Hardware (nVidia, Intel o Texas Instruments).

Android domina el tercer trimestre de 2018 con un 86,8 % de cuota de mercado, según IDC[12], de acuerdo al último informe de International Data Corporation (IDC), las ventas de dispositivos Android han superado ampliamente a las de la firma de la manzana que se conforma con un 13,2 % del pastel que, dicho sea de paso, es ligeramente superior a la cuota del Q3-2017, que se quedó en 12,4 %.

3.2.2. Arquitectura

Android presenta una arquitectura basada en 4 niveles figura 1, que detallamos acontinuación por orden ascendente[13]

- Un kernel linux versión 2.6 que sirve como base de la pila de software y se encarga de las funciones más básicas del sistema: gestión de drivers, seguridad, comunicaciones, etc
- Una capa de bibliotecas de bajo nivel en C y C++, como SQLite para persistencia de datos; OpenGL ES para gestión de gráficos 3D, con aceleración 3D opcional y Webkit como navegador web embebido y motor de renderizado HTML
- Un framework para el desarrollo de aplicaciones, dividido en subsistemas para gestión del sistema como el .Administrador de paquetes", el .Administrador de telefonía"(para la gestion del hardware del teléfono anfitrión) o el acceso a APIs sofisticadas de geolocalización o mensajería XMPP. Los desarrolladores tienen acceso completo a los mismos APIs del framework usados por las aplicaciones base. La arquitectura está diseñada para simplificar el reuso de componentes; cualquier aplicación puede publicar sus capacidades y cualquier otra aplicación puede luego hacer uso de esas capacidades (sujeto a reglas de seguridad del framework). Éste mismo mecanismo permite que los componentes sean reemplazados por el usuario.También incluye un

[10] Blanco y col., "Metodología de desarrollo ágil para sistemas móviles. Introducción al desarrollo con Android y el iPhone".

[11] Alliance, *¿Qué se necesita para construir un mejor teléfono móvil?*

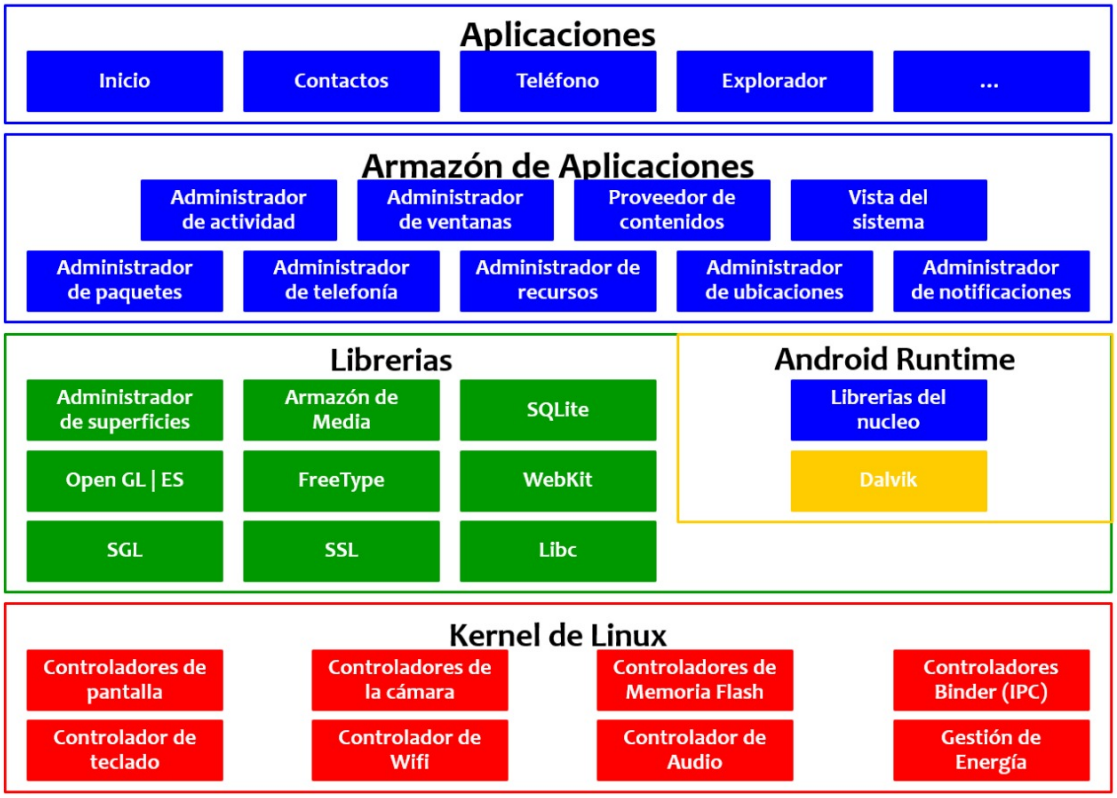
[12] Android, *Android domina el tercer trimestre de 2018 con un 86,8cuota de mercado, según IDC*.

[13] Blanco y col., "Metodología de desarrollo ágil para sistemas móviles. Introducción al desarrollo con Android y el iPhone".

sistema de vistas para manejar el interfaz de usuario de las aplicaciones, que incluyendo posibilidad de visualización de mapas o renderizado html directamente en el interfaz gráfico de la aplicación

- Aplicaciones: Las aplicaciones base incluyen un teléfono, cliente de email, programa de envío de SMS, calendario, mapas, navegador, contactos... que pueden a su vez ser usados por otras aplicaciones.

Figura 1: Arquitectura de Android



Fuente: `https://tecnologiamovil128806266.files.wordpress.com/2018/11/02_pilasoftware-1024x578.png?w=660v`

3.2.3. Entorno de desarrollo

Android Studio es el entorno de desarrollo integrado (IDE) oficial para el desarrollo de aplicaciones para Android y se basa en IntelliJ IDEA. Además del potente editor de códigos y las herramientas para desarrolladores de IntelliJ, Android Studio ofrece aún más funciones que aumentan la productividad durante la compilación de apps para Android, como las siguientes[14]:

- Un sistema de compilación flexible

[14]Developers, *Conoce Android Studio*.

- Un emulador rápido con varias funciones
- Un entorno unificado en el que puedes realizar desarrollos para todos los dispositivos Android
- Ejecución instantánea para aplicar cambios mientras tu app se ejecuta sin la necesidad de compilar un nuevo APK
- Integración de plantillas de código y GitHub para ayudarte a compilar funciones comunes de las apps e importar ejemplos de código
- Gran cantidad de herramientas y frameworks de prueba
- Herramientas Lint para detectar problemas de rendimiento, usabilidad, compatibilidad de versión, etc.
- Compatibilidad con C++ y NDK
- Soporte incorporado para la plataforma en la nube de Google, lo que facilita la integración de mensajería en la nube de Google y motor de App.

3.3. DISPOSITIVOS MÓVILES

Un dispositivo móvil es un microordendaor de procesamiento y memoria inferior a las de un ordenador que generalmente incluye una pantalla y un método de entrada (ya sea táctil o teclado en miniatura). Tienen sistemas operativos que pueden ejecutar aplicaciones y cuentan con las siguientes características:

- Capacidades especiales de procesamiento
- Conexion permanente o intermitente a una red
- Memoria limitada
- Diseños específicos

3.3.1. Smartphone

Es un teléfono móvil inteligente con sistema operativo, amplio procesamiento, aplicaciones y conectividad a Internet, sus características son:

- Funciones básicas como agenda, mensajes, llamadas, calendario, etc
- Gestión de cuentas de correo y posibilidad de conexión a redes sociales.
- Permiten la instalación de programas de terceros.
- Conexión a Internet, y sincronización con diferentes ordenadores.
- Lectura, edición y reproducción de editores de texto, hojas de cálculo, fotos, videos, etc.

3.4. APLICACIONES MÓVILES

Las aplicaciones móviles también llamadas Apps son programas que se pueden descargar y se pueden acceder desde un dispositivo móvil.

En los últimos años el uso de Smartphones, iPhone y tablets ha penetrado tan profundamente la sociedad, que actualmente se constituye no solo como un elemento de comunicación, sino que diariamente cobra importancia como herramienta de información y productividad en diferentes ámbitos como son lo social, laboral, académico y otros. El desarrollo de apps móviles para estos dispositivos es una pieza fundamental en el mundo moderno. Sin embargo el camino que debe emprender el desarrollador está lleno de conocimiento que se actualiza día a día. Actualmente son muchos los lenguajes de programación que se utilizan en diferentes partes del proceso. Aunque todo en internet se puede resumir en 2 grandes campos. El frontend y el backend, el primero hace referencia a lo que tiene el usuario, el diseño de páginas, aplicaciones, usabilidad, y todo lo que interactúa con el cliente. El segundo tiene que ver con todo lo que está detrás de internet, la red de servidores y los sistemas de almacenamiento de datos. Estas dos capas se unen a través de un puente llamado "capa de transporte".

3.5. FRONTEND

Frontend es la parte de un programa o dispositivo a la que un usuario puede acceder directamente. Son todas las tecnologías de diseño y desarrollo que encargan de la interactividad con los usuarios. Para un frontend la creatividad es el recurso más valioso, ya que tendrá que tomar fuentes, colores, imágenes y todos lo recursos de los cuales disponga para crear sitios agradables que se vean bien en todos los dispositivos y resoluciones.

Redux es una librería que implementa el patrón de diseño Flux, con algunas variaciones. Redux es una librería JavaScript muy pequeña (Apenas 2KB en total), con muy poco código. Su API apenas son 5 funciones y lo más importante es que es JavaScript puro, por lo que es agnóstica al framework y puede utilizarse con cualquier librería.

3.5.1. React Native

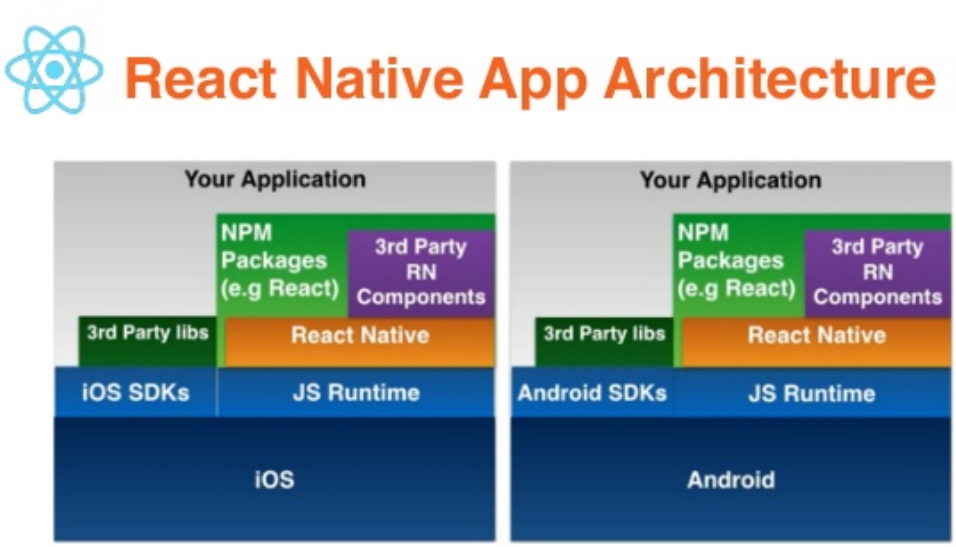
Aunque no es un lenguaje de programación sino un framework de JavaScript está tomando gran popularidad en los desarrolladores frontend. Creada por el equipo de Facebook, React Native permite crear aplicaciones móviles usando solo JavaScript. Utiliza el mismo diseño que React, lo que le permite componer una interfaz de usuario móvil rica utilizando componentes declarativos, las aplicaciones que se están creando con RN no son aplicaciones web para dispositivos móviles ya que React Native utiliza los mismos bloques de construcción fundamentales de UI que las aplicaciones normales de iOS y Android. En lugar de usar Swift, Kotlin o Java, puede poner esos bloques de construcción juntos usando JavaScript y React. Además se combina sin problemas con componentes escritos en Swift, Java o Objective-C. Es fácil desplegar el código nativo si necesita optimizar algunos aspectos de su aplicación. También es fácil crear una parte de la aplicación en React Native, y otra

parte de la aplicación utilizando directamente el código nativo. Esto lo convierte en una excelente opción multiplataforma ya que permite funcionar en Android y IOS sin hacer muchos cambios. Actualmente empresas con millones de usuarios como FACEBOOK, INSTAGRAM, UBER, PINTEREST, WIX entre muchas otras utilizan React Native en sus aplicaciones móviles[15].

El API de React Native ofrece acceso a un gran número de funcionalidades nativas, y en Facebook están trabajando muy duro para ampliar cada vez más este abanico de posibilidades, pero puede darse el caso de que sea necesario añadir código nativo al desarrollo que todavía no esté accesible (o incluir alguna librería de terceros por ejemplo). Puede comunicar el código javascript con código nativo mediante 'envoltorios', teniendo acceso total al hardware y a los APIs nativos. La principal ventaja que tiene React Native frente a sus competidores Web App es la experiencia de usuario nativa. Esta experiencia nativa se consigue principalmente utilizando componentes visuales nativos y mediante las animaciones[16]. La Figura 2, muestra la arquitectura de una App en reac native.

React Native es como React, pero usa componentes nativos en lugar de componentes web como bloques de construcción. Entonces, para comprender la estructura básica de una aplicación React Native, se tiene que comprender algunos de los conceptos básicos de React, como JSX, componentes de estado y propiedades.

Figura 2: Arquitectura de una App en react native



Fuente: `https://www.slideshare.net/sambhu7/introduction-to-react-native-71847255/`

Componentes y APIs

[15]source, *React Native Build native mobile apps using JavaScript and React*.
[16]Almeria, *¿Qué es React Native?*

React Native proporciona una serie de componentes incorporados[17]:

- Componentes básicos

  La mayoría de las aplicaciones terminarán usando uno de estos componentes básicos:

  - Vista: El componente más fundamental para la construcción de una interfaz de usuario
  - Texto: Un componente para mostrar texto.
  - Imagen: Un componente para mostrar imágenes.
  - Entrada de texto: Un componente para ingresar texto en la aplicación a través de un teclado.
  - Vista de desplazamiento: Proporciona un contenedor de desplazamiento que puede alojar múltiples componentes y vistas.
  - Hoja de estilo: Proporciona una capa de abstracción similar a las hojas de estilo CSS.

- Interfaz de usuario

  Procesa controles de interfaz de usuario comunes en cualquier plataforma utilizando los siguientes componentes:

  - Botón: Un componente de botón básico para el manejo de toques que se debe representar bien en cualquier plataforma.
  - Recogedor: Representa el componente del selector nativo en iOS y Android.
  - Deslizador: Un componente utilizado para seleccionar un solo valor de un rango de valores.
  - Cambio: Representa una entrada booleana.

- Vista de listas

  Representan elementos que se muestran actualmente en la pantalla. Esto los convierte en una excelente opción para mostrar largas listas de datos:

  - Lista plana: Un componente para la representación de listas desplazables
  - Lista de secciones: Como una lista plana, pero para las listas de secciones.

- Específico de Android

  Muchos de los siguientes componentes proporcionan envoltorios para las clases de Android más utilizadas:

  - Controlador de regreso: Detecta pulsaciones de botón de hardware para navegación trasera.

[17]Native, *Componentes y APIs*.

- Selector de fechas Android: Abre el diálogo estándar de selección de fecha de Android.
- Diseño de cajones Android: Representa un diseño de cajones Android.
- Permisos Android: Proporciona acceso al modelo de permisos introducido en Android M.
- Barra de progreso Android: Representa una barra de progreso Android.
- Selector de tiempo Android: Abre el diálogo estándar del selector de tiempo de Android.
- Notificaciones Android: Crea una alerta de Android.
- Barra de herramientas Android: Representa una Barra de herramientas en Android.
- Paginador de vistas de Android: Contenedor que permite voltear a izquierda y derecha entre vistas de niños.

Propiedades

La mayoría de los componentes se pueden personalizar cuando se crean, con diferentes parámetros. Estos parámetros de creación son llamados propiedades.

Por ejemplo, un componente React Native básico es la imagen. Cuando se crea una imagen, se puede usar una propiedad llamada fuente para controlar qué imagen muestra.

Sus propios componentes también pueden usar propiedades. Esto le permite crear un único componente que se usa en muchos lugares diferentes en la aplicación.[18].

Figura 3: Ejemplo de Propiedades

Fuente: `https://facebook.github.io/react-native/docs/props`

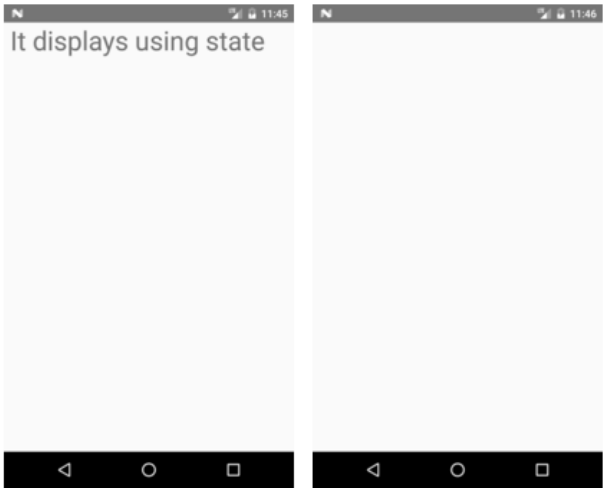
[18]Native, *Props*.

Estado

Hay dos tipos de datos que controlan un componente: propiedades y estado. Propiedades las establece el padre y se fijan durante toda la vida útil de un componente. Para los datos que van a cambiar, se tiene que usar los estados. En general, se debe inicializar estado en el constructor y luego llamar ajustar estado cuando se desee cambiarlo.

Por ejemplo, si se desea que un texto que parpadee todo el tiempo. El texto en sí se establece una vez cuando se crea el componente que parpadea, por lo que el texto en sí es una propiedad. El "si el texto está actualmente activado o desactivado" cambia con el tiempo, por lo que debe mantenerse el estado[19].

Figura 4: Ejemplo de Estado



Fuente:
`https://habiletechnologies.com/blog/understanding-the-basic-components-of-react-native/`

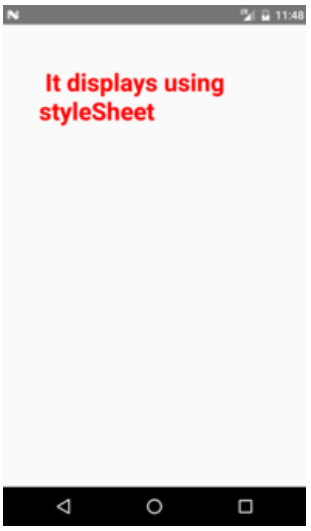
Estilo

Con React Native, no se usa un lenguaje especial o sintaxis para definir estilos. Simplemente lo usa de la aplicación utilizando JavaScript. Todos los componentes principales aceptan una propiedad llamada estilo. Los nombres y valores generalmente coinciden con el funcionamiento de CSS en la web.

La propiedad estilo puede ser un objeto JavaScript simple y antiguo lo que usualmente se utiliza. También puede pasar una matriz de estilos: el último de la matriz tiene prioridad, por lo que puede usar esto para heredar estilos. Un patrón común es hacer que el componente acepte una propiedad que, a su vez, se utiliza para diseñar subcomponentes[20]. Se pueden usar éstos "en cascada" como en CSS.

[19] Native, *State*.
[20] Native, *Style*.

Figura 5: Ejemplo de Estilo



Fuente: `https://habiletechnologies.com/wp-content/uploads/2017/02/screenshot_1487830700-196x337.png`

Altura y Ancho

La altura y el ancho de un componente determinan su tamaño en la pantalla[21] y se dividen en dos:

- Dimensiones fijas: La forma más sencilla de establecer las dimensiones de un componente es agregar un estilo fijo ancho y fijo alto. Todas las dimensiones en React Native son unitarias y representan píxeles independientes de la densidad. Establecer las dimensiones de esta manera es común para los componentes que siempre deben mostrarse exactamente al mismo tamaño, independientemente de las dimensiones de la pantalla.
- Dimensiones flexibles: Se usa flex en el estilo de un componente para que el componente se expanda y se reduzca dinámicamente según el espacio disponible. Normalmente se usará flex: 1, que dice a un componente que llene todo el espacio disponible, compartido de manera uniforme entre otros componentes con el mismo padre. Cuanto mayor sea el valor flex dado, mayor será la proporción de espacio que tomará un componente en comparación con sus hermanos.

[21] Native, *Height and Width*.

