ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

DESARROLLO DE UN PROTOTIPO DE APLICACIÓN ANDROID PARA EL MANEJO DE FINANZAS PERSONALES

TRABAJO DE INTEGRACIÓN CURRICULAR PRESENTADO COMO REQUISITO PARA LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN TECNOLOGIAS DE LA INFORMACION

CRISTIAN ALEXANDER RODRÍGUEZ DURÁN

cristian.rodriguez@epn.edu.ec

DIRECTOR: DR. LUIS FELIPE URQUIZA AGUIAR

luis.urquiza@epn.edu.ec

Quito, febrero 2022

CERTIFICACIONES

Yo, Cristian Alexander Rodríguez Durán declaro que el trabajo de integración curricular aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

CRISTIAN ALEXANDER RODRÍGUEZ DURÁN

Certifico que el presente trabajo de integración curricular fue desarrollado por Cristian Alexander Rodríguez Durán, bajo mi supervisión.

LUIS FELIPE URQUIZA AGUIAR
DIRECTOR

DECLARACIÓN DE AUTORÍA

A través de la presente declaración, afirmamos que el trabajo de integración curricular aquí descrito, así como el producto resultante del mismo, son públicos y estarán a disposición de la comunidad a través del repositorio institucional de la Escuela Politécnica Nacional; sin embargo, la titularidad de los derechos patrimoniales nos corresponde a los autores que hemos contribuido en el desarrollo del presente trabajo; observando para el efecto las disposiciones establecidas por el órgano competente en propiedad intelectual, la normativa interna y demás normas.

CRISTIAN ALEXANDER RODRÍGUEZ DURÁN

LUIS FELIPE URQUIZA AGUIAR

.

DEDICATORIA

Dedico el presente trabajo a mi familia y a mi amigo "Tommy" por el apoyo y cariño.

AGRADECIMIENTO

Gracias papá Dios por todo lo que has puesto en mi frente, por lo superado y por lo que me queda por superar, gracias por las personas que pusiste en mi camino durante mi paso por la Universidad.

A mi amiga Grace Mayanquer por todo el apoyo, consejos, horas de estudio y ayuda dentro y fuera de clases muchas gracias.

Muchas gracias al Doctor Luis Urquiza por todo el apoyo, paciencia y comprensión en el desarrollo de este proyecto, su esfuerzo fue fundamental y estoy seguro que sin su ayuda no hubiera logrado culminar.

A mi amiga Dianita Sigcha por las habladas, risas y por su constante apoyo en mi paso por la universidad.

A mi madre por su apoyo, paciencia y por guiarme a construir la persona que soy.

A las personas que me permitió conocer la Universidad y contribuyeron de alguna forma en mi paso por la misma, Lucho Herrera, Juan Maldonado, Andrés Alcocer, Ernesto Gangotena, Erik Duchicela, Erika Viteri, Fabián Olmedo, Sebas Bahamonde, y al equipo FyS.

A Gianni Défaz gracias por tu apoyo en gran parte de mi paso por la universidad, por las fuerzas y ánimos que me brindaste.

Finalmente tengo que agradecer a la familia Défaz Visuete que en momentos difíciles me supieron extender su mano, misma que en el trascurso de mi vida universitaria ha sido de mucha ayuda.

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE CÓDIGOS

RESUMEN

El presente trabajo consiste en el desarrollo de un prototipo de aplicación Android para el manejo de finanzas personales utilizando el framework React Native junto con los servicios brindados por Firebase, el cual permite al usuario tener un registro de sus consumos de forma semiautomática para ayudarlo a evitar una situación de sobreendeudamiento, esto al realizar la lectura de correos electrónicos emitidos por la entidad bancaria después de un consumo y cuando el usuario desee registrar uno manualmente, el aplicativo organizara dicha información entregando al usuario un detalle organizado por tipo de consumo y fecha.

El presente trabajo se encuentra detallado en cuatro capítulos, donde aspectos generales, descripción de herramientas utilizadas y tecnología aplicada constan en el capítulo uno

El segundo capítulo cuenta con la recolección de datos para definir la arquitectura, diseño y tecnología empleada en el prototipo.

En el tercer capítulo en base a la información anterior se desarrolla el prototipo y se obtiene los resultados de las pruebas generadas por este.

En el capítulo final se detalla las conclusiones y recomendaciones del desarrollo del prototipo de aplicación Android.