Figura 6: Ejemplo de Altura y Ancho

Fuente: https://1.bp.blogspot.com/-dCRrrX5obV8/XJY51YGKyaI/AAAAAAAAClo/x0rGYL9CQsQJv373LcyjFG7qNfLiE74TwCLcBGAs/s400/Screenshot_1553349072.png

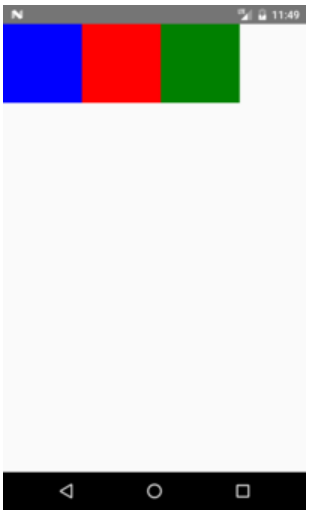
Diseño con flexbox

Un componente puede especificar el diseño de sus hijos utilizando el algoritmo flexbox. Flexbox está diseñado para proporcionar un diseño consistente en diferentes tamaños de pantalla. Es normal que se utilice la siguiente combinación para lograr la disposición correcta[22]:

- Dirección flexible: Agregarlo un componente estilo determina el eje primario de su diseño.
- Justificar contenido: Agregarlo al estilo de un componente determina la distribución de los niños a lo largo del eje primario
- Alinear elementos: Agregarlo al estilo de un componente determina la alineación de los hijos a lo largo del eje secundario (si el eje primario es fila, entonces el secundario es columna, y viceversa)

[22]Native, *Layout with Flexbox*.

Figura 7: Ejemplo de Diseño con Flexbox



Fuente: `https://habiletechnologies.com/wp-content/uploads/2017/02/screenshot_1487830770.png`

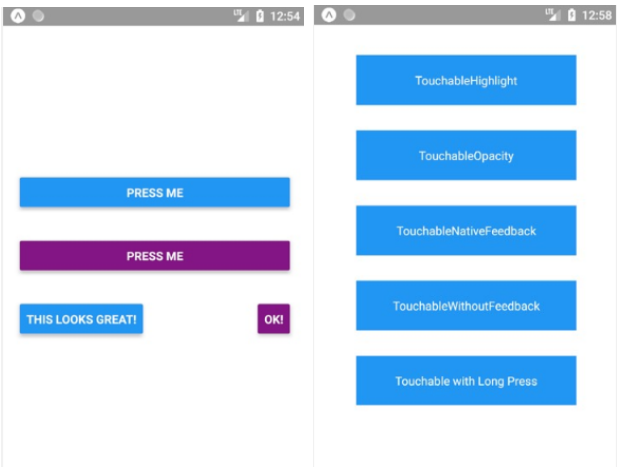
Toques de manejo

Los usuarios interactúan con aplicaciones móviles principalmente a través del tacto. Pueden usar una combinación de gestos, como tocar un botón, desplazarse por una lista o hacer zoom en un mapa. React Native proporciona componentes para manejar todo tipo de gestos comunes, así como un completo sistema de respuesta de gestos para permitir un reconocimiento de gestos más avanzado, pero el componente básico que probablemente le interesará es el botón básico[23]:

- Botón básico: Proporciona un componente de botón básico que se representa muy bien en todas las plataformas
- Tocables: Si el botón básico no se ve bien para la aplicación, se puede crear un botón propio usando cualquiera de los componentes "táctiles" provistos por React Native. Los componentes "táctiles" brindan la capacidad de capturar gestos tocando y pueden mostrar comentarios cuando se reconoce un gesto.

[23]Native, *Handling Touches*.

Figura 8: Ejemplo de Toques de manejo



(a) Botón básico (b) Tocables

Fuente: `https://facebook.github.io/react-native/docs/handling-touches`

Vista de desplazamiento

La Vista de desplazamiento es un contenedor genérico que puede alojar múltiples componentes y puntos de vista. Los elementos desplazables no necesitan ser homogéneos, y puede desplazarse tanto vertical como horizontalmente (configurando la propiedad horizontal). También se puede configurar para permitir la paginación a través de vistas mediante el uso de gestos de deslizar utilizando los accesorios de paginación habilitada. El deslizamiento horizontal entre vistas también se puede implementar utilizando el componente ver localizador de Android.

Funciona mejor para presentar una pequeña cantidad de cosas de un tamaño limitado. Todos los elementos y vistas se representan, incluso si no se muestran actualmente en la pantalla. Si tiene una lista larga de más elementos de los que caben en la pantalla, debe usar una lista plana en su lugar[24].

[24]Native, *Using a ScrollView*.

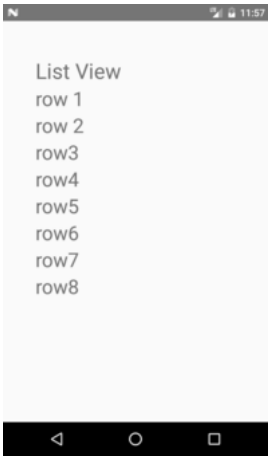
Figura 9: Ejemplo de Vista de desplazamiento

Fuente: `https://habiletechnologies.com/wp-content/uploads/2017/02/screenshot_1487831144-199x337.png`

Vistas de lista

La Lista plana o desplazamiento de datos cambiantes, pero estructurados de manera similar. Funciona bien para largas listas de datos, donde la cantidad de elementos puede cambiar con el tiempo. A diferencia de los más genéricos vista de desplazamiento, la lista plana muestra los únicos elementos que actualmente están en la pantalla, no todos los elementos a la vez[25].

Figura 10: Ejemplo de Vistas de lista



Fuente: `https://habiletechnologies.com/wp-content/uploads/2017/02/screenshot_1487831246-198x337.png`

[25]Native, *Using List Views*.

Navegación con React

Nace de la necesidad de la comunidad nativa React de contar con una solución de navegación extensible pero fácil de usar escrita en su totalidad en JavaScript (para que pueda leer y comprender toda la fuente), además de poderosas primitivas nativas[26]:

- Navegación de pestañas: Posiblemente el estilo más común de navegación en aplicaciones móviles es la navegación basada en pestañas. Esto puede ser pestañas en la parte inferior de la pantalla o en la parte superior debajo del encabezado (o incluso en lugar de un encabezado).

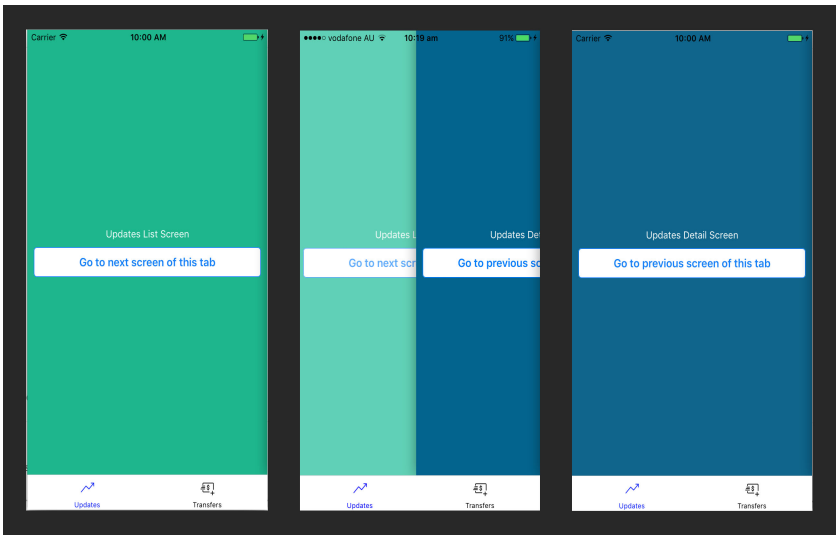
Figura 11: Navegador de pestañas

Fuente: `https://static.javatpoint.com/tutorial/react-native/images/react-native-create-material-button-tab-navigator-output1.png`

- Navegador de pila: Proporciona una forma para que la aplicación realice la transición entre pantallas y administre el historial de navegación. Si la aplicación usa solo un navegador de pila, entonces es conceptualmente similar a cómo un navegador web maneja el estado de navegación: la aplicación empuja y saca elementos de la pila de navegación a medida que los usuarios interactúan con ella, y esto hace que el usuario vea diferentes pantallas. Además, proporciona los gestos y animaciones que esperaría en Android y iOS al navegar entre rutas en la pila.

[26] Navigation, *Fundamentos*.

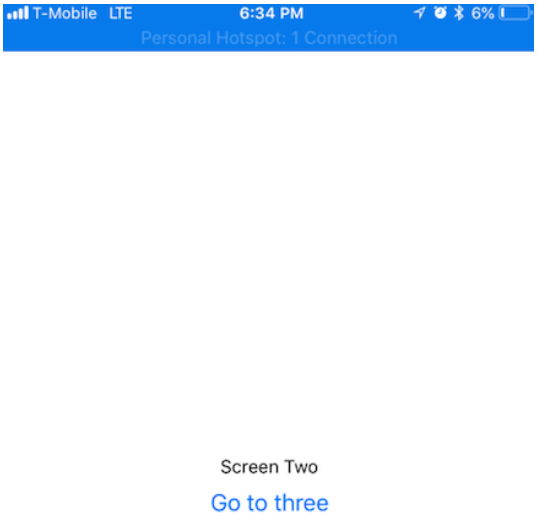
Figura 12: Navegador de pila



Fuente: `https://i.stack.imgur.com/25MyO.jpg`

- Navegador de cambio: Muestra solo una pantalla a la vez. De forma predeterminada, no controla las acciones de retroceso y restablece las rutas a su estado predeterminado cuando se aleja.

Figura 13: Navegador de cambio



Fuente: `https://miro.medium.com/max/433/1*a60Cj3sdhDMRPN4i0MTBVg.png`

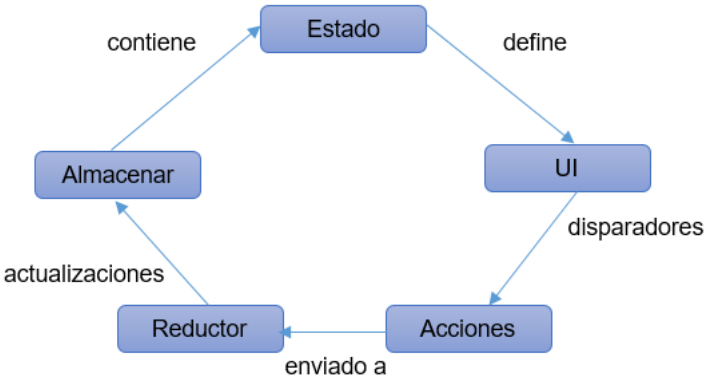
3.5.2. Redux

Redux es un contenedor predecible del estado de aplicaciones JavaScript. Ayuda a escribir aplicaciones que se comportan de manera consistente, corren en distintos ambientes (cliente, servidor y nativo), y son fáciles de probar. Además de eso, provee una gran experiencia de desarrollo, gracias a edición en vivo combinado con un depurador sobre una línea de tiempo. Se Puede usar Redux combinado con React, o cual cualquier otra librería de vistas. Funciona especialmente bien con librerías como React y Deku porque permiten describir la interfaz de usuario como una función de estado, y Redux emite actualizaciones de estado en respuesta a acciones[27].

React Redux

Permite construir componentes que reaccionan a los cambios del estado de la aplicación. Los componentes afectados por un cambio de estado se vuelven a representar con los nuevos datos. Los componentes también envían acciones, por ejemplo, cuando se hace clic en un botón, en la figura 15 se observa el flujo de datos.

Figura 14: Flujo de datos de React Redux



Fuente: `https://cdn-images-1.medium.com/max/1200/0*cntBtPADjE2ykLSP.png`

- Acciones: Son un bloque de información que envía datos desde la aplicación al almacenamiento y son la única fuente de información de éste. Las envía al almacenamiento usando store.dispatch(). Son objetos planos de JavaScript. Una acción debe tener una propiedad tipo, que indica el tipo de acción a realizar. Los tipos normalmente son definidos como strings constantes.

[27] Redux, *Básico*.

- Reductores: Las acciones describen que algo pasó, pero no especifican cómo cambió el estado de la aplicación en respuesta. Esto es trabajo de los Reductores. Éstos cumplen las siguientes funciones:
  - Modificar el estado
  - Puede haber mas de un reductor
  - Es un función pura
  - Devuelve el siguiente estado
- Almacenar: Es el objeto que los reúne. El Almacenamiento tiene las siguientes responsabilidades:
  - Contiene el estado de la aplicación;
  - Permite el acceso al estado via getState();
  - Permite que el estado sea actualizado via dispatch(action);
  - Registra los oyentes via subscribe(listener);
  - Maneja la anuliación del registro de los oyentes via el retorno de la función de subscribe(listener).

Componentes y Contenedores

- Componentes
  - Propósito: Margen, estilos
  - Para leer datos: Lee datos de las propiedades
  - Para manipular datos: Invoca llamada de retorno (callback) desde las propedades
- Contenedores
  - Propósito: Búsqueda de datos, actualizaciones de estado
  - Para leer datos: Se suscribe al estado en Redux
  - Para manipular datos: Envía acciones a Redux

Diseño de la Jerarquía de Componentes

- Components Desgin Se pueden ver los siguientes componentes de presentación y sus props surgir a través de esta breve descripción:
  - TodoList: es una lista que mostrará las tareas pendientes disponibles.
  - Todo: es un asunto pendiente.
  - Link: es el enlace con su callback. props
  - Footer: es donde el usuario cambia las tareas pendientes visibles actualmente.

  - App es el componente raíz que representa todo lo demás.

- Containers Desgin También se necesitan algunos componentes contenedores para conectar los componentes de presentación a Redux. Por ejemplo, el componente de presentación TodoList necesita un contenedor como VisibleTodoList que se suscribe al store de Redux y debe saber cómo aplicar el filtro de visibilidad. Para cambiar el filtro de visibilidad, proporcionaremos un componente contenedor FilterLink que renderiza un Link que distribuye la debida acción al hacer clic:
  - VisibleTodoList: filtra los asuntos de acuerdo a la visibilidad actual y renderiza el TodoList
  - FilterLink: obtiene el filtro de visibilidad actual y renderiza un Link

3.6. BACKEND

Backend es la capa de acceso a datos de un software o cualquier dispositivo, que no es directamente accesible por los usuarios, además contiene la lógica de la aplicación que maneja dichos datos. El Backend también accede al servidor, que es una aplicación especializada que entiende la forma como el navegador solicita cosas. Algunas de las funciones que se gestionan en la parte del backend son:

- El Desarrollo de funciones que simplifiquen el proceso de desarrollo.
- Menor almacenamiento del dispositivo
- Sincronización de datos entre múltiples dispositivos
- Manejo de datos fuera de línea
- Acciones de lógica.
- Conexión con bases de datos.
- Uso de librerías del servidor web (por ejemplo para implementar temas de caché o para comprimir las imágenes de la web)
- Minimizar gasto de la batería

3.6.1. Firebase

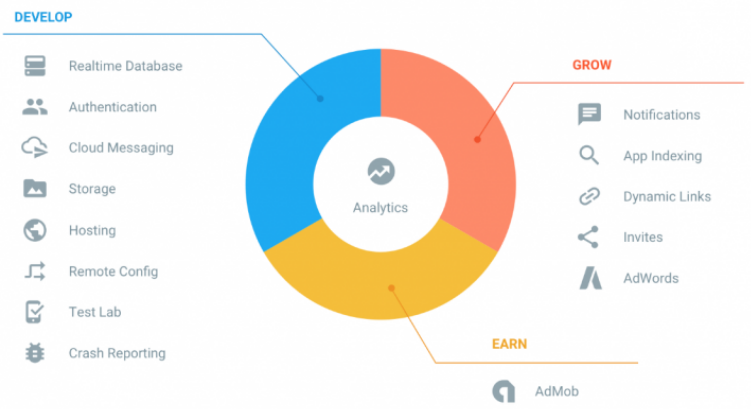
Es una base de datos noSQL y es la apuesta de google como la nueva y mejorada plataforma de desarrollo móvil en la nube, que ayuda a desarrollar rápidamente aplicaciones de alta calidad. Se trata de una plataforma disponible para diferentes plataformas (Android, iOS, web)[28]. Algunas de sus características son:

- Compila Apps rápido, sin administrar la infraestructura

[28] Firebase, *Firebase*.

- Con el respaldo de Google y la confianza de Apps reconocidas
- Una plataforma, con productos que funcionan mejor juntos.s

Figura 15: Entorno de Firebase



Fuente: `https://elandroidelibre.elespanol.com/wp-content/uploads/2016/05/Screen-Shot-2016-05-19-at-00.13.32-750x463.png`

3.6.2. Servicios de Firebase

Firebase ofece diferentes tipos de servicios para trabajar de una manera más sencilla las aplicaciones.

- Autenticación de Firebase: Proporciona servicios de backend, SDK fáciles de usar y bibliotecas de IU ya elaboradas para autenticar a los usuarios. Admite la autenticación mediante contraseñas, números de teléfono, proveedores de identidad populares, como Google, Facebook y Twitter, y mucho más[29].
- Base de datos en tiempo real de Firebase: Almacena y sincroniza datos con nuestra base de datos NoSQL alojada en la nube. Los datos se sincronizan con todos los clientes en tiempo real y se mantienen disponibles cuando la app no tiene conexión. Firebase Realtime Database es una base de datos alojada en la nube. Los datos se almacenan en formato JSON y se sincronizan en tiempo real con cada cliente conectado. Cuando compilas apps multiplataforma con nuestros SDK de iOS, Android y JavaScript, todos los clientes comparten una instancia de la base de datos en tiempo real y reciben actualizaciones automáticamente con los datos más recientes[30].
- Almacenamiento Firebase: Es un servicio de almacenamiento de objetos potente, simple y rentable construido para la escala de Google. Los SDK de Firebase para

[29]Firebase, *Firebase Authentication*.
[30]Firebase, *Firebase Realtime Database*.

el almacenamiento en la nube agregan la seguridad de Google a las operaciones de carga y descarga de archivos para las Apps de Firebase, sin importar la calidad de la red[31].

[31] Firebase, *Cloud Storage*.

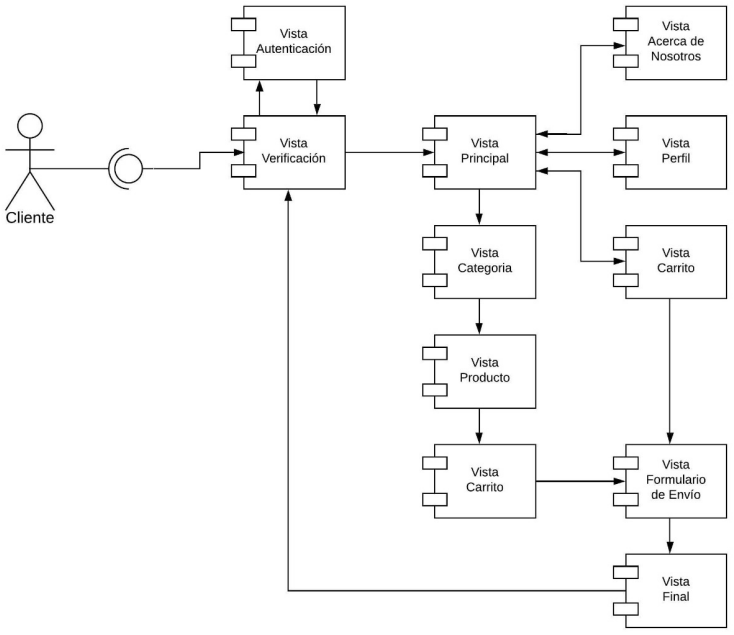
4. ANALISIS Y DISEÑO DEL PROYECTO

En este apartado se pretende analizar los requerimientos de la aplicación de forma completa. También se planteará el diseño de la arquitectura del sistema de tal manera que cumpla estos requisitos.

4.1. CAPTURA DE REQUERIMIENTOS

En la figura 16, se observa la captura de requerimientos que debe suplir la aplicación interactuando con el usaurio.

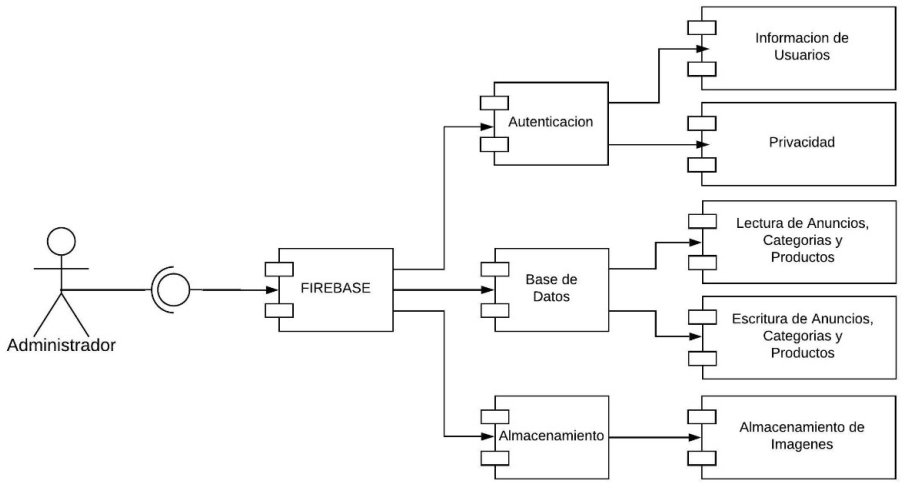
Figura 16: Interacción del usuario



Fuente: `Autores`

En la figura 17, se observa la captura de requerimientos para una correcta interacción con el administrador.