PALABRAS CLAVE: Android, React Native, Firebase, finanzas personales

ABSTRACT

The present work consists of the development of an Android application prototype for the management of personal finances using the React Native framework together with the services provided by Firebase, which allows the user to have a record of their consumption in a semi-automatic way to help avoid a situation of over-indebtedness, this by perform the reading of emails issued by the bank after consumption and when the user wishes to register one manually, the application will organize said information, giving the user a detail organized by type of consumption and date.

The present work is detailed in four chapters, where general aspects, description of tools used and constant applied technology in chapter one

The second chapter has the data collection to define the architecture, design and technology used in the prototype.

In the third chapter, based on the previous information, the prototype is developed and the results of the tests produced by it are obtained.

The final chapter details the conclusions and recommendations of the development of the Android application prototype.

KEYWORDS: Android, React Native, Firebase, personal finances

1. INTRODUCCIÓN

Debido al avance de la tecnología y a la popularidad que han ganado los dispositivos móviles, se ha incrementado el número de personas conectadas a internet, siendo desde finales del año 2018 la mayoría respecto a aquellas que no tienen acceso a la red, de acuerdo a la Unión Internacional de Telecomunicaciones [1].

El uso de internet ha generado una evolución en todas las industrias, siendo las entidades financieras un claro ejemplo; actualmente un número importante de usuarios utilizan métodos alternativos al efectivo como método de pago, ya sea transferencias o tarjetas bancarias débito/crédito.

El uso cotidiano de dichas tarjetas ha provocado en los clientes una pérdida de seguimiento de sus consumos, donde la capacidad de endeudamiento no es clara creando un problema a fin de mes. Las entidades financieras suelen enviar notificaciones en forma de mensaje de texto o correo electrónico al momento que se realiza un consumo cualquiera, siendo esta última la más común, pero no suficiente para que los clientes tengan claro sus gastos [2].

Por tal razón se propone una aplicación móvil con el fin de manejar las finanzas del usuario de forma semiautomática.

El prototipo aprovecha la información brindada por la entidad bancaria a través de los correos enviados luego de realizar los consumos con tarjeta, filtra el remitente y toma los datos relevantes del correo recibido para brindarle al usuario los consumos mejor organizados, en el caso de que el consumo lo genere el cliente en efectivo el prototipo permite ingresarlo, actualizarlo o eliminarlo si así lo cree pertinente para tener un registro aún más completo, con el fin de tener los consumos realizados de forma organizada.

El prototipo inicialmente presenta al cliente los consumos realizados y cuanto es el gasto total mensual, como dichos consumos tienen asignados una categoría también se presenta una gráfica informativa de los consumos por categorías (alimentación, servicios básicos, salud, entretenimiento, transporte y varios), con el fin de que el usuario tenga presente en cuál de estas se está generando mayor consumo respecto al resto, finalmente en caso de que el usuario desee tomar control y evitar gastar más de un valor “x” en cierta categoría puede activar un mensaje de alerta, en caso de que sea sobrepasado ese valor referencial, se presenta un mensaje indicando aquello para que el cliente tome acción en ese aspecto y así evitar a fin de mes encontrarse en números rojos.

1.1 OBJETIVOS

El objetivo general de este Proyecto Técnico es: Desarrollar un prototipo de aplicación Android para el manejo de finanzas personales

Los objetivos específicos del Proyecto Técnico son:

- Analizar el uso del Framework React Native, sus herramientas y lenguaje de programación para el desarrollo de la aplicación móvil sobre el sistema operativo Android.

- Diseñar la aplicación móvil para que cumpla con las características planteadas.

- Implementar las funciones, módulos y características de acuerdo al diseño realizado previamente.

- Analizar los resultados de las pruebas realizadas.

1.2 ALCANCE

Este trabajo de integración se presenta el prototipo de una aplicación móvil para el sistema operativo Android 6 o superior. Dicha aplicación tiene el siguiente funcionamiento:

El cliente se autenticará con su correo electrónico de Gmail mismo que debe estar registrado en su entidad bancaria y automáticamente se creará un perfil con los datos de este y se almacenará en la nube, por políticas de seguridad Google no permite la lectura de correos en Gmail para desarrolladores, por lo que el cliente posterior a su autenticación, debe configurar desde su computadora en Gmail el reenvío de correos hacia uno general, para posteriormente ser manejado en el servidor de correos de Zoho mismo que facilita su lectura.