Figura 17: Interacción del administrador



Fuente: `Autores`

4.2. ANALISIS DE REQUERIMIENTOS

Luego de realizar un analisis y teniendo en cuenta la relacion con el mercado ademas del objetivo principal de la aplicación se determinó que el sistema debe cubrir los siguientes requerimientos:

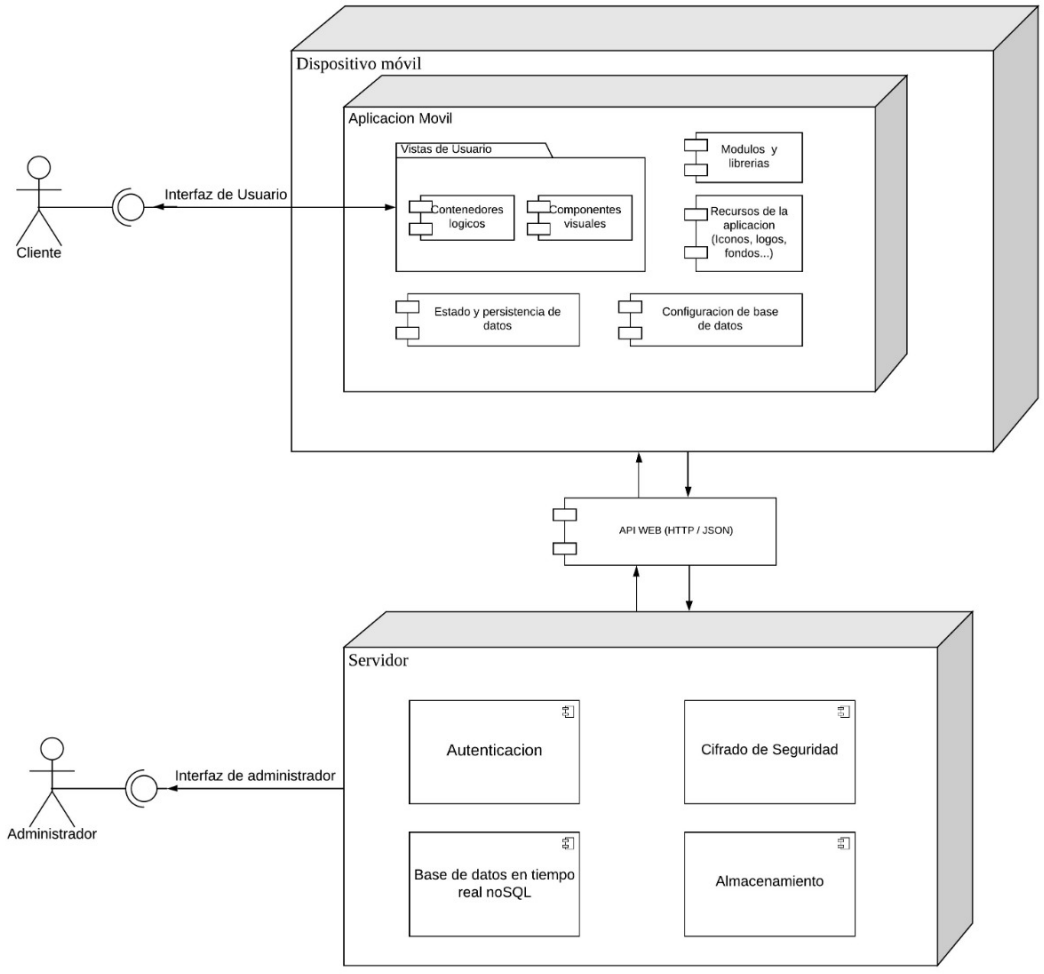
- Debe ofrecer un catalogo de productos y/o servicios organizados por categorías de tal manera que se pueda visualizar a través de un aplicativo móvil en un dispositivo Android alimentado por un servidor alojado en la nube.
- Las listas de productos y categorias deben almacenarse en una base de datos confiable, segura, que garantice la disponibilidad del sistema y que este presta a modificaciones por parte del administrador.
- Debe poseer un carrusel en la vista principal de la app donde se permita visualizar promociones, anuncios, eventos y/o cualquier clase de información a la que se desee dar prioridad, deseablemente que tenga habilitada la autoreproduccion.
- Debe permitir realizar cambios en el catalogo como imagenes, productos y precios sin que el usuario necesite actualizar la aplicación.
- Debe acompañar al cliente en el proceso de seleccion e ir almacenando las decisiones de compra en un carrito que posteriormente se deberá enviar a la base de datos junto con los datos de envío proporcionados por el cliente al momento de confirmar la compra.

- Debe contar con un sistema de autenticacion de usuarios robusto y escalable de tal manera que el cliente pueda navegar en la aplicacion omitiendo este paso pero que sea requerido al momento de concretar un pedido.

4.3. DISEÑO DE LA ARQUITECTURA DEL SISTEMA

Luego, en base a los requerimientos se diseñó el diagrama de despliegue de la figura 18, es la primera descripcion de la arquitectura del sistema donde se relacionan los componentes hardware con los componentes de software y la forma en la que interactuan con los usuarios.

Figura 18: Arquitectura UML de despliegue

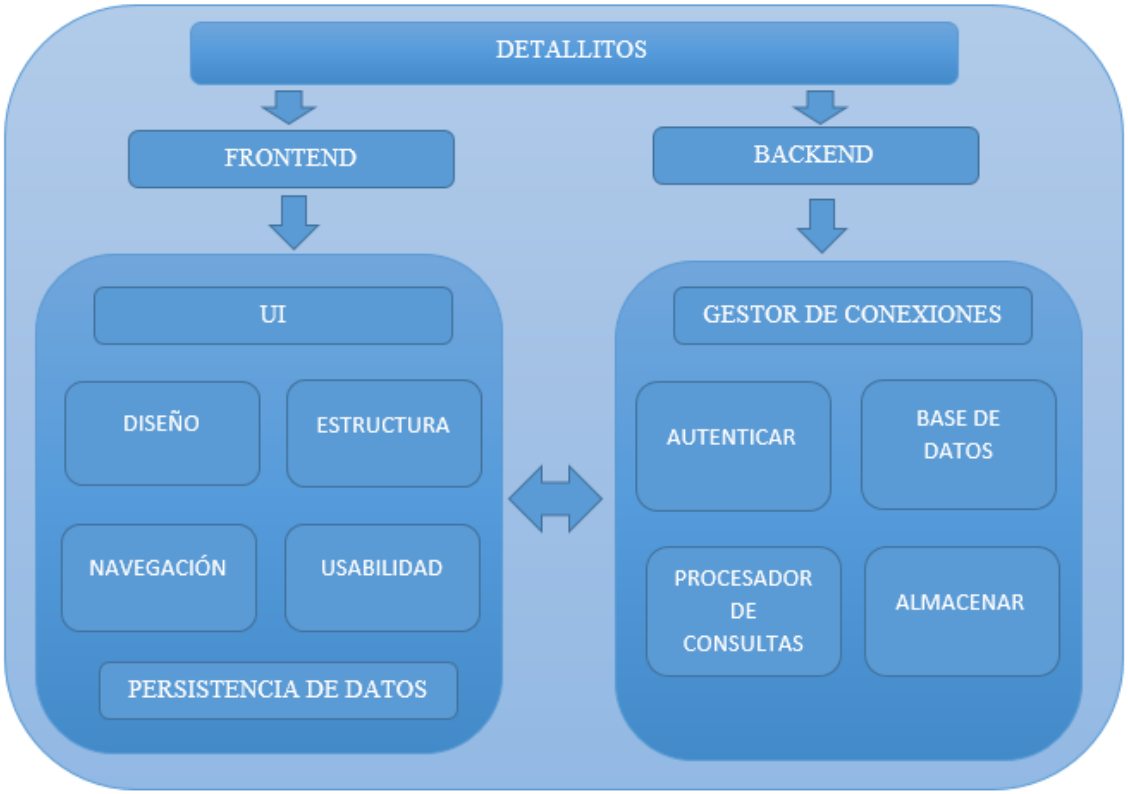


De esta forma podemos seccionar el diseño y desarrollo del sistema en 2 pilares fundamentales, figura 19

- Una aplicación frontend diseñada para Android con el framework React Native

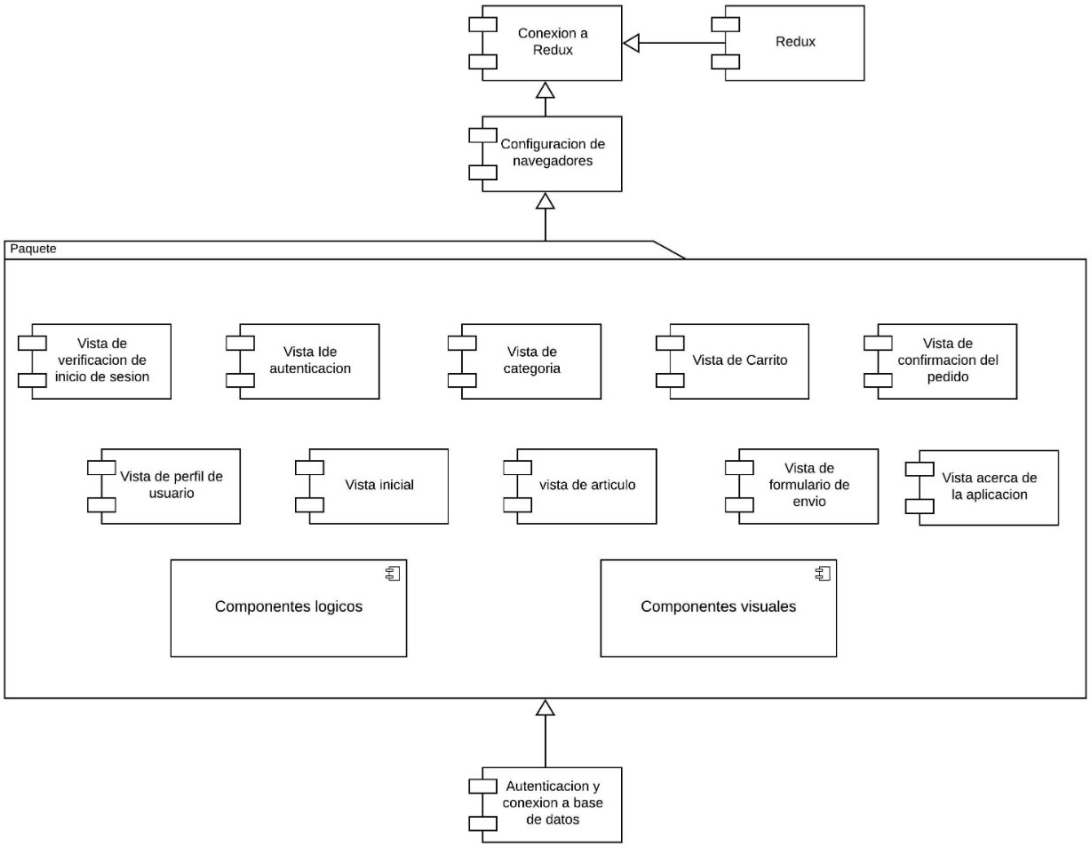
- Un backend en tiempo real realizado con Firebase

Figura 19: Pilares del sistema Detallitos



En la figura 20 se puede observar el diagrama de componentes donde se relacionan los subsistemas de software que se deben sincronizar en la aplicación.

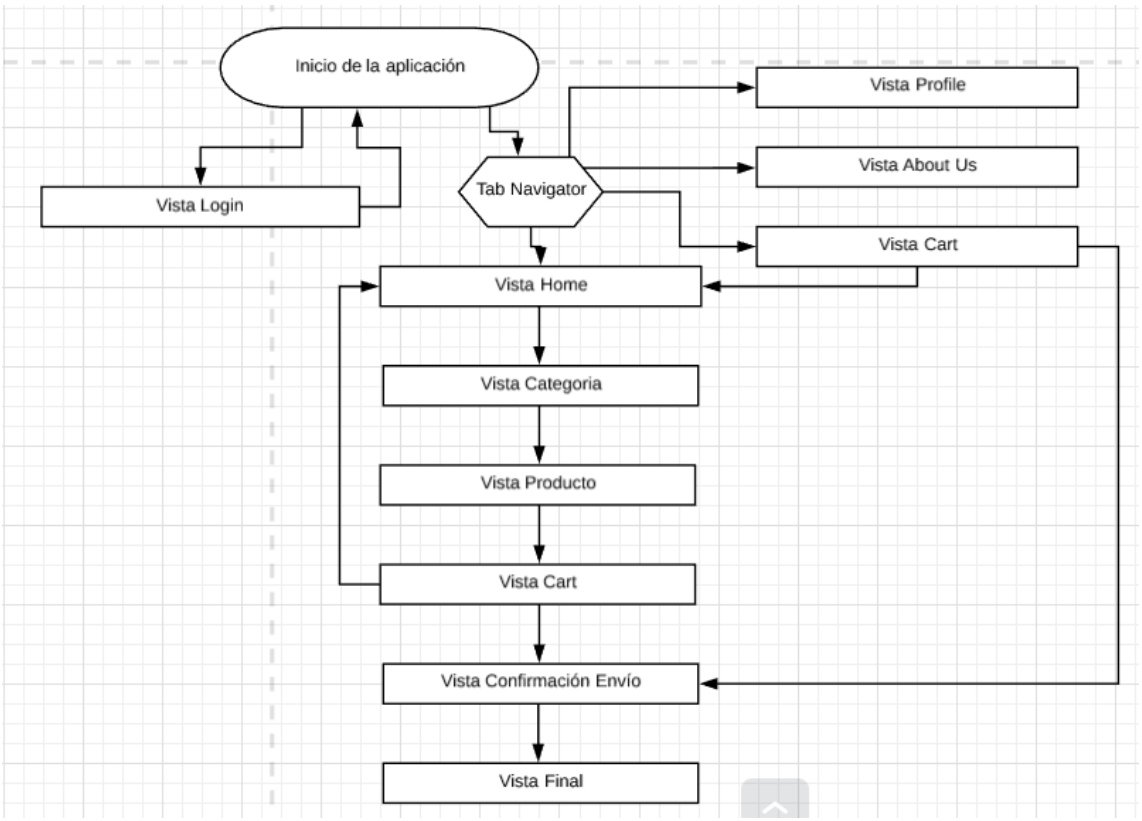
Figura 20: Diagrama de componentes de la aplicacion movil



4.4. NAVEGACIÓN

En la figura 21 se muestra el esquema de navegacion entre pantallas de la aplicación.

Figura 21: Esquema de navegación

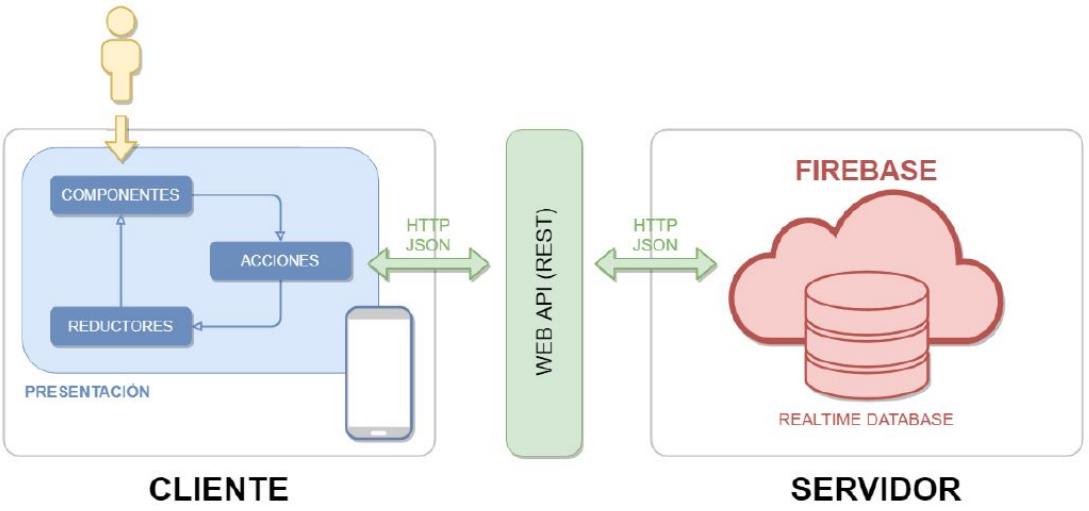


4.5. MODELO DE CONEXIÓN CLIENTE SERVIDOR

En la figura 22 se observa el modelo cliente-servidor implementado en la aplicación Detallitos con Firebase, aquí se puede apreciar como a través de Internet el cliente es decir un smartphone con sistema operativo Android realiza una conexiona bidireccional con nuestro servidor Firebase permitiendo así realizar peticiones de lectura y escritura usando el protocolo http.

Este modelo nos permite también tener una representación actualizada de lo que sucede en el backend para el lado del cliente y viceversa, es decir, que cada vez que haya un cambio en la información de la base de datos ya sea por un nuevo producto, categoría, promociones y/o descuentos el cliente recibirá esta actualización sincronizando y reflejando esta nueva información en el dispositivo, por otra parte, cuando el cliente envíe una petición de escritura con un nuevo pedido, este objeto sera visualizado inmediatamente en la base de datos.

Figura 22: Modelo de conexión cliente servidor



5. DESARROLLO E IMPLEMENTACIÓN DE LA APLICACIÓN

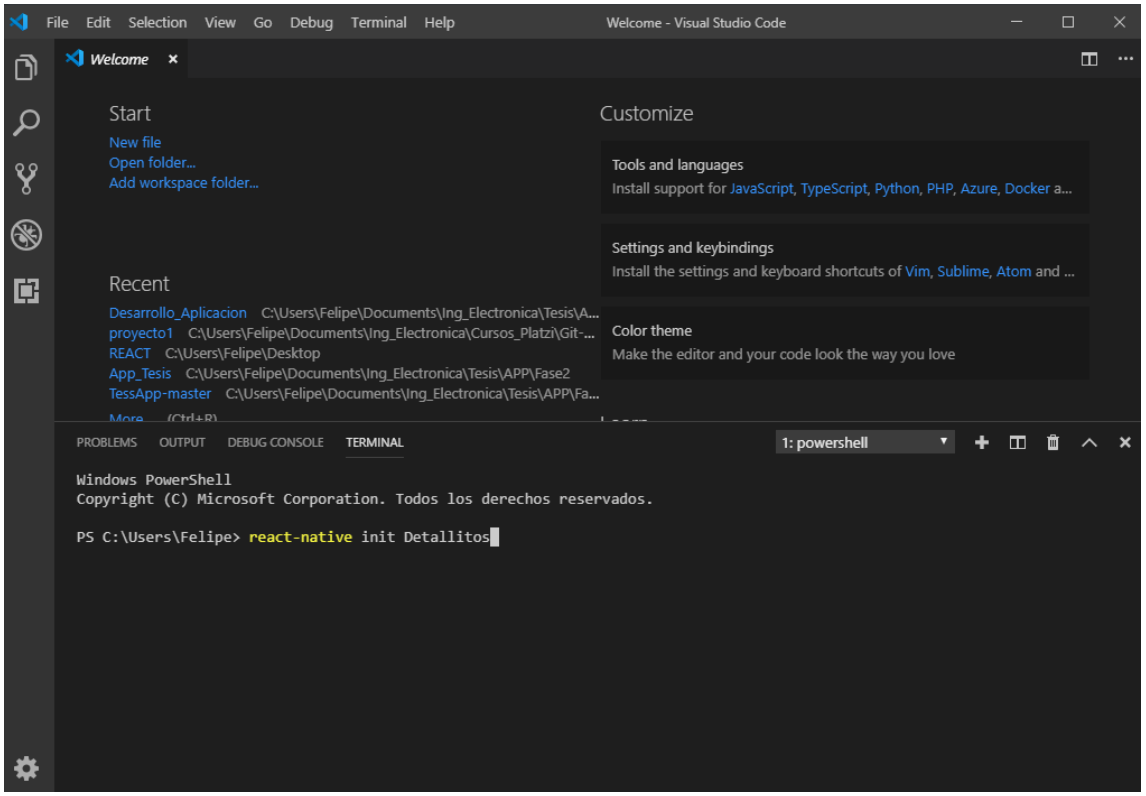
5.1. CONFIGURACIÓN DEL ENTORNO DE DESARROLLO

Para realizar la codificación se decide trabajar con react native, ya que posee paquetes de bloques actualizados para el desarrollo de aplicaciones móviles en Android, es decir cuenta con componentes y APIs, recursos de imágenes, módulos nativos, componentes nativos,etc., y para llevar a cabo la codificación se utilizo el editor de texto visual studio code, el cual es un editor de código fuente ligero pero potente que se ejecuta en su escritorio. En el anexo D se puede observar el proceso de configuración del entorno para poder ejecutar una aplicación de React Native.

Creación del proyecto

Para llevar a cabo el desarrollo de la aplicación Desde la terminal de Visual Studio Code se ejecuta el siguiente comando para generar el proyecto llamado como Detallitos, figura 61.

Figura 23: Desarróllo de la App.



5.2. DEPENDENCIAS DE LA APLICACIÓN

Las dependencias en la aplicación son las librerías o paquetes con las cuales se llevan a cabo todos los procesos que hay para poner en marcha una app tal y como se muestra en el anexo C figura 45, los cuales son descargados por consola por medio de un comando npm el cual es facilitado por nodejs.

Al crear un nuevo proyecto en React Native, éste trae por defecto un archivo javascript llamado App cómo se observa en el anexo C figura 46, siendo el componente principal de la aplicación, es aqui donde se conecta a redux y se importan módulos usando ES6.

5.3. NAVEGACIÓN ENTRE PANTALLAS

Para navegar entre vistas es necesario primero crear un navigator, un archivo donde se relacionen los componentes que envuelven cada vista y el tipo de navegación a utilizar, además de unas configuraciones generales para todas las vistas, dichos ajustes podrán ser cambiados en cada una de las pantallas. Anexo C figura 47, las pantallas de la aplicación están dentro del AppNavigator listas a mostrarse según la petición de la aplicación.

Gracias a que la propiedad de navegación se pasa a todos los componentes enlazados en el navegador de pila, se puede usar el método navigate() anexo C figura 48, para dirigirse a una ruta específica.

5.4. ESTILOS EN REACT NATIVE

Los estilos para react native son un tipo de escritura donde para cada componente se crea un objeto que extiende de la hoja de estilos con el método create(), dependiendo del tipo de componente al que se vaya a pasar este objeto de estilo se pueden definir o no ciertas propiedades. En el anexo C figura 49 se puede ver los estilos aplicados al componente de categoría (componente que se renderiza múltiples veces en la lista de categorías mostrada en la vista Home) donde se aplican estilos a los 3 tipos de componentes mas usados en react native; contenedor, imagen y texto.

5.5. CONECTANDO A LA TIENDA DE REDUX

Siguiendo el principio de inmutabilidad del estado de redux, en la función SET_CATEGORY_LIST que se encuentra en el reducer mostrado como ejemplo en el anexo C figura 50, se devuelve un nuevo estado donde se agrega un objeto action.payload con el parámetro que se pasó como propiedad.

Importando el método conect() de react redux se puede decorar el componente llamando una función que mapea el estado y retorna como propiedad las variables que se indiquen. Anexo C figura 51.

5.6. AUTENTICACIÓN CON GOOGLE

Para autenticar a cada usuario con la cuenta de google se debe llamar una función donde primero se conecten las configuraciones de GoogleSignin con el cliente Web del proyecto Firebase anexo C figura 52, se habilita el modal de google play services y solicita información del usuario junto con un token para validar las credenciales en Firebase.

5.7. PETICIONES A LA BASE DE DATOS

Hay muchos métodos para interáctuar con el servicio de base de datos ofrecido por Firebase, en este proyecto se utilizan 2. El primero es de lectura, aquí es donde la app solicita información a la base de datos para mostrarla al usuario, para ello es necesario referenciar la lista que vamos a leer y hacer la petición con el método once(), luego se presentan estos datos como un objeto JSON que posteriormente se transforman en un array. Este array es enviado a la tienda de redux a través del método dispatch() llamando la acción SET_CATEGORY_LIST (para este caso) anexo C figura 53, una vez haga parte del estado estos datos estarán disponibles para ser persistidos hasta que una nueva petición los sobre escriba.

La segunda función que se utilizó fue la de escritura, donde un usuario después de seleccionar los artículos de su interés y detallar los datos de envío, ésta función escribe dicha información en la base de datos de Firebase en un perfil seleccionado para él dentro de una lista de usuarios con el comando push().set() anexo C figura 54 y es allí donde quedan almacenado todos estos datos.

5.8. FRONTEND

5.8.1. Estructura

Para entender la estructura del proyecto es necesario entender un poco como funciona React, entender que un componente puede anidar otros componentes llamados “hijos” y que este a su vez puede tener un componente llamado “padre” que se montan en la aplicación y se renderizan de manera jerárquica. El desarrollo del proyecto se realiza bajo la filosofía de “Components and Containers”, donde los components llevan toda la parte grafica de la interfaz de usuario y los containers la logica. En su mayoría, los components están definidos por funciones de javascript que importan y utilizan métodos de React, a su vez, se pueden llamar desde otro component o container creando dicha estructura anidada. Por otra parte, los Containers o contenedores son aquellos componentes descritos por clases javascript que se extienden del prototipo “Component” de React, aquí descansa toda la lógica de la aplicación, peticiones a APIs, etc. Los containers también son conocidos como Smart Components.

En detallitos cada Vista esta descrita por una carpeta ubicada en “src/screens” donde contiene todo el código fuente de la misma, esto es:

- Un Smart Component que envuelve toda la vista y renderiza los containers alojados en la carpeta "./containers"
- Una Carpeta Container que almacena los archivos javascript que exportan cada contenedor como una clase JS, ademas pasan ciertas propiedades a los componentes, se suscriben a eventos onPress, adquieren datos desde el backend, pasan propiedades a los componentes que determinan la UI, conexiones al estado de redux, dispatchs a acciones, etc.
- Una Carpeta Components donde estan ubicadas todas las funciones que retornan un componente a renderizar en los containers con toda la información visual preestablecida o adquirida por las propiedades cargadas desde otro componente.

En la carpeta fuente también tenemos un subdirectorio llamado sections, donde gracias a react exportamos componentes comúnmente utilizados, como por ejemplo el layout que envuelve las vistas, un botón que puede ser importado desde cualquier lugar de la aplicación y recibe una propiedad de texto a renderizar, componentes de carga, etc.

Otro directorio llamado firebase contiene todos los archivos de configuración y conexión a base de datos y autenticación de google (también gestionada por firebase)y por último una carpeta llamada redux donde se encuentran todos los archivos de configuración de redux, se crea el store, se configura la persistencia de datos, se crean los reducers y se definen las acciones a utilizar en la aplicación, aquí también definimos en un archivo el estado inicial de la aplicación.

De esta manera Detallitos organizó su código fuente, Además, está la estructura general de React Native, donde múltiples archivos y directorios se crearon al momento de inicializar el proyecto y otos modulos npm fueron cargados posteriormente para suplir los requisitos de la app, muchos de ellos tuvieron que ser modificados de acuerdo a los requerimientos del desarrollo específico. En este directorio raíz también está el subdirectorio android, donde se aloja todo el código java que hace posible que la aplicación corra de manera nativa y se realicen algunas configuraciones especificas para la plataforma

Algunos de los archivos mas importantes aquí son:

- "App.js": Es el componente principal de la aplicación
- "app-navigator.js": contiene toda la configuración y flujo de navegación
- "package.json": un objeto con la información de todas las dependencias, frameworks y versiones utilizados en la app

5.8.2. Usabilidad

La usabilidad que se tiene en cuenta en la aplicación se basa en las interfaces dinámicas con las cuales el usuario puede interáctuar entre si y entender el uso de cada una de

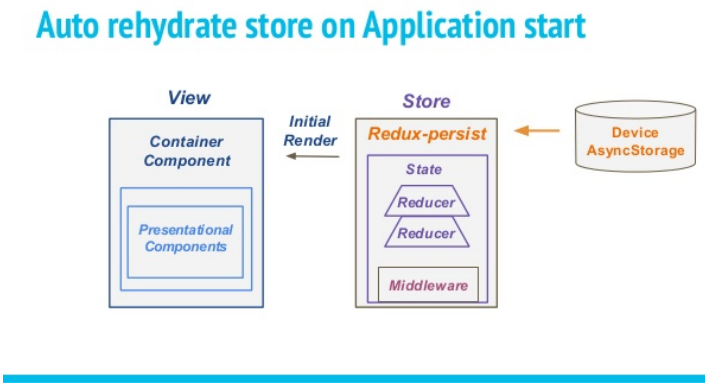
ellas. También, en los patrones de hábitos que los usuarios utilizan comúnmente en otras aplicaciones acelerando así la curva de aprendizaje y haciéndola muy intuitiva a la hora de navegar.

La aplicación actúa de manera eficiente minimizando al máximo los requerimientos al cliente. El inicio de sesión con google facilita enormemente el registro de un nuevo usuario puesto que no solicita datos de ingreso ya que valida esta información de la mano de las APIs de google brindando una experiencia mas agradable al consumidor.

5.8.3. Persistencia de datos

La persistencia de datos en la aplicación, sigue la referencia del flujo de información que se muestra en la figura 24, aquí se muestra lo que sucede cuando un usuario utiliza la aplicación y navega entre esta, los datos generados ya sea por parte del cliente o por peticiones a la base de datos se almacenan de manera asíncrona en la memoria celular pasando primero a través de los reducers al store, cuando la app se va a alimentar nuevamente de los datos estos sigen el mismo flujo, primero al store, y luego a la UI, esto conservando uno de los principios de redux donde la interfaz de usuario solo puede ser alterada por el store y de esta manera asegurar que no hallan cambios inesperados en la aplicación.

Figura 24: Persistencia de datos



Fuente: https://image.slidesharecdn.com/architecturingreactnativeapp-161204102156/95/philip-shurpik-architecting-react-native-app-20-638.jpg?cb=1483563445

5.9. BACKEND

5.9.1. Gestor de conexiones

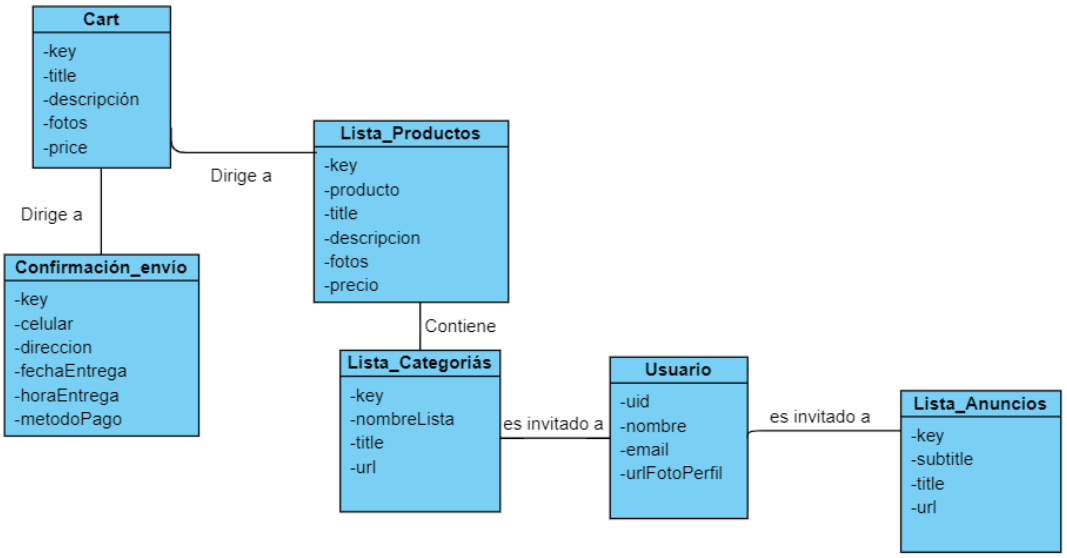
Para la creación de la base de datos se utiliza Firebase como desarrollo de ésta, debido a las ventajas útiles de los servicios que presenta, su facilidad de conexión e implementación. Ya que utiliza el modelo noSQL para almacenar datos en la aplicación en nodos independientes

y no en tablas relacionales. Sin embargo, las lecturas y escrituras a los nodos sí se hacen muchas veces a partir de información previa extraída de otros nodos. Por ejemplo, cuando un usuario realiza la confirmación de envío de una compra realizada, en el nodo de ésta queda guardado el identificador del usuario que la realizó. El esquema utilizado es el diagrama de clases representado en la Figura 25.

La función de cada nodo de la rama principal de la aplicación es la siguiente:

- Usuarios: Datos de los usuarios de la aplicación.
- Lista Productos: Datos de los productos.
- Lista Categorías: Guarda los datos de las diferentes categorías
- Cart: Guarda los datos de los productos seleccionados
- confirmación envío: Guarda los datos de entrega del producto

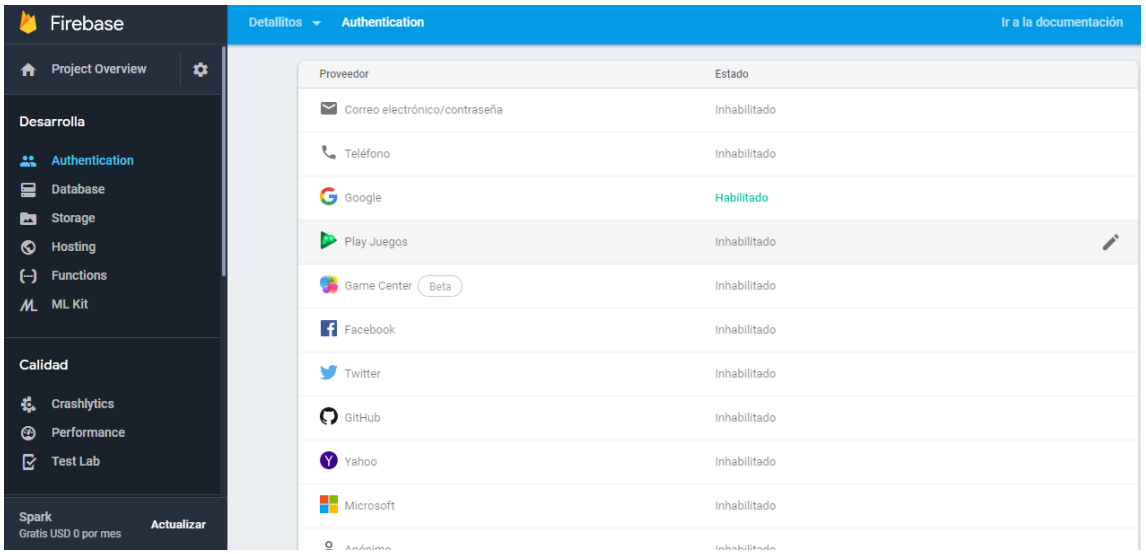
Figura 25: Diagrama de clases Detallitos App



5.9.2. Servicio de autenticación

En esta parte como se muestra en la Figura 26 se habilita el servicio de autenticación por correo. Aquí se pueden habilitar otros métodos de autenticación que presta Firebase, aunque para el proyecto se utilizó el mencionado anteriormente.

Figura 26: Habilitando la autenticación por correo

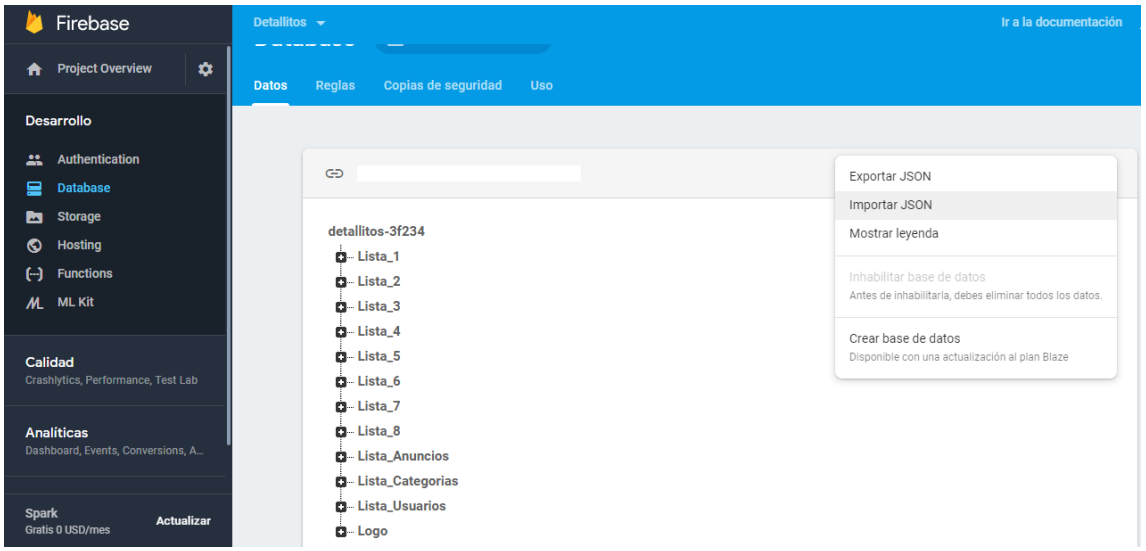


5.9.3. Servicio base de datos en tiempo real

Como se muestra en la Figura 27 se crea la base de datos de la aplicación. Firebase permite importar o exportar archivos Json, donde por medio del editor Visual Code, se realizó la codificación específica con el diseño de la base de datos y se cargo posteriormente. Es aquí donde se encuentran alojados todos los datos de la aplicación, tales como; la lista de usuarios, la lista de categorías con sus respectivos productos y la lista de anuncios.

En el anexo A se muestra cómo se ha estructurado la base de datos Detallitos App, utilizando la interfaz de usuario de Firebase. Estos datos se guardan en un árbol JSON, sin embargo la interfaz de Firebase permite ver la estructura completa. Hay que tener en cuenta que las figuras del anexo resumen la base real utilizada.

Figura 27: Base de datos en tiempo real

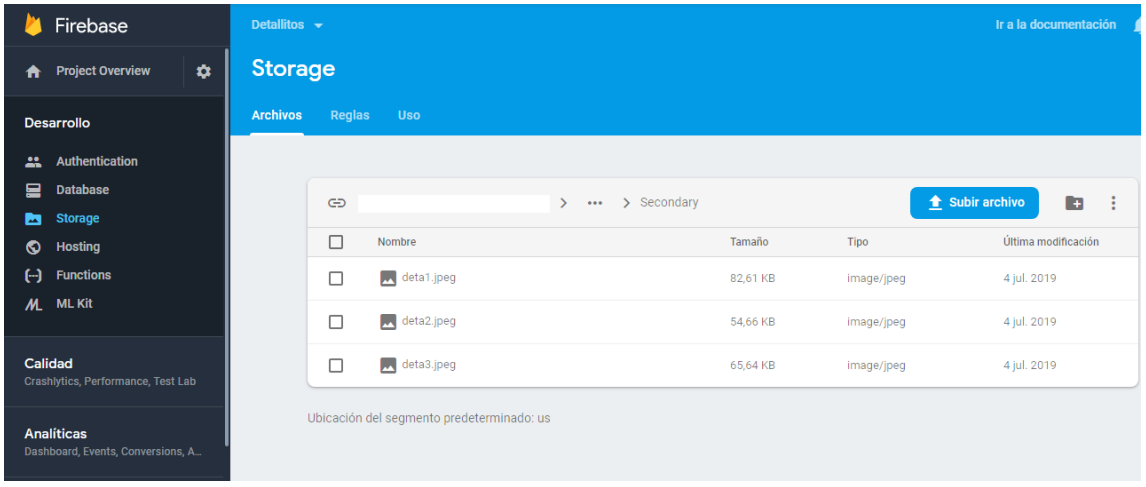


5.9.4. Servicio de almacenamiento

Como se muestra en la Figura 28 se utiliza el servicio de storage para almacenar todas las imágenes utilizadas por la aplicación Detallitos para su posterior uso.

En el anexo B se muestra cómo se ha estructurado el storage Detallitos App, es allí en donde de una manera ordenada se han creada carpetas y subcarpetas dependiendo la lista y las categorías de la app, las cuales contienen las imágenes respectivas, éstas quedan guardadas de una manera estática, asignandoles urls únicas y ésto se hace utilizando la interfaz de usuario de Firebase. Hay que tener en cuenta que las figuras del anexo resumen la base real utilizada.

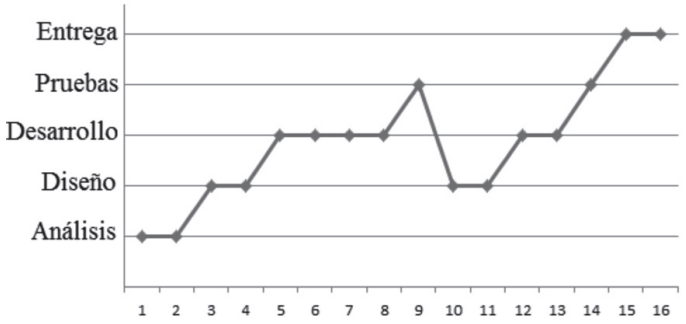
Figura 28: Almacenamiento



6. RESULTADOS

La aplicación móvil Detallilos ofrece un amplio portafolio de productos y servicios a los usuarios de la ciudad de Neiva que deseen adquirirlos para festividades de celebraciones especiales. Detallitos se desarrolló considerando las tecnologías de desarrollo nativo presentes en un tiempo de cuatro meses y obteniendo una evaluación de éxito satisfactoria por parte de un grupo de usuarios; En la figura 29 se muestra la ejecución de cada fase con la dedicación del tiempo en semanas.

Figura 29: Tiempos de ejecución de Detallitos



El servicio desarrollado está soportado por la playstore de Android y solo trabaja para dispositivos móviles que cuenten con este sistema operativo. Esta aplicación va dirigida para toda la población, desde personas jóvenes hasta adultas gracias a su versatilidad en todas sus interfaces.