En caso de que el cliente reciba un correo por consumo y este configurado el reenvío antes mencionado, Zoho hace uso de Webhooks para enviar hacia un endpoint (url) del proyecto el correo previamente filtrado esto en formato JSON.

Con el correo recibido y el uso de funciones cargadas en la nube se extraerá la información relevante como entidad bancaria, fecha, valor total y beneficiario o lugar de consumo, en base a los datos recolectados se asignará una categoría al consumo y se almacenará en la base de datos de forma automática, de no existir una categoría directa se la guardará

como “Varios” donde el cliente podrá modificarlo manualmente de ser necesario, el cliente también podrá agregar, actualizar o eliminar consumos de forma manual.

Una vez con todos los datos necesarios ya obtenidos los consumos serán visibles al usuario en la pantalla inicial dando un resumen de este, así como también el valor del gasto total mensual, también se contará con una gráfica informativa de dichos consumos generada por categorías como, por ejemplo: alimentación, entreteniendo, salud entre otros. Finalmente, en el prototipo el cliente puede activar un mensaje en la aplicación que sirva como alerta de cuando haya superado un valor “x” ingresado por el usuario, en una categoría seleccionada.

La aplicación constará con un módulo de Login en el cual el usuario se autenticará con la cuenta de Gmail donde recibirá los correos de la entidad bancaria y de donde se procederá a extraer la información.

Otro modulo es del de Home, en donde se presentan los consumos registrados, el consumo total mensual de los mismos, también se encuentra la opción de añadir, actualizar o eliminar consumos de forma manual y se puede acceder a los mensajes que sirven como alertas de consumo en la categoría solicitada.

En el módulo Estadísticas, el usuario puede visualizar los consumos realizados por categorías del mes, representado en una gráfica informativa.

Dentro del módulo Alertas, el cliente puede ingresar un valor referencial máximo en alguna categoría para tener presente si llega a superarlo, ya que de ocurrir se le presentara un mensaje en la interfaz de notificaciones, esto permite tener aún más presente la información importante para el cliente.

Finalmente, el módulo Perfil muestra la información del usuario registrado con su cuenta de Gmail.

En el presente trabajo de titulación no se pretende exportar datos ni archivos en ningún formato, tampoco alterará información del correo electrónico involucrado, solamente se tomarán los datos de los consumos generados por el usuario, y dicho análisis se hará para una entidad bancaria, el prototipo no generará ninguna notificación push. La distribución de actividades del presente proyecto seguirá la metodología ágil Kanban.

El presente trabajo de titulación constará con un producto final demostrable

1.3 MARCO TEÓRICO

1.3.1 REACT

Es una librería JavaScript de código abierto, declarativo y basado en componentes, para el desarrollo de interfaces de usuario, dichas interfaces son divididas en piezas pequeñas denominadas componentes, que en conjunto pueden formar estructuras complejas, que conservan su estado y permiten que el código sea reutilizable y mantenible con el tiempo.

React utiliza programación reactiva, esto conlleva a que se renderiza el código cada que cambia el estado del componente o recibe nuevas propiedades, este proceso con React se optimiza por el concepto de DOM virtual, mismo que al detectar dichos cambios de estado y propiedades hace una comparación del estado anterior del DOM y almacena solamente dicho cambio [1].

1.3.2 REACT NATIVE

Es un framework para el desarrollo de aplicaciones móviles tanto para Android como para iOS, basado en JavaScript y utilizando las herramientas de React, dichas aplicaciones pueden interactuar con módulos nativos del dispositivo, teniendo una experiencia de usuario prácticamente nativa.

El funcionamiento de React Native se basa en tres capas, capa nativa, capa bridge y capa JavaScript como se muestra en la Figura 1.1.

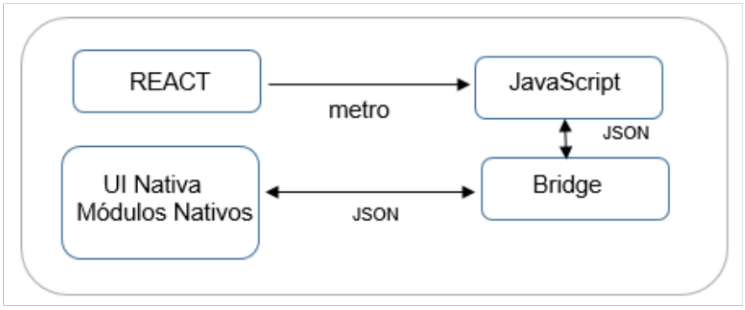


Figura 1.1 Arquitectura React Native

La capa nativa se encuentra más cercana al dispositivo y maneja los elementos de la interfaz de usuario, mientras que la capa bridge, es un puente de comunicación bidireccional que transporta datos por lotes serializados y asíncronos desde la capa

JavaScript a la capa nativa. La capa JavaScript es la que ejecuta el código con la lógica del aplicativo una vez se haya recibido la carga útil procedente del brigde. El proceso de ejecución se puede observar en la Figura 1.2 [2].

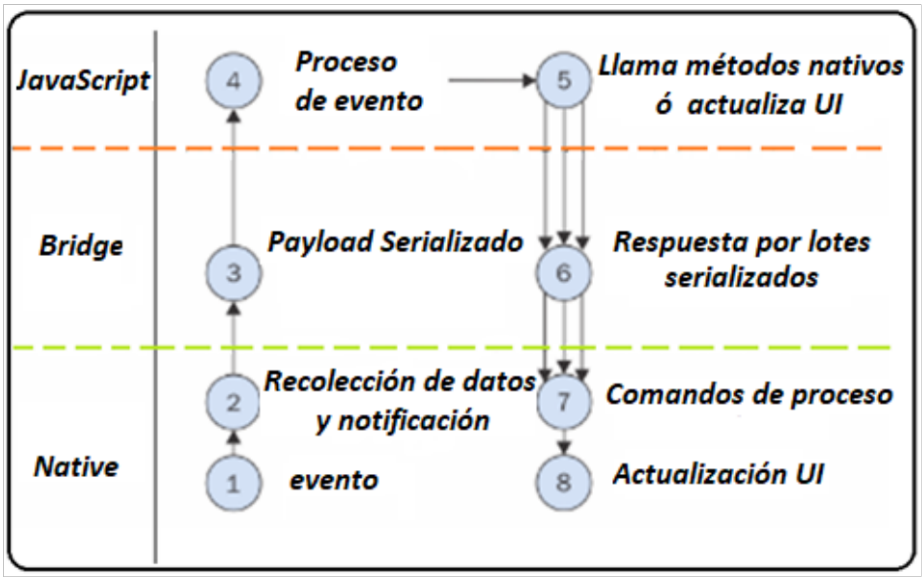


Figura 1.2 Proceso de ejecución de eventos entre capas [2].

1.3.3 REDUX

Es una librería para controlar el estado de aplicaciones JavaScript, pudiéndose adaptar a frameworks basados en dicho lenguaje como React native, Redux intenta hacer predecibles los cambios de estado, cumpliendo ciertas restricciones las cuales se reflejan en tres principios [3].

Una sola fuente de la verdad: El estado se almacena en un solo store, permitiendo que pueda ser serializado y enviado al cliente sin necesidad de pasos adicionales, también resulta más fácil depurar la aplicación ya que solo hay un árbol de estado [4].

En la Figura 1.3 se observa una arquitectura de componentes con Redux en la imagen (a) y sin Redux, imagen (b), en esta última se presenta un solo store que alimenta a todos los componentes para cualquier cambio que requiere una acción.

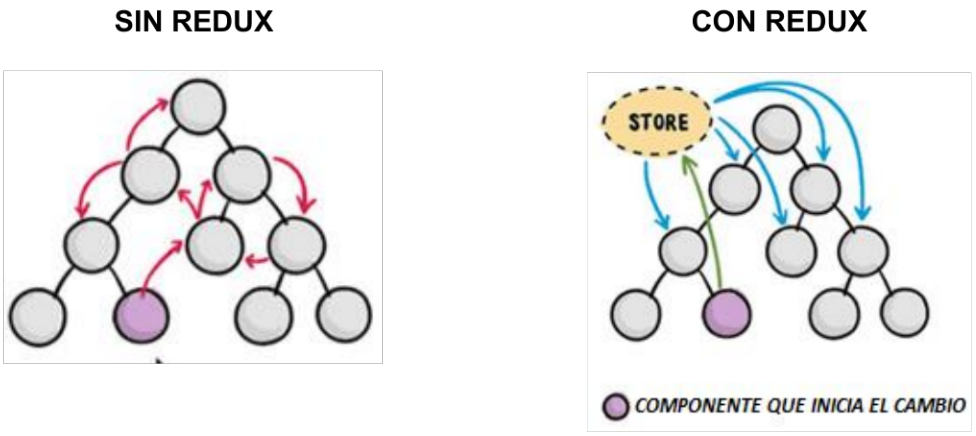


Figura 1.3 Arquitectura de componentes sin Redux (a) y con Redux (b) [4].