La aplicación móvil permite a los usuarios utilizar su dispositivo móvil para:

- Ingresar sus datos al momento de realizar el login en la aplicación
- Interáctuar entre sus vistas de manera rápida y sencilla tanto para devolverse a una vista anterior como para regresar a ésta.
- Escoger una categoría específica y desplegar una lista de productos que se encuentren disponibles en el momento.
- Escoger el producto de su interés y desplegar en una vista la descripción de éste y en caso de alguna modificación en la sección de notas adicionales que se encuentra en esta misma vista notificarla.
- Confirmación del envío, en donde aparecerá la fecha para la cual desea el pedido, la dirección de entrega, la hora y por ultimo el medio de pago que le sea mas factible.

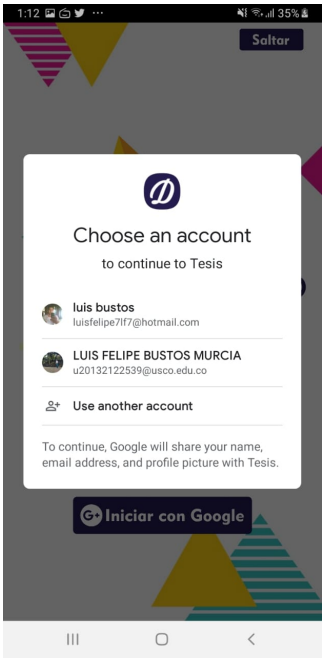
6.1. INTERFAZ

Como resultado final, hemos conseguido una versión de la aplicación con todos los requisitos iniciales, además de otras funcionalidades posteriormente planteadas. La App es totalmente funcional para Android; sin embargo, está desarrollada para iOS. Por último, el acabado final es muy consistente y con un alto grado de usabilidad, como se muestra en las Figuras 30, 31, 32 y 33.

Figura 30: Capturas de la versión final (1/4)

(a) Presentación

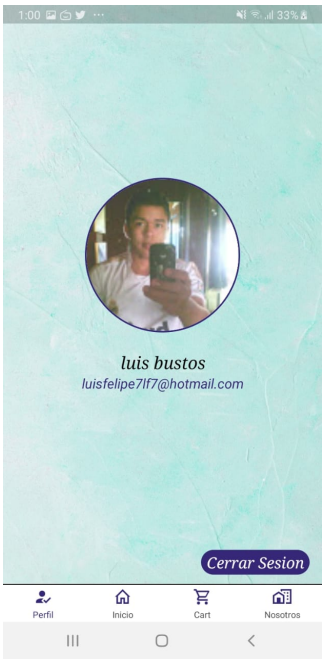
(b) Vista Login y/o Omitir



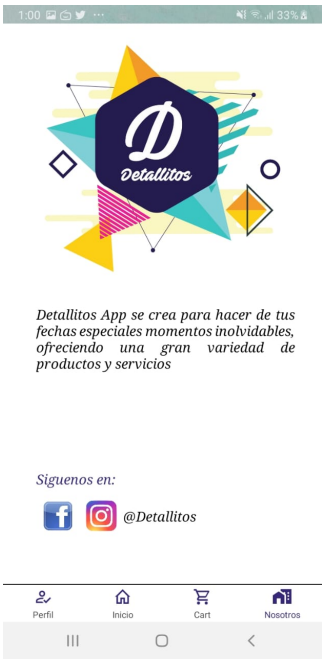
(c) Ingreso

Figura 31: Capturas de la versión final (2/4)

(a) Vista Inicio



(b) Vista Perfil

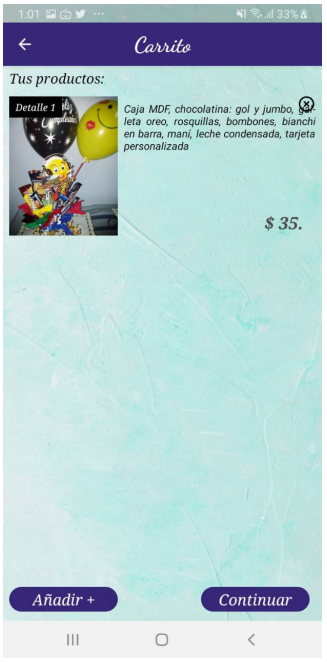


(c) Vista Nosotros

Figura 32: Capturas de la versión final (3/4)

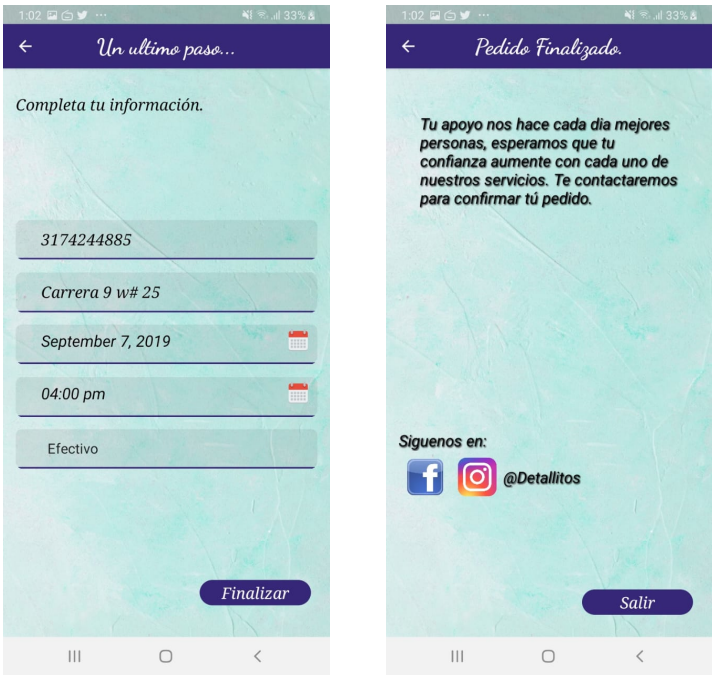
(a) Vista categoría cumpleaños

(b) Vista Producto



(c) Vista Carrito

Figura 33: Capturas de la versión final (4/4)



(a) Vista Confirmación de envío

(b) Vista Final

6.2. SERVIDOR

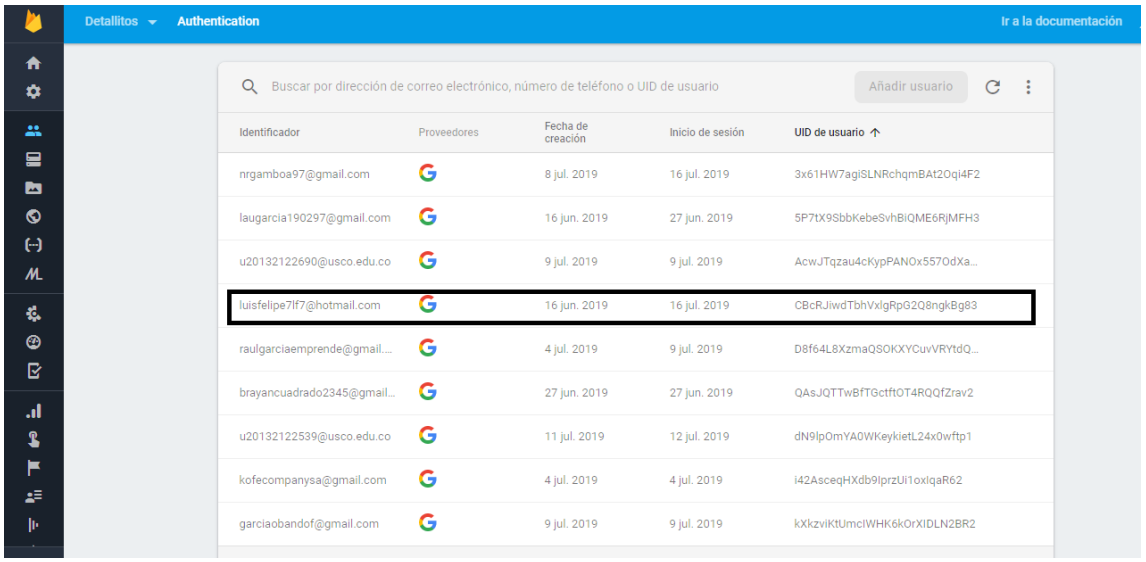
Se realizó la comprobación del login del usuario mediante el ingreso a la aplicación y en la parte de autenticación se obtuvieron todos los datos de los usuarios que previamente utilizaron la App y también se obtuvieron los datos del cliente cuando accedió a realiar una compra.

6.2.1. Autenticación

Las pruebas realizadas al ingreso del usuario e interacción con autenticación de Fireabse, arrojaron resultados de la verificación y si éste ya estaba previamente registrado, dependían del ingreso de cada usuario a la aplicación generando conectividad dentro del aplicativo móvil.

Durante el primer ciclo de pruebas se observó que cuando se le puso el servicio de entrada por correo electrónico, el gradle de Android Studio no compilaba la App por ende no se conectaba a la base de datos y se tuvo que descargar la librería de google-service y despues de ésto fue resuelto el error detectado por lo que fue con éxito la verificación por correo electrónico. En la figura 34 se pueden observar los resultados con éxito entre la App y la autenticación con Firebase.

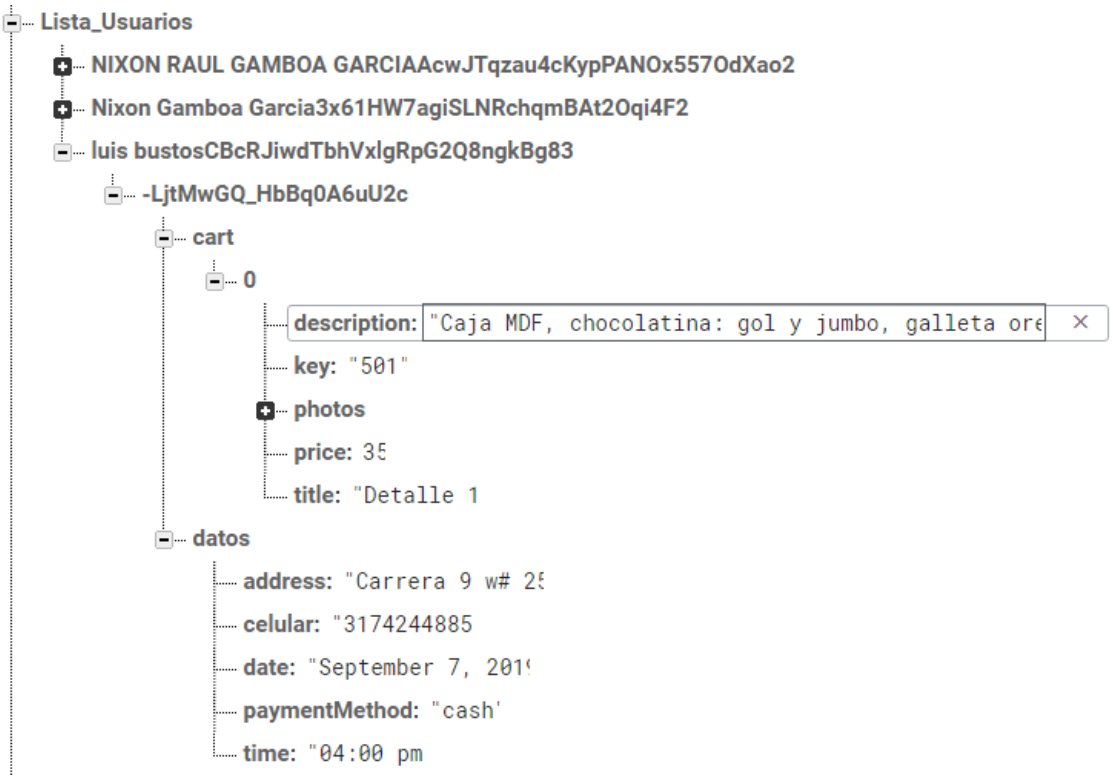
Figura 34: Autenticación de los usuarios



6.2.2. Base de datos

Las pruebas realizadas al momento de efectuar una compra e interacción con la base de datos, arrojaron los datos con éxito indicando, el número de de teléfono, dirección, datos del producto, fecha y hora de entrega, y el método de pago tal y como se muestra en la figura 35.

Figura 35: Base de datos del usuario y el tipo de producto



6.3. ASPECTOS TÉCNICOS Y FUNCIONALES

Los siguientes son los aspectos que se tuvieron en cuenta dentro de las pruebas que se realizaron a la arquitectura de la aplicación móvil:

6.3.1. Conexión

Los dispositivos donde fue instalada la aplicación móvil son un Samsung J8 y Huawei Y7, además se realizáron diferentes pruebas en distintas versiones de android desde la 6 hasta la 9 arrojando resultados con éxito y la mayoría de estos dispositivos cumplen con las siguientes especifiaciones:

- Memoria interna — 64GB.
- Memoria RAM — 4GB.
- Sistema operativo — Android 6.0, 7.0, 8.0, 9.0
- Conectividad — 4G, GPS, Bluetooth 4.2, Dual SIM, Radio FM...

6.3.2. Rendimiento

Cuando hablamos del rendimiento de la aplicación hacemos referencia al consumo de memoria disponible en el dispositivo y al tiempo que le toma mostrar los resultados en pantalla.

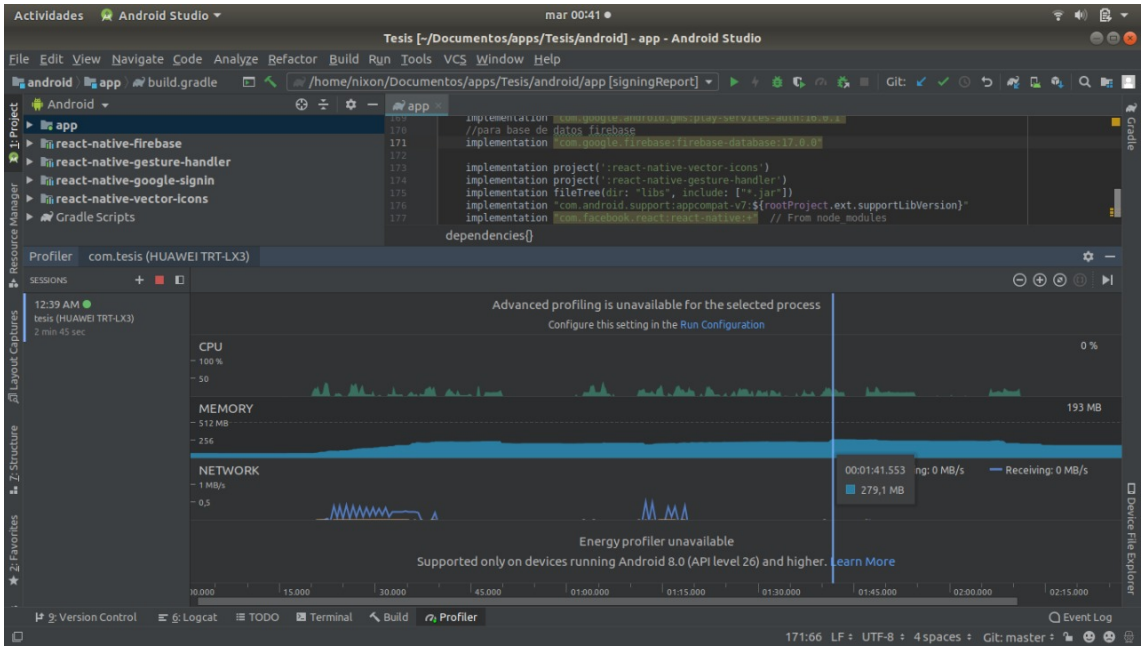
Con respecto al consumo de memoria del dispositivo se recurrió a varias técnicas de optimización del código para que la App final fuera liviana y los datos guardados en la memoria cache no generara mucho gasto.

Se midió el tiempo que le toma a la aplicación pasar del Login a mostrar el resultado en la pantalla del dispositivo la vista de categorias, mientras está conectado a una red inalámbrica con buena y mala intensidad de señal, lo que permitiría obtener mediciones mas aproximadas a un ambiente donde la señal disponible a las condiciones climatológicas, mantenimiento de las entidades que prestan el servicio de internet, entre otras, puedan afectar el rendimiento de la App.

El tiempo tomado por la aplicación para mostrar el resultado en pantalla luego realizar el proceso de Login sobre una red inalámbrica cuya intensidad era óptima fue de 2 segundos aproximadamente mientras que el tiempo tomado sobre una red inalámbrica con una intensidad baja fue de 4 segundos aproximadamente.

En la figura 36 se puede observar los recursos que consume la aplicación en un celular mientras se navega en ésta a lo largo del tiempo, llegando a consumir menos del 50 % de CPU en sus puntos maximos y un consumo de memoria RAM inferior a 279.1MB.

Figura 36: Rendimiento de la aplicación Detallitos



6.3.3. Seguridad

La aplicación móvil ofrece los siguientes niveles de seguridad a los usuarios:

- Los datos ingresados a la aplicación se guardan de manera privada, es decir no se pueden leer, ni escribir por ninguna persona malintencionada y se guardarán siguiendo las políticas de seguridad de información personal, esta base de datos queda aislada, es decir la única aplicación autorizada para utilizarlos es Detallitos para un fin comercial.
- Cuando el usuario es registrado, la aplicación le permite navegar entre sus vistas, tener información de las diferentes promociones, descuentos y posiblemente premios por ser un cliente preferencial.
- La aplicación móvil desarrollada no tiene acceso a la información que se encuentra almacenada en el dispositivo ya que para esto es necesario implementar ciertas librerías que no fueron usadas durante el desarrollo de la misma.

7. CONCLUSIONES

Se diseñó la aplicación móvil Detallitos dando cumplimiento a cada uno de sus requerimientos, es decir se ha desarrollado una herramienta que ofrece diferentes variedades de productos y servicios según la categoría que el usuario escoja a través de dispositivos móviles inteligentes que tengan instalada previamente la aplicación. Esta App ha demostrado un funcionamiento optimo, cumpliendo cada uno de los procesos necesarios para su éxito, incluyendo el ofrecimiento de una navegación sencilla entre sus vistas y una interfaz muy amigable con el usuario.

Detallitos está enfocada en ese complemento que hace falta en la adquisición de productos y servicios en la logística de celebraciones especiales, ya que, en la actualidad para poder acceder a éstos, utilizan métodos muy ortodoxos, tales como las redes sociales para mostrarlos, en los cuales muestran un sinnúmero de productos que se vuelven molestos para al usuario al momento de elegir el de su preferencia y por ende terminan escogiendo cualquier producto aveces no de su agrado, e incluso hacer modificaciones de éstos se torna molesto con quien ofrece el servicio. Ahora se puede ingresar a la aplicación, seleccionar la categoría de su preferencia, elegir uno de los productos que se encuentran listados, y si así lo desean, hay una sección de notas adicionales en donde el cliente puede modificar o agregar mas detalles a dicho producto, luego solicitar la compra de éste por medio de la confirmación de envío. Teniendo así por resultado que el panorama de clientes vaya aumentando por brindar una solución a sus necesidades, tales como, una aplicación versátil, el costo de los productos y servicios amigable con su bolsillo y, por otra, realmente importante escoger lo que éste desea, ya que, de una manera muy práctica los usuarios podrán acceder desde cualquier dispositivo móvil Android a la playstore y descargar la App. Por ultimo generando una nueva forma de empleo, nuevos ingresos y mayor disponibilidad de tiempo a los domiciliarios y todo el recurso humano encargado del diseño de los productos.

A nivel tecnológico se presentaron varias dificultades durante el desarrollo de la App. La principal fue el aprendizaje de nuevas tecnologías, como un nuevo lenguaje de programación para aplicaciones móviles en la interacción con los dispositivos utilizados. Esta dificultad se fue superando en el transcurso del desarrollo de la aplicación, permitiendo a los realizadores aplicar nuevos conocimientos teóricos en un producto final.

La arquitectura de la aplicación fue diseñada de una manera muy práctica en donde se detalla el funcionamiento de ésta, la conexión entre el dispositivo móvil y la base de datos de firebase. Si hay un cambio en la base de datos se visualizará sin ningún problema en la App debido a que en ésta se almacena toda la información y los datos. Por la seguridad y privacidad de de los datos que hay en ésta son guardados de manera privada y sólo los encargados de la App tienen acceso. La aplicación móvil no tendrá acceso a ningún tipo de información que se encuentre almacenada en el dispositivo donde sea instalada.

Firebase es la base de datos implementada en la aplicación móvil debido a que ésta utiliza el modelo no relacional, por ende cuando surja la necesidad de realizar cambios no se

encontrará ningún problema, ademas de que cuenta con innumerables servicios, tales como: la autenticacion; la cual es un servicio encargado de guardar los datos en firebase, si el usuario ya se encuentra registrado se encarga de devolver dicha petición a la App, la base de datos; donde se almacena todos los datos de los usuarios y la aplicación, por último el almacenamiento; en donde se encuentran todas las imágenes de los productos y servicios ofrecidos.

Para incentivar el uso de la aplicación móvil Detallitos, se propone realizar anuncios vía Internet y también por medio de la difusión voz a voz, donde los usuarios puedan con su dispositivo móvil realizar la respectiva descarga de ésta y ponerla en funcionamiento.

8. RECOMENDACIONES

La arquitectura de desarrollo de la aplicación es escalable, lo que permite adaptar nuevas funcionalidades dentro de ésta, cada vez que se requiera para que los potenciales usuarios tengan siempre una experiencia muy confortable cada vez que ingresen a ella.

Tener muy presente al momento de descargar la aplicación móvil de la playstore que se esté utilizando alguna red Wifi con buena latencia, aunque la aplicación móvil no es tan pesada, con conexiones de baja latencia se demoraría unos minutos en descargarse e instalarse en el dispositivo móvil.

Para hacer uso de la aplicación se debe tener conexión a Internet ya sea por medio de datos móviles o conexión Wifi, aunque ésta ofrece experiencia offline, de tal modo que cuando el usuario se desconecta de Internet puede consumir los datos en imágenes persistidos cómo lo haría un local storage en la web, una vez se restablezca la conexión a internet se actualizan los datos si es que éstos cambiaron mientras estuvo desconectado.

Utilizar la aplicación móvil únicamente en la ciudad de Neiva en donde va a estar prestando sus catálogos de productos y servicios, para sus alrededores se puede llegar a un acuerdo con el cliente dependiendo la magnitud de sus requerimientos que sea beneficioso para ambas partes.

Aventurarse a utilizar la inteligencia artificial en la aplicación móvil ya que de ésta manera el cliente podría interáctuar con ésta, indicarle como le gustaría tener el producto de su interés, y al final desplegar algunas opciones que se encuentran en la lista de catálogos, según lo indicado previamente, este campo apenas se esta empezando a explotar en el comercio electrónico y seria una manera muy didáctica y práctica entre la aplicación y el cliente.

BIBLIOGRAFÍA

[1] Open Handset Alliance. *¿Qué se necesita para construir un mejor teléfono móvil?* url http://www.openhandsetalliance.com/. 2019.

[2] Daniel Almeria. *¿Qué es React Native?* url https://clouddistrict.com/blog-dev/que-es-react-native/. 2018.

[3] Xataka Android. *Android domina el tercer trimestre de 2018 con un 86,8cuota de mercado, según IDC.* url https://www.xatakandroid.com/moviles-android/android-domina-tercer-trimestre-2018-86-8-cuota-mercado-idc. 2018.

[4] Santiago Andrés Arango Varón y Danilo Andrés Escobar Buitrago. "Mercapp: Aplicativo móvil para la recepción, solicitud y gestión de domicilios en las tiendas de barrio de Bogotá". B.S. thesis. Facultad de Ingeniería, 2016.

[5] Paco Blanco y col. "Metodología de desarrollo ágil para sistemas móviles. Introducción al desarrollo con Android y el iPhone". En: *Dr. en Ing. Sist. Telemáticos* (2009), págs. 1-30.

[6] DTyOC. *Sistemas operativos.* url https://dtyoc.com/2016/10/03/sistemas-operativos- moviles/. 2016.

[7] Android Developers. *Conoce Android Studio.* url https://developer.android.com/studio/intro?hl=es-419. 2018.

[8] Firebase. *Cloud Storage.* url https://firebase.google.com/docs/storage/?hl=es-419. 2019.

[9] Firebase. *Firebase Authentication.* url https://firebase.google.com/docs/auth?hl=es-419. 2019.

[10] Firebase. *Firebase Realtime Database.* url https://firebase.google.com/docs/database/?authuser=
2019.

[11] Firebase. *Firebase.* url https://firebase.google.com/?hl=es-419. 2019.

[12] Oscar Javier Higuera Benavides. "Veci-Domicilios". En: (2017).

[13] Diana Carolina Lemus Acosta, Luis Alejandro Poveda Poveda y col. "Diseño e implementación de una aplicación móvil para agencias de viajes". B.S. thesis. Universidad Piloto de Colombia, 2016.

[14] Brito Lubo, Mezly Beatriz y Ángel Camilo Pinzón Doncel. "Diseño de una aplicación móvil para la oferta de servicios de información (tendencias, precios y ubicación) enfocado a las prendas de vestir, accesorios y calzado en la ciudad de Bogotá DC". En: (2017).

[15] Maira Cecilia Gasca Mantilla, Luis Leonardo Camargo Ariza y Byron Medina Delgado. "Metodología para el desarrollo de aplicaciones móviles". En: *Tecnura: Tecnología y Cultura Afirmando el Conocimiento* 18.40 (2014), págs. 20-35.

[16] React Native. *Componentes y APIs.* url https://facebook.github.io/react-native/docs/components-and-apis. 2019.

[17] React Native. *Handling Touches*. url https://facebook.github.io/react-native/docs/handling-touches. 2019.

[18] React Native. *Height and Width*. url https://facebook.github.io/react-native/docs/height-and-width. 2019.

[19] React Native. *Layout with Flexbox*. url https://facebook.github.io/react-native/docs/flexbox. 2019.

[20] React Native. *Props*. url https://facebook.github.io/react-native/docs/props. 2019.

[21] React Native. *State*. url https://facebook.github.io/react-native/docs/state. 2019.

[22] React Native. *Style*. url https://facebook.github.io/react-native/docs/style. 2019.

[23] React Native. *Using List Views*. url https://facebook.github.io/react-native/docs/using-a-listview. 2019.

[24] React Native. *Using a ScrollView*. url https://facebook.github.io/react-native/docs/using-a-scrollview. 2019.

[25] React Navigation. *Fundamentos*. url https://reactnavigation.org/docs/en/getting-started.html. 2019.

[26] Redux. *Básico*. url https://es.redux.js.org/docs/basico/. 2019.

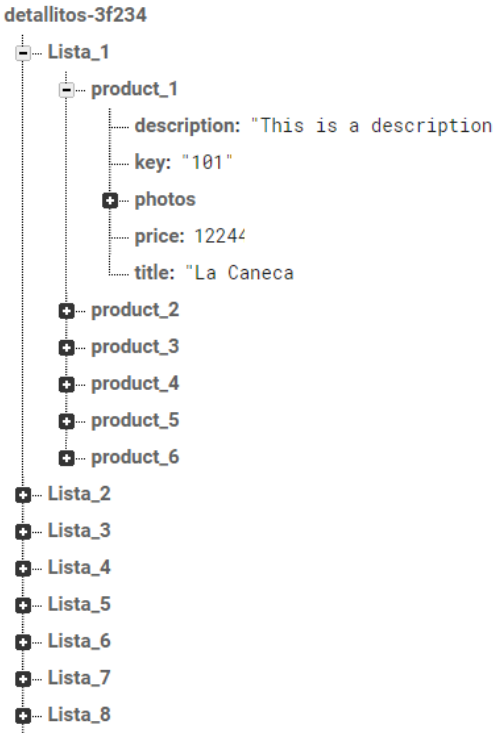
[27] Rosa Marina Sequeira Plama. "Aplicacion movil bajo la plataforma Android para la suscripcion y notificaciones de la Asociacion de Emprendedores de la comunidad de renovacion familiar" HOSANNA", año 2014". Tesis doct. Universidad Nacional Autónoma de Nicaragua, Managua, 2014.

[28] Equipo Facebook open source. *React Native Build native mobile apps using JavaScript and React*. url https://facebook.github.io/react-native/. 2019.

ANEXOS

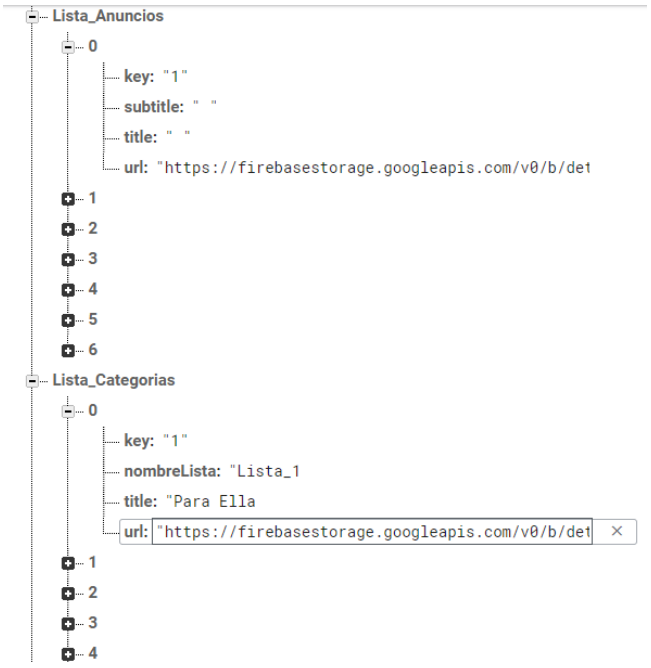
Anexo A. ESQUEMA DE LA BASE DE DATOS

Figura 37: Resumen de la estructura de la base de datos Detallitos (I)



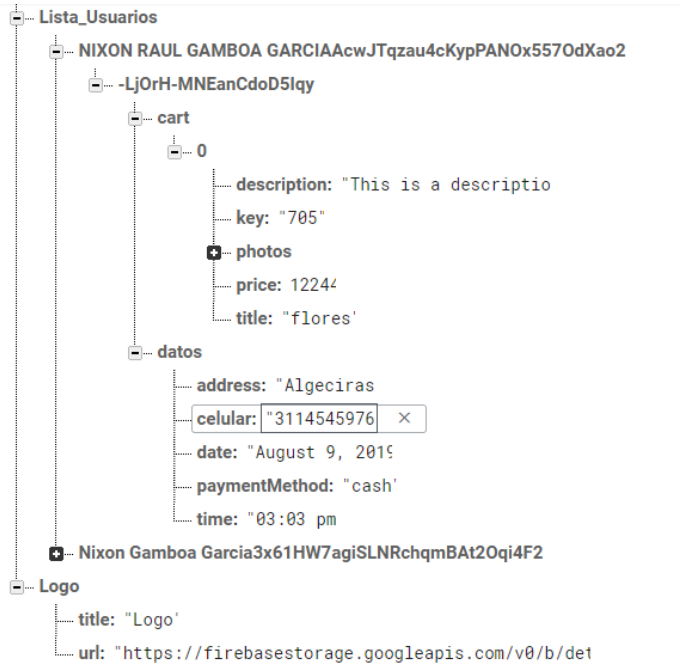
Fuente: Autores

Figura 38: Resumen de la estructura de la base de datos Detallitos (II)



Fuente: Autores

Figura 39: Resumen de la estructura de la base de datos Detallitos (III)



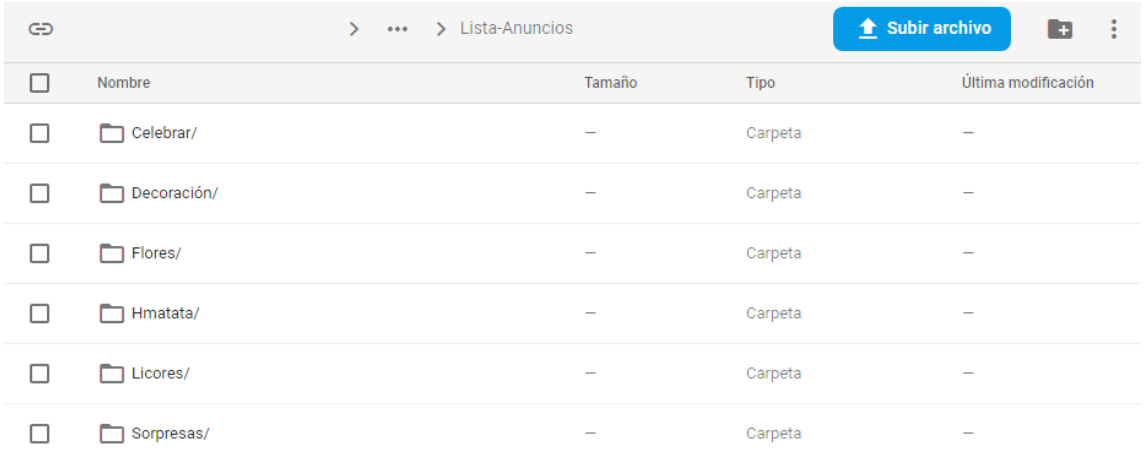
Fuente: Autores

Anexo B. ESTRUCTURA DEL ALMACENAMIENTO

Figura 40: Resumen de la estructura del almacenamiento Detallitos (I)

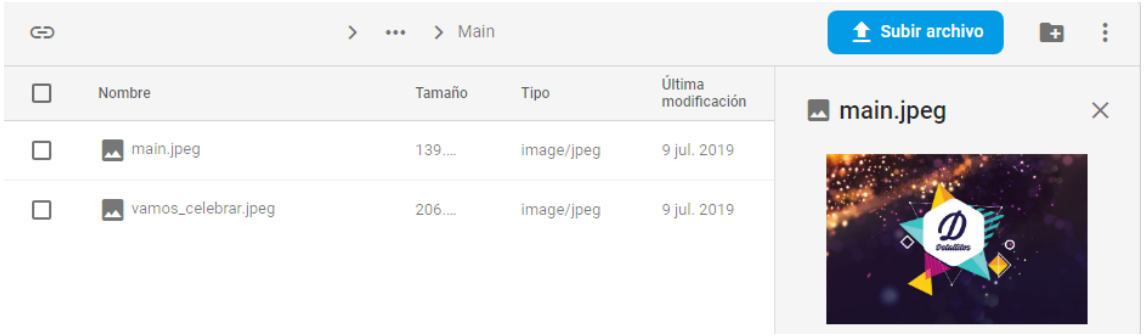
Fuente: Autores

Figura 41: Resumen de la estructura lista de anuncios Detallitos (II)



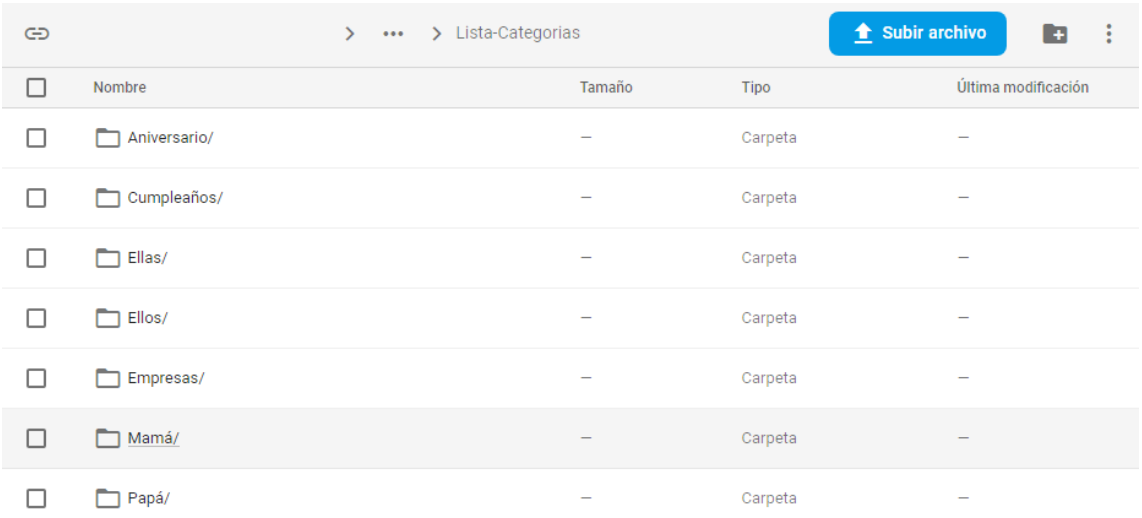
Fuente: Autores

Figura 42: Resumen de la estructura lista de anuncios Detallitos (III)



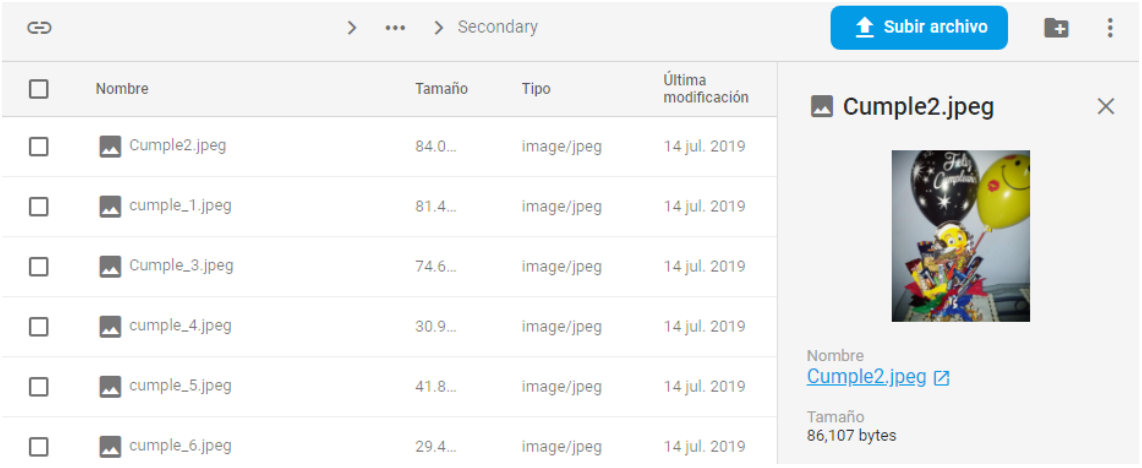
Fuente: Autores

Figura 43: Resumen de la estructura lista de categorías Detallitos (IV)



Fuente: Autores

Figura 44: Resumen de la estructura lista de categorías Detallitos (V)



Fuente: Autores

Anexo C. CODIGO FUENTE

Figura 45: Módulos npm utilizados

```
"dependencies": {
  "core-js": "^3.1.3",
  "firebase": "6.2.0",
  "react": "16.8.3",
  "react-native": "0.59.8",
  "react-native-datepicker": "^1.7.2",
  "react-native-firebase": "5.4.3",
  "react-native-gesture-handler": "1.2.2",
  "react-native-google-signin": "^2.0.0",
  "react-native-snap-carousel": "3.8.0",
  "react-native-vector-icons": "6.5.0",
  "react-navigation": "3.11.0",
  "react-redux": "7.0.3",
  "redux": "4.0.1",
  "redux-persist": "5.10.0",
  "update": "0.7.4"
},
```

Fuente: Autores

Figura 46: Archivo App.js

```
import React, {Component} from "react";
import {store, persistor} from './src/redux/store';
import { Provider } from 'react-redux';
import { PersistGate } from 'redux-persist/integration/react';

import AppNavigator from './app-navigator';
import Load from './src/sections/loading';

class App extends Component {
  render() {
    return (
      <Provider
        store ={store} >
        <PersistGate
          loading={<Load/>}
          persistor={persistor} >
              <AppNavigator />
        </PersistGate>
      </Provider>
    );
  }
}

export default App;
```

Fuente: Autores

Figura 47: Navegación entre pantallas

```
const AppNavigator = createStackNavigator(
  { Home,
    Category,
    Article,
    Cart,
    RegisterForm,
    Finish
  },
  {
    initialRouteName:'Home',
    defaultNavigationOptions: {
      title:'Detallitos',
      gesturesEnabled:true,
    },
    headerMode:'screen',
    cardStyle:{ backgroundColor: '#47add4'},
    headerLayoutPreset:'center',
    headerTransitionPreset:'uikit',
  }
);
```

Fuente: Autores

Figura 48: Método navigate()

```
this.props.navigation.navigate('Cart')
```

Fuente: Autores

Figura 49: Estilos en react native

```
const styles = StyleSheet.create({
    container:{
        margin:3,
        borderRadius: 12,
        overflow: 'hidden',
        borderColor: '#6664',
        borderWidth: 2,
        height: 110,
        flex:1,
    },
    category:{
        width: '100%',
        height: '100%',
        justifyContent:'center',
        alignItems: 'center',
        resizeMode:'cover',
    },
    textContainer:{
        justifyContent:'center',
        alignItems: 'center',
        backgroundColor: '#fff7',
        borderColor: '#fff2',
        borderWidth: 3,
        borderTopRightRadius: 25,
        borderBottomLeftRadius: 25,
    }
```

Fuente: Autores

Figura 50: Conectando al store de redux (I)

```
function reducer(state = stateInit, action){
    switch (action.type){
        case 'SET_CATEGORY_LIST':{
            return{...state, ...action.payload}
        }
```

Fuente: Autores

Figura 51: Conectando al store de redux (II)

```
import React,{Component} from 'react';
import {FlatList, ActivityIndicator, View, Text} from 'react-native';
import {connect} from 'react-redux';

import Product from '../components/product';
import Layout from '../components/layout';

function mapStateToProps(state){
    console.log(state)
      return {
        list:state.products
      }
    }
```

Fuente: Autores

Figura 52: Autenticación con google

```
// Calling this function will open Google for login.
async function googleLogin() {
  try {
    await GoogleSignin.configure({
      webClientId: '812674531905-qqra17572up6nkicqtuicug52e4c2osl.apps.googleusercontent.com'
    });
    await GoogleSignin.hasPlayServices({
      showPlayServicesUpdateDialog: true,
    });
    const userInfo = await GoogleSignin.signIn();
    // create a new firebase credential with the token
    const credential = firebase.auth.GoogleAuthProvider.credential(userInfo.idToken, userInfo
    // login with credential
    const firebaseUserCredential = await firebase.auth().signInWithCredential(credential)
```

Figura 53: Leer datos de Firebase

```
firebase.database().ref('Lista_Categorias').once('value',(data)=>{
const categoryList = data.toJSON();
const list = Object.values(categoryList)
this.props.dispatch({
  type:'SET_CATEGORY_LIST',
  payload:{ categories:list }
  })
}).catch((error)=>{
  console.log(error);
})
```

Figura 54: Escribir datos en Firebase

```
firebase.database().ref('Lista_Usuarios/'+name+uid).push().set(
    {
        datos:this.props.data,
        cart:this.props.cart,
    }
```

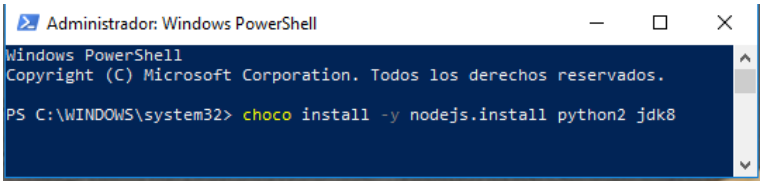
Anexo D. CONFIGURACIÓN DEL ENTORNO

Instalación de dependencias

Se instala Node, la interfaz de línea de comandos de React Native, Python2, un JDK, Android Studio y visual studio code de la siguiente manera:

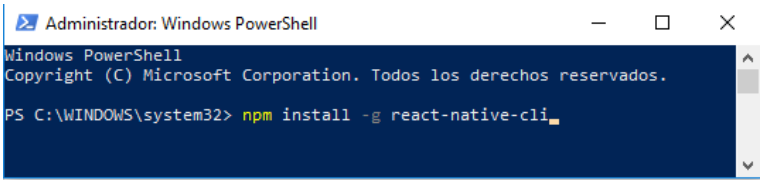
- Chocolatey: Se instala este administrador de paquetes popular para Windows.
- Node, Python2, JDK: En el Símbolo del sistema de administrador, se ejecuta el siguiente comando mostrado en la figura 55.