El estado es de solo lectura: Se puede modificar el estado solo si se emite una acción describiendo que cambió, esto garantiza que no se altere el estado de la aplicación a causa de otros eventos tales como callbacks o sockets [4].

Los cambios se realizan con funciones puras: Los Reducers reciben el estado actual y la acción que se realizará emitiendo como respuesta un nuevo estado que no altera al anterior [4].

En la Figura 1.4 se muestra el diagrama de flujo de Redux, donde un evento causa una acción, que se almacena en el árbol de estado (store) y devuelve un nuevo valor para actualizar la interfaz de usuario si es requerido [3].

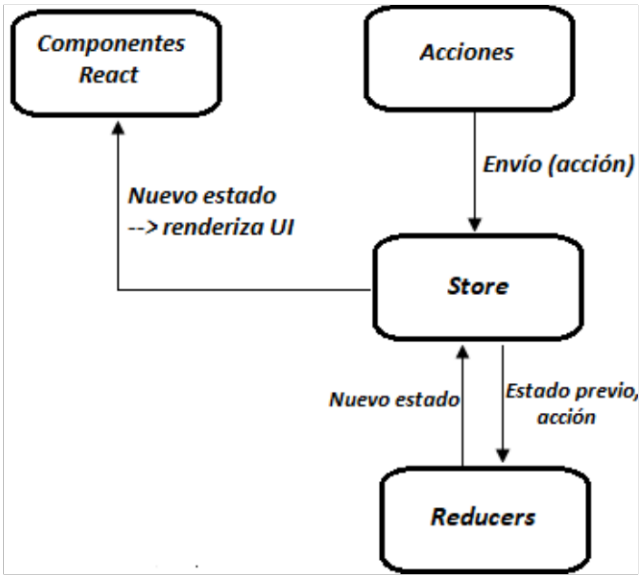


Figura 1.4 Estructura de Redux [3].

1.3.4 FIREBASE

Firebase es un Backend como servicio (Baas) usado en el desarrollo web y aplicaciones móviles con el respaldo de Google, es una herramienta que permite el desarrollo multiplataforma, en donde se puede incluso iniciar el desarrollo del proyecto de forma gratuita, al ser un servicio backend permite que su crecimiento sea más rápido, brindando al desarrollador más tiempo para elaborar la parte Fronted del mismo [5].

Firebase maneja JSON como formato principal para el intercambio de datos, dicho formato es independiente del lenguaje de programación utilizado y consta de dos estructuras, colecciones nombre/valor o también conocido como objetos y arreglos de valores. El Código 1.1 presenta un ejemplo del formato JSON, en donde se puede apreciar dos objetos definidos desde la línea 2 a la 13 el primero y el segundo desde la 14 a la 25, mientras que los valores son los que se encuentran posterior a los dos puntos sean texto, booleanos o números.

```
1  const users = [
2    {
3      id: 'id1',
4      name: 'Cristian R',
5      email: 'card@example.com',
6      profileImage: 'https://examp1.com/imageprofile/1.png',
7      validation: false,
8      cursos: [
9        "HTML",
10       "CSS",
11       "JS"
12     ]
13   },
14   {
15     id: 'id2',
16     name: 'Karen Mena',
17     email: 'karen@example.com',
18     profileImage: 'https://examp1.com/imageprofile/2.png',
19     validation: false,
20     cursos: [
21       "React",
22       "Redux",
23       "Cloud"
24     ]
25   },
26 ];
```

Código 1.1 Ejemplo formato JSON [5].

Firebase al ser un conjunto de herramientas en la nube ofrece varias ventajas, entre las más destacadas se tiene: Rapidez de desarrollo, monitoreo de errores, seguridad y fácil escalamiento [6].

La Figura 1.5 presenta los diversos servicios de Firebase

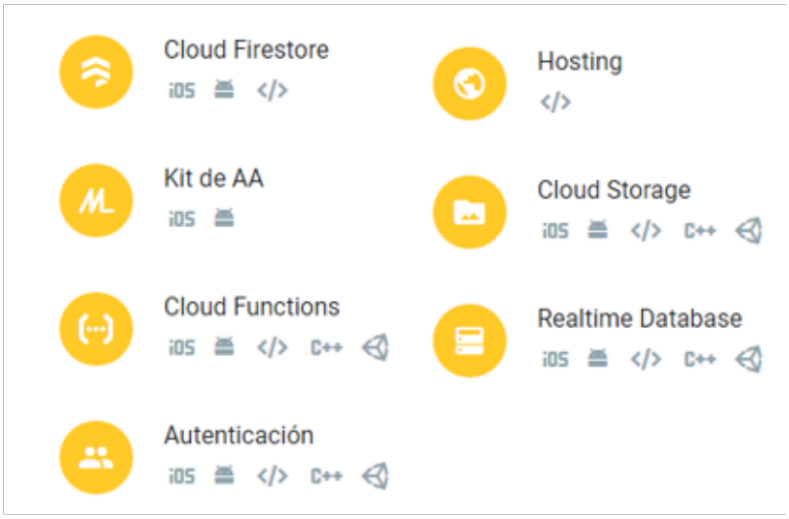


Figura 1.5 Servicios Firebase [6].

- Cloud Firestore

Es una base de datos NoSQL, flexible y escalable que ofrece soporte sin conexión y mantiene sus datos sincronizados entre aplicaciones del cliente. Además, brinda una integración sin interrupciones con el resto de productos de Firebase [7].

- Kit de AA

Es un conjunto de herramientas y servicios creado con el objetivo de proporcionar aprendizaje automático, integrado con un conjunto de API disponibles en la nube, se enfocan en casos de usos habituales como reconocimiento de textos, etiquetas de imágenes y puntos de referencia. El aprendizaje automático de Firebase permite integrar modelos de aprendizajes previamente generados de ser necesarios.

- Cloud Functions

Es un framework sin servidores que permite ejecutar automáticamente el código de backend en respuesta a solicitudes HTTPS. El código en JavaScript o TypeScript almacenado en la nube se ejecuta en entorno administrativo [8].

- Autenticación

El objetivo de este servicio es facilitar el sistema de autenticación de usuarios, proporcionando servicios de backend SDK y bibliotecas UI fáciles de utilizar, es compatible con los proveedores mostrados en la Figura 1.6 [9].

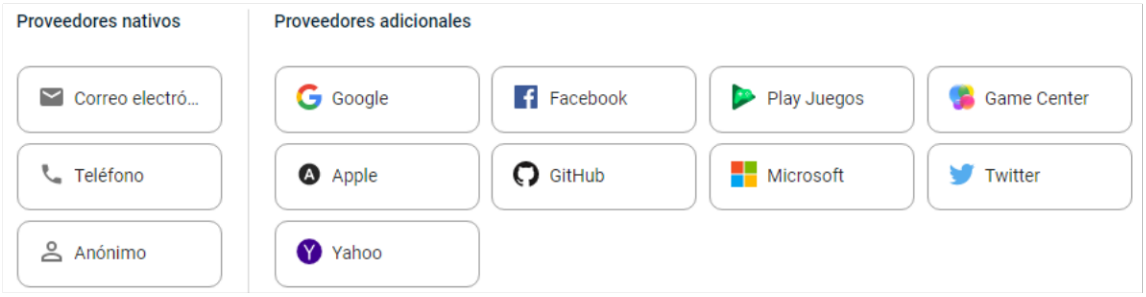


Figura 1.6 Proveedores de Autenticación

- Firebase Hosting

Es un servicio de hosting seguro y confiable para aplicaciones web, proporciona y configura automáticamente un certificado SSL para cada sitio que se implementa.

- Cloud Storage

Este servicio permite a las aplicaciones subir archivos a la nube y poder compartirlo con otros usuarios realizando conexiones seguras.

- Real Time Database

Es una base de datos NoSQL alojada en la nube, los datos almacenados se sincronizan en tiempo real y se encuentran disponibles aun sin tener conexión.