Figura 55: Instalación Node, Python2, JDK



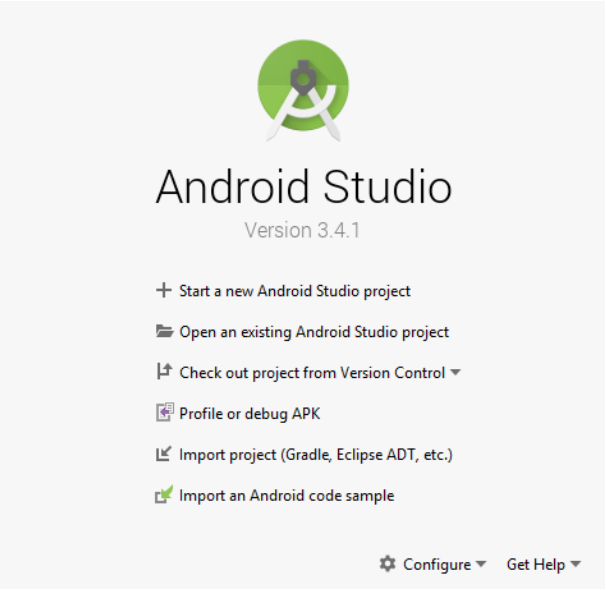
- The React Native CLI: Se ejecuta el siguiente comando en shell mostrado en la figura 56, el cual instala la interfaz de línea de comandos de React Native.

Figura 56: Instalación The React Native CLI



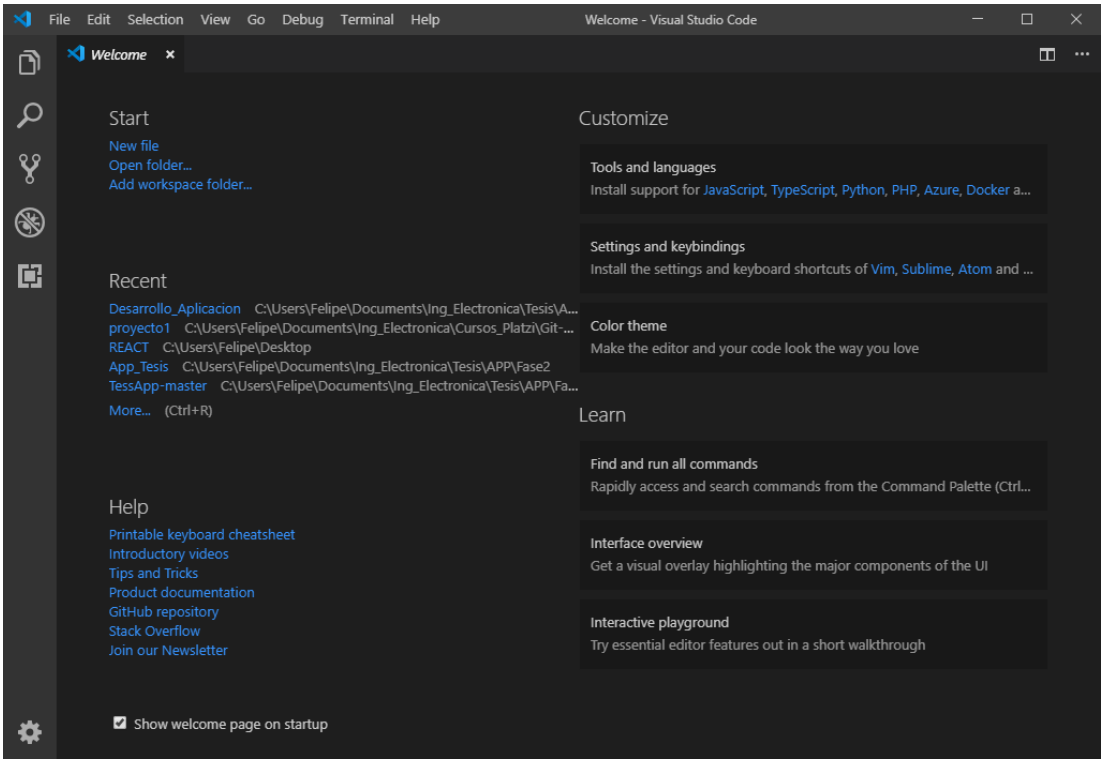
- Android Studio: Se descarga y se instala de su pagina oficial la version 3.4.1, figura 57.

Figura 57: Instalación Android Studio



- Visual Studio Code: Se descarga y se instala de su pagina oficial la version 1.35.1, figura 58.

Figura 58: Instalación Visual Studio Code

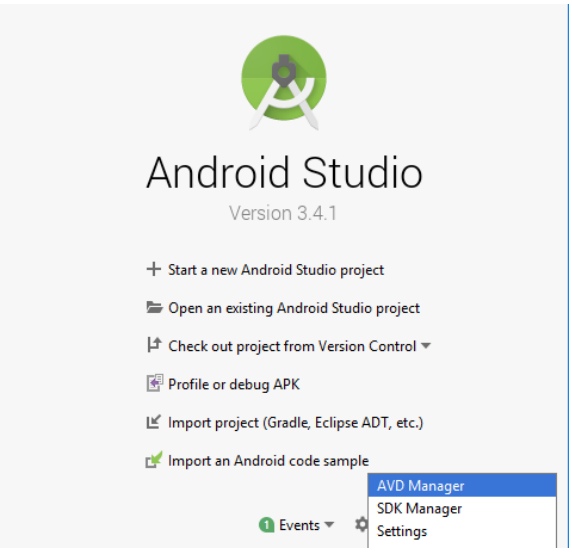


Configuración del emulador de Android Studio

Para poder simular la aplicación en tiempo real, se debió hacer uso del emulador que el software Android Studio proporciona, por la tanto se tuvo que crear este dispositivo de la siguiente manera:

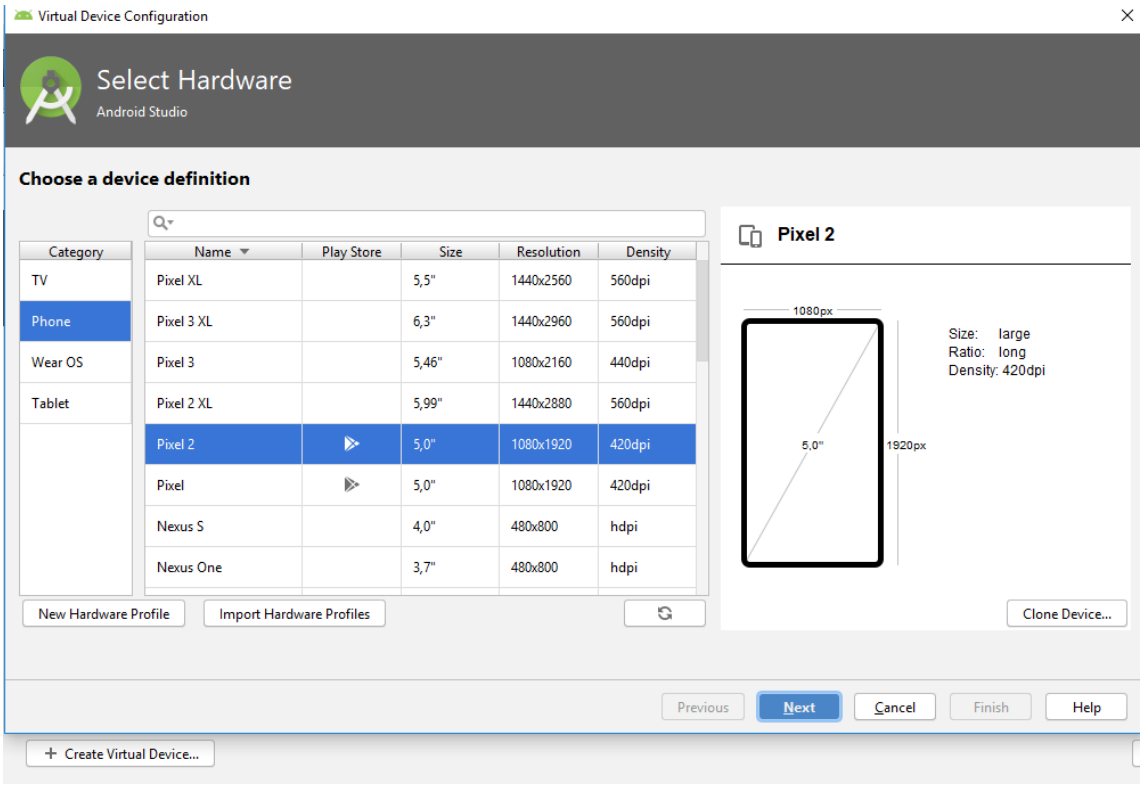
- Paso 1: Se ejecuta el software android Studio y se da clic en la opción configuración-AVD Manager

Figura 59: Configuración-AVD Manager



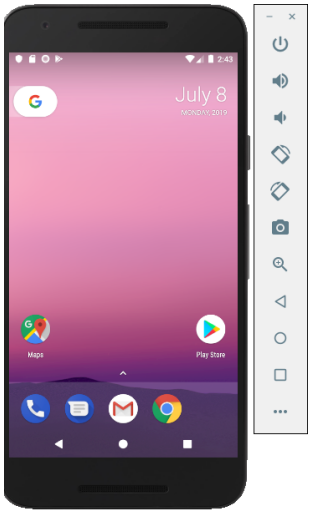
- Paso 2: En la interfaz de AVD Manager se busca la opción create virtual device para configurar el nuevo dispositivo con las características adecuadas para simular la aplicación.

Figura 60: crear un dispositivo virtual



- Paso 3: Después de haber terminado con las exigencias del dispositivo virtual se obtiene como resultado éste y listo para simular la App.

Figura 61: Dispositivo virtual

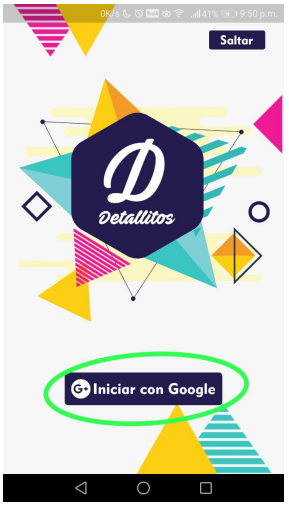


Anexo E. MANUAL DE USUARIO

Figura 62: Abriendo la aplicación

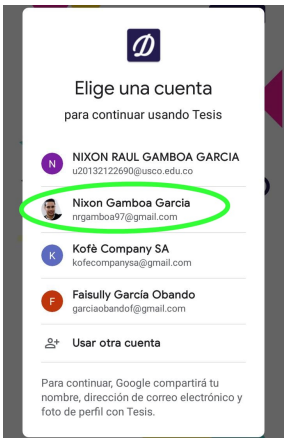
Al iniciar la aplicacion se encontrará con la opcion "Iniciar con Google" que abrirá una interfaz donde se le permitirá utilizar su cuenta personal para autenticarse como ususario en la aplicacion.

Figura 63: Inicio de sesion



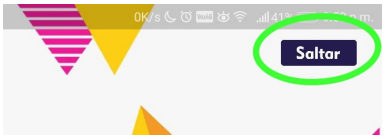
Seleccione la cuenta con la que desea iniciar sesion o agregue una cuenta diferente a las que tiene relacionadas en el dispositivo.

Figura 64: Seleccion de cuentas



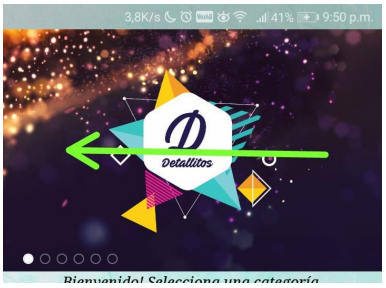
Tambien puede omitir temporalmente este paso para navegar en la aplicacion y ver el catalogo de productos y servicios.

Figura 65: Saltar



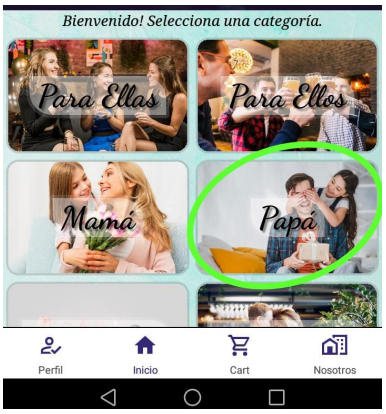
Una vez realizado este paso encontrará la vista principal de la aplicacion, aqui va a encontrar un carrusel informativo donde podra obserbar promociones, descuentos. Puede desplazar este carrusel lateralmente o sostener para pausar la auto reproduccion.

Figura 66: Carrusel de anuncios



Tambien vera una lista de categorias donde podrá encontrar justo lo que esta buscando y para la persona que lo está buscando, si hace tap sobre alguna categoría abrirá una lista de productos relacionados a esta categoría.

Figura 67: Categorías



En la lista de productos podrá ver una imagen reducida del articulo, el nombre y el precio del mismo. Si quiere obtener mas detalles puede dar tap sobre la imagen y se desplegará una vista del articulo, puede retroceder utilizando los botones de android o haciendo tap en la flecha hacia atrás que está junto al nombre del producto.

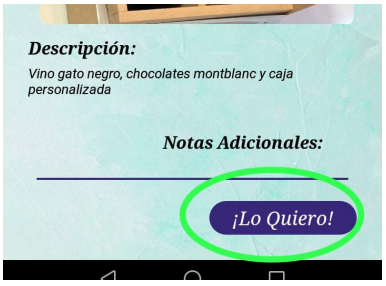
Figura 68: Productos y/o servicios

Si entra a la vista del articulo puede ver un carrusel con una o mas imagenes del producto y/o servicio, deslice horizontalmente para saltar entre ellas.

Figura 69: Carrusel de imagenes del producto

En esta vista tambien encontrará una breve descripcion y la opcion de personalizar su compra mediante notas adicionales, si esto es lo que está buscando presione el boton “¡Lo Quiero!” y asi lo agregará al carrito de compras.

Figura 70: Seleccionar un item

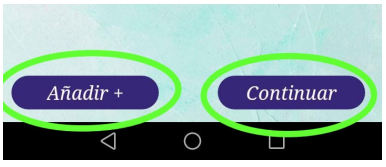


Si cambia de opinion puede remover su seleccion presionando la “X” que acompaña cada item en el carrito.

Figura 71: Eliminar de la lista

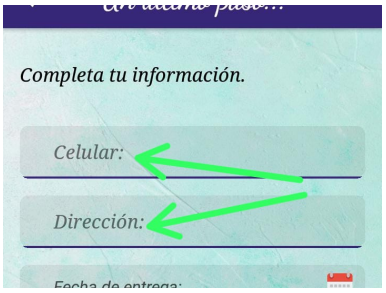
Si desea añadir otro articulo a la lista puede seleccionar la opcion “Añadir +”, pero si cree que eso es todo lo que está buscando presione el botón “Continuar”, en este momento debe haber verificado su inicio de sesion, si no lo ha hecho no se preocupe, el sistema se lo recordará.

Figura 72: Botones del carrito



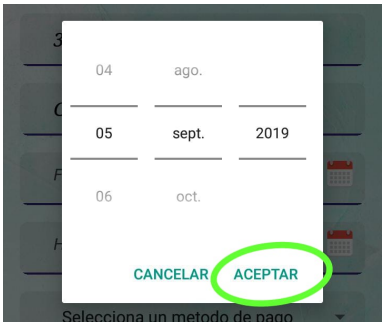
Ingrese un numero de celular válido (10 digitos) y la direccion donde se hará la entrega del domicilio.

Figura 73: Datos de envío



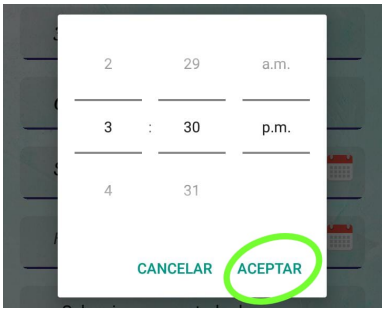
Seleccione la fecha de entrega, recuerde que la fecha maxima será hasta 3 meses despues del dia en curso, la fecha minima dependerá de la disponibilidad de los productos y/o servicios.

Figura 74: Fecha de entrega



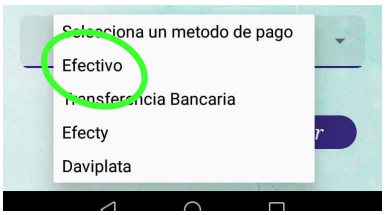
Luego seleccione la hora de entrega, recuerde que el servicio de domicilio está habilitado entre las seis de la mañana y diez de la noche, el sistema no le permitirá seleccionar otro horario fuera del establecido.

Figura 75: Hora de entrega



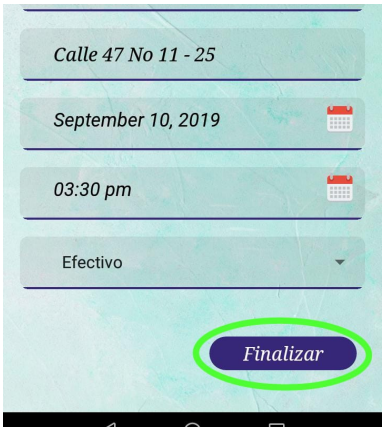
Finalmente el método de pago, seleccione el que mas se ajuste a su comodidad.

Figura 76: Método de pago



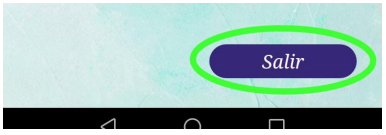
Por ultimo, el boton finalizar enviará su pedido junto con los datos de entrega al servidor del sistema.

Figura 77: Finalizar



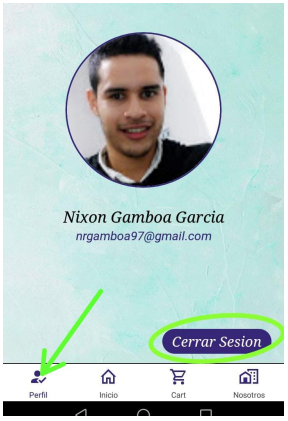
Una ultima pantalla confirmará el exito de la operación y le dara la opcion de salir.

Figura 78: Salir



Si desea cerrar sesion en el dispositivo puede hacer tap en la pestaña de perfil y presionar el boton “Cerrar Sesion”.