1.3.5 WEBHOOKS

Un webhook es una herramienta que permite notificar un evento generado desde un sistema o aplicación hacia otro, dichas notificaciones son devoluciones de llamada HTTP “POST” en formato JSON hacia una URL de destino, configurada como punto final del webhook. En el Código 1.2 se presenta un ejemplo de respuesta de un webhook teniendo a Usaurio1 como emisor y Usuario2 como receptor, en la figura mencionada se presenta desde la línea 1 a la 13 los identificadores del correo notificado como respuesta del webhook y de la línea 14 en adelante la estructura de este en formato HTML [10].

```
{
     "summary": " Texto del mensaje",
     "sentDateInGMT" : 1560866021000 ,
     "subject": " Ejemplo: ejemplo de respuesta webhook ",
     "messageId": 1560840837125110000 ,
     "toAddress": " \" Usuario2 \ "<carodriguez@ejemploig.com> ",
     "folderId": 3881227000000013000 ,
     "zuid": 647772765 ,
     "ccAddress ":" ",
     " tamaño ": 55503 ,
     "sender": " Usuario1 ",
     " selectedTime ": 1560840837126 ,
     "fromAddress": " usuario1@ejemploig.com ",
     "html": "<meta /> <div> <div style = \ "font-family: & quot;
Trebuchet ms & quot ;, Arial, Helvetica, sans-serif; font-size:
12pt; \"> <div> Texto1, <br /> </div> <div> <br /> </div> <div>
Cuerpo del mensaje. <br /> </div> <div id = \ "\"> <div> <img src =
\ "/ zm / ImageDisplay? F = 1. png & amp; mode = inline & amp; cid
=0.28869215260.3894179596053002321.16b695cdb49__inline__img__src &
amp; \ "width = \" 145 \ "height = \" 145 \ "style = \" float:
left; \ "/> <br /> </div> < div> <br /> </div> <div> <br /> </div>
<div> Texto2, <br /> </div> <div> Remitente <br />"
}
```

Código 1.2 Ejemplo de respuesta de un webhook [11].

A diferencia de una API un webhook gestiona notificaciones de forma directa cuando ocurre el evento sin necesidad que el receptor realice consultas reiteradas, esto se puede observar con mejor detalle en la Figura 1.7 teniendo en la imagen (a) el funcionamiento de una API y en la imagen (b) la de un webhook.

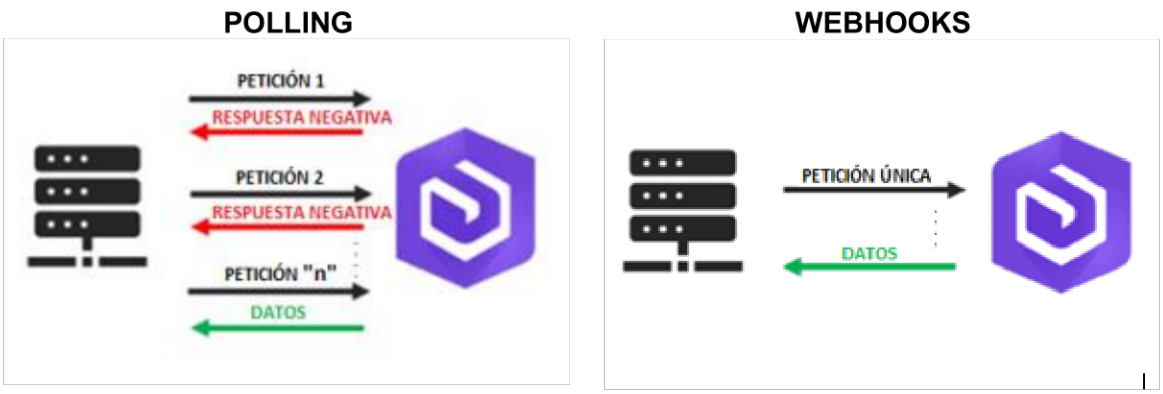


Figura 1.7 Comparación del funcionamiento de una API (a) vs Webhooks (b) [10].

1.3.6 HERRAMIENTAS

1.3.6.1 NODE

Es un entorno de ejecución en el lado del servidor basado en JavaScript y que utiliza el motor V8 de Google. Node utiliza el empaquetador metro para brindar servicio a aplicaciones React native, esta toma un archivo de entrada y varias opciones para devolver un solo archivo JavaScript con el código y dependencias [12].