Figura 79: Cerrar sesion

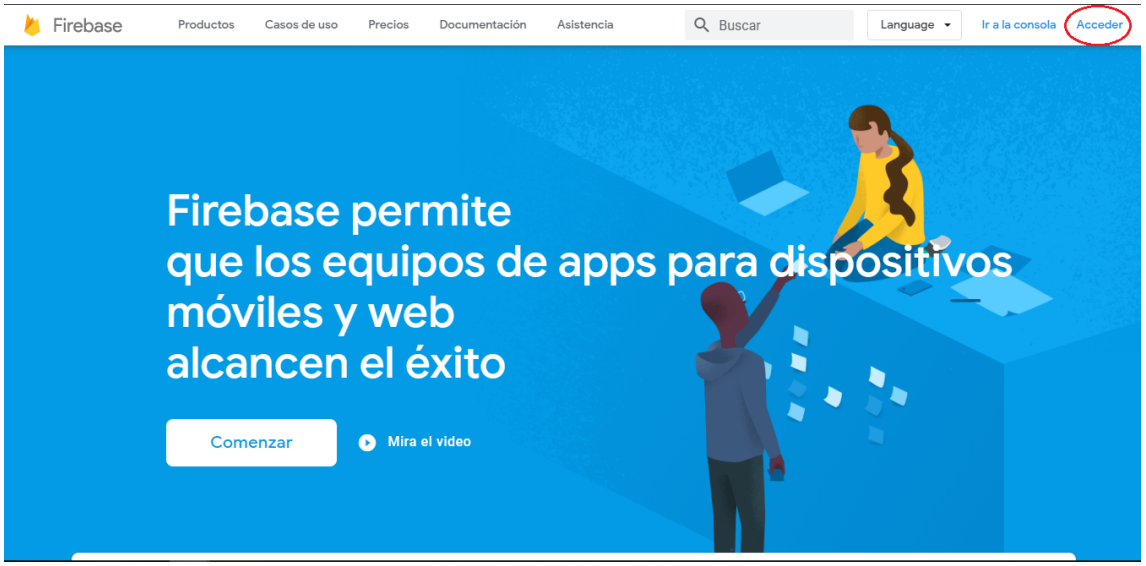


Anexo F. MANUAL DE ADMINISTRADOR

Ingreso al servidor Firebase

En el campo Acceder, se permite el aceceso al inicio de sesion.

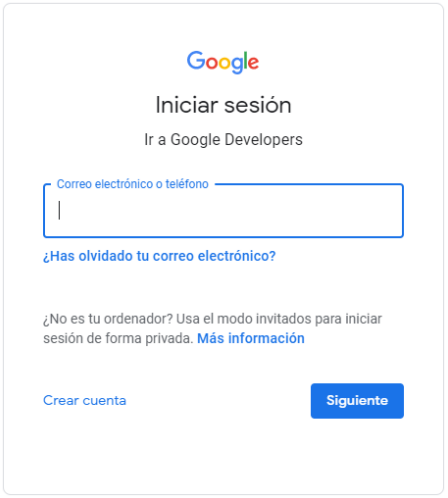
Figura 80: Firebase



Inicio de sesion

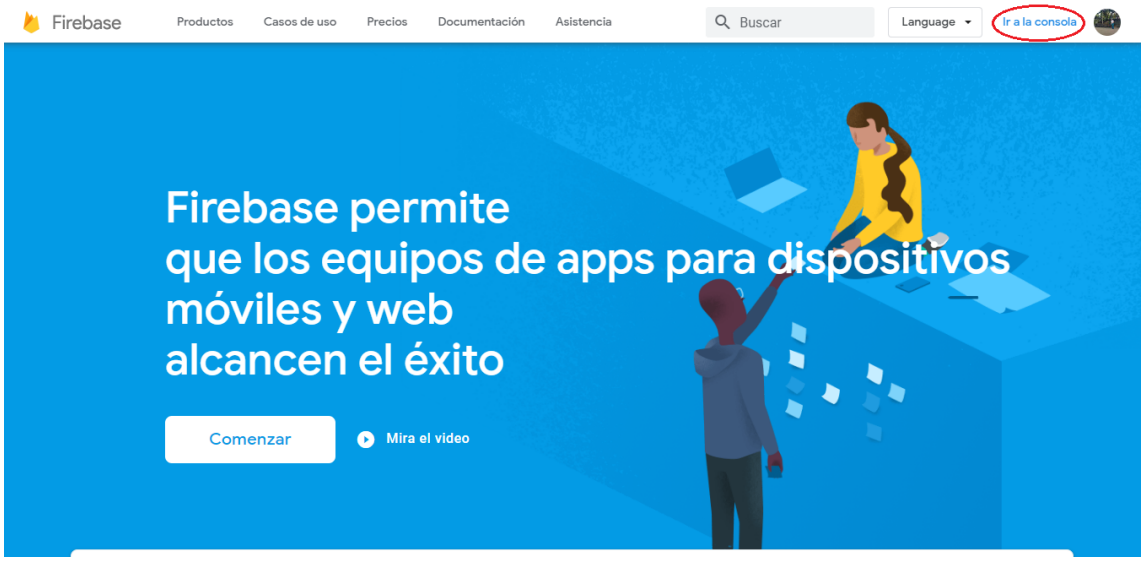
Se debe ingresar con una cuenta de google

Figura 81: Inicio de sesion



Después del ingreso, se va a la consola de Firebase ya que es alli en donde se encuentra el servidor del proyecto.

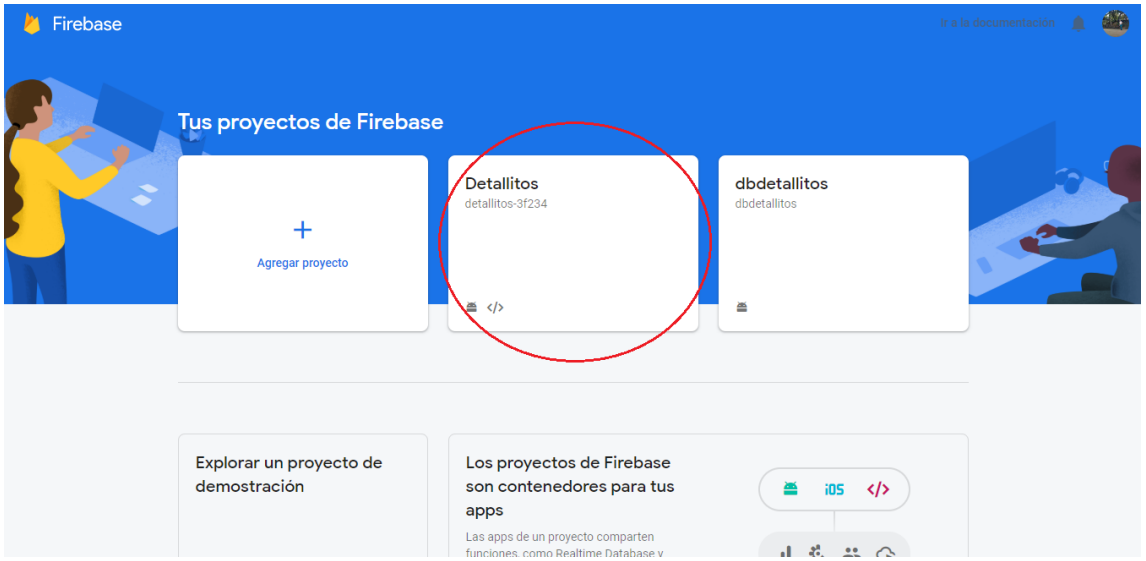
Figura 82: Ingreso a la consola de Firebase



Proyectos en Firebase

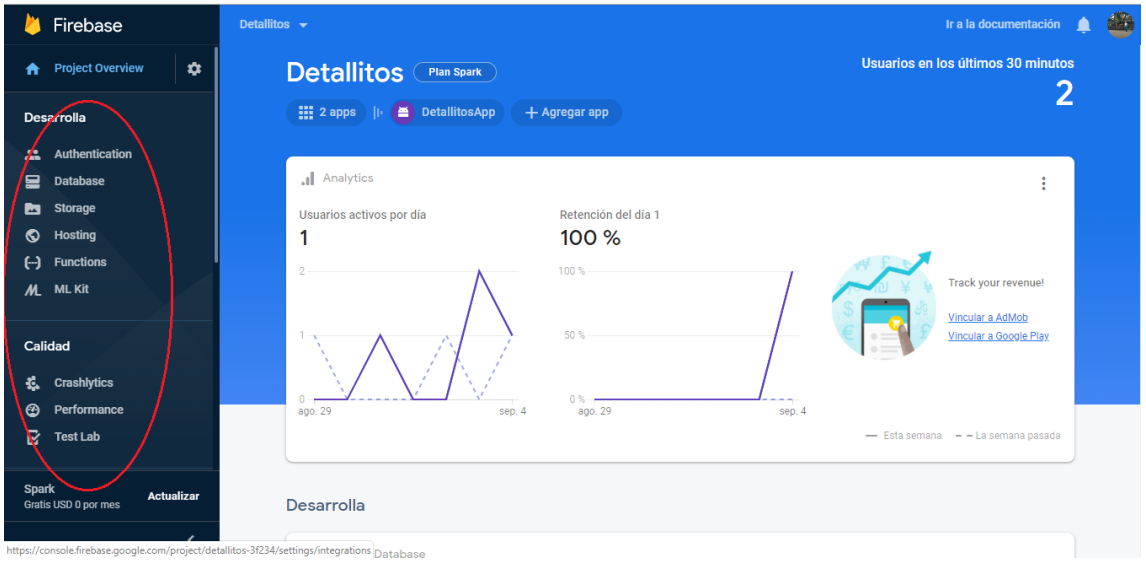
Una vez se ingresa a consola, en Firebase se despliega la lista de proyectos en los cuales se ha estado trabando y se debe seleccionar el de interés.

Figura 83: Proyectos en Firebase



Luego se observa la descripción general del proyecto y el acceso a todos los servicios con los que éste cuenta.

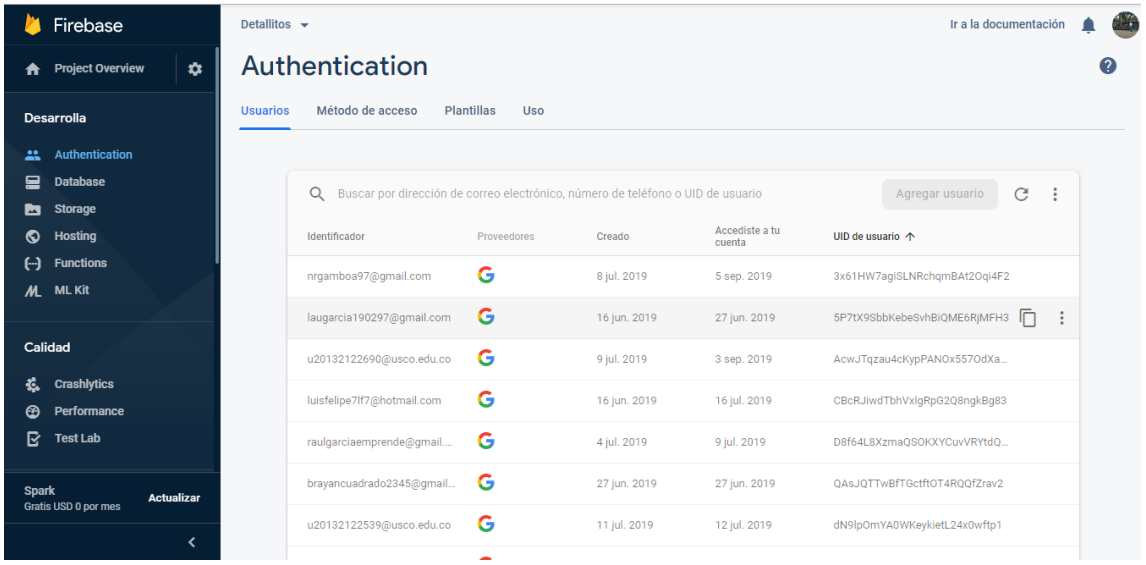
Figura 84: Descripción general del proyecto y sus servicios



Servicio de Autenticación

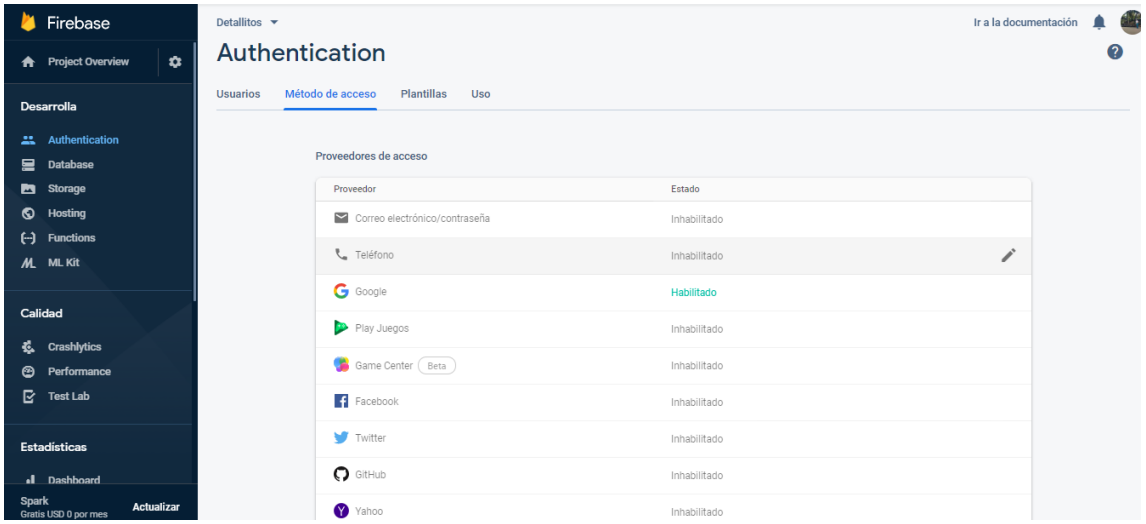
Es aquí en donde se puede observar los usuarios activos de la aplicación y realizar alguna modificación respectiva o búsqueda dependiendo el caso.

Figura 85: Servicio de Autenticación



Se puede encontrar con los diferentes métodos de acceso que pueden ser habilitados para la aplicación.

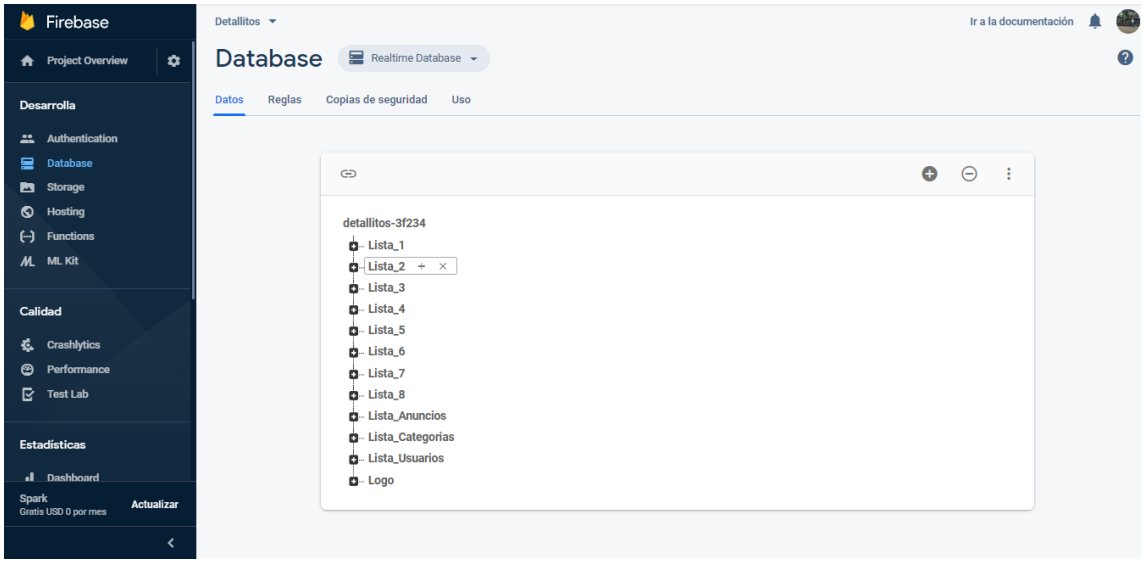
Figura 86: Métodos de acceso



Servicio de Base de datos

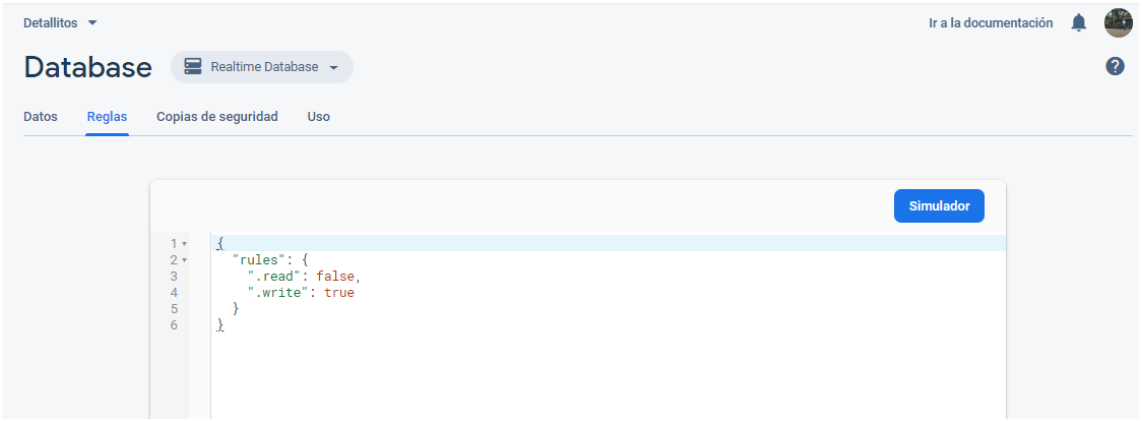
Es aquí en donde se puede observar la base datos utilizada para la aplicación, agregar o eliminar campos según sea el caso.

Figura 87: Base de datos



La privacidad de la base de datos se encuentra en las reglas y es aquí en donde se dan los accesos permitidos para que la App se pueda conectar.

Figura 88: Reglas



Servicio de Almacenamiento

Si necesita almacenar imágenes, se debe ir al servicio de almacenamiento en donde se cargan y se guardan con su respectivo link asignado por Firebase o por el administrador según el caso.

Figura 89: Servicio de Almacenamiento

Para crear una carpeta se debe dar clic en el logo de archivo(+)

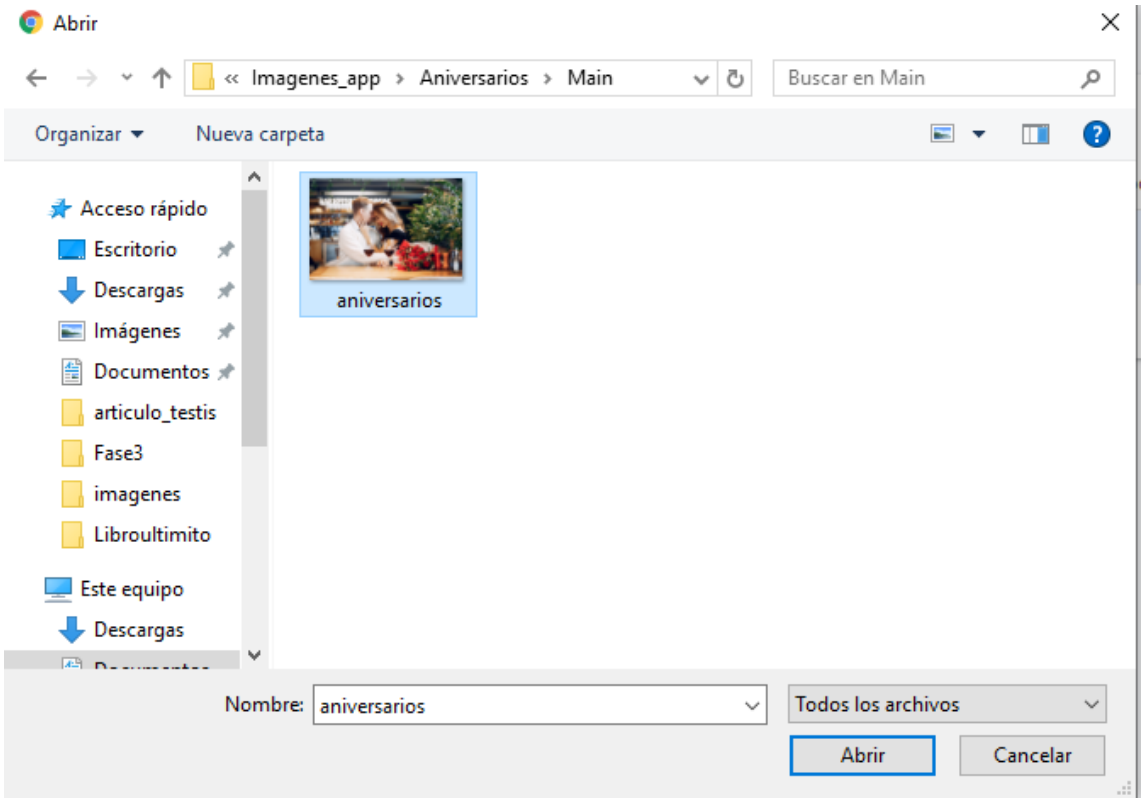
Figura 90: Crear carpeta

Para cargar una imágen se debe ir a la opción subir archivo.

Figura 91: Cargar imagen

Se debe escoger la ruta específica en donde se encuentra la imágen en el ordenador y dar clic en abrir

Figura 92: Abrir imagen



Luego se podrá visualizar la imágen cargada previamente, con su ubicación de almacenamiento y url de descarga.

Figura 93: Abrir imagen