La Figura 1.8 indica los sistemas operativos en los cuales se puede instalar node desde la página oficial.

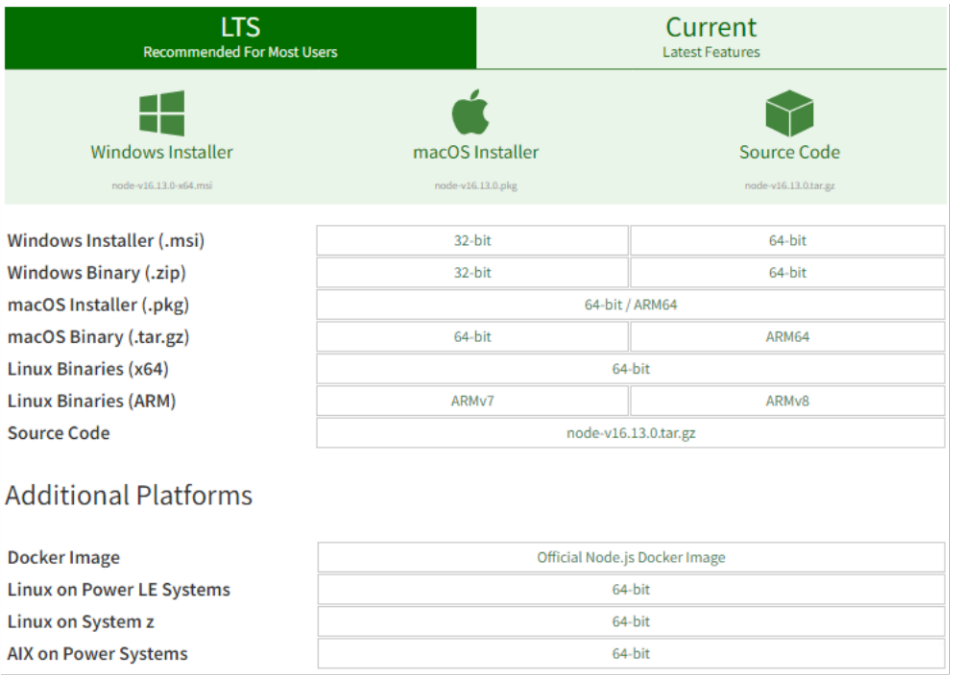


Figura 1.8 Sistemas Operativos en los cuales node está disponible [13].

Node cuenta con un administrador de paquetes llamado npm, mismo que se encarga de gestionar las descargas del proyecto basándose en el archivo “package.json” en donde se encuentran todas las dependencias así como de nuevas librerías de ser necesarias.

1.3.6.2 REACT NATIVE CLI

React Native CLI es la herramienta que interactúa con los proyectos permitiendo crearlos, enlazar dependencias y ejecutar las aplicaciones. Para crear un proyecto JavaScript se hace uso del comando “react-native init MiProyecto” en caso de utilizar TypeScript es “npx react-native init MiProyecto --template react-native-template-typescript” [14].

La Figura 1.9 muestra la estructura de un proyecto React Native con TypeScript

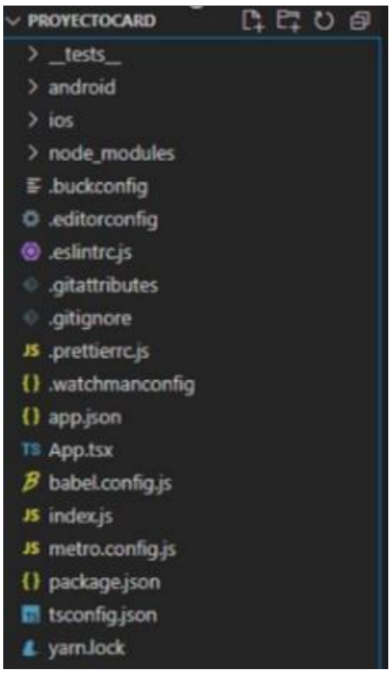


Figura 1.9 Estructura Proyecto React Native

1.3.6.3 FIREBASE CONSOLE

La consola de Firebase permite crear, administrar y agregar servicios a los proyectos desde su página web oficial https://console.firebase.google.com/?hl=es. En la Figura 1.10 se presenta la interfaz en donde se puede agregar un proyecto nuevo fácilmente.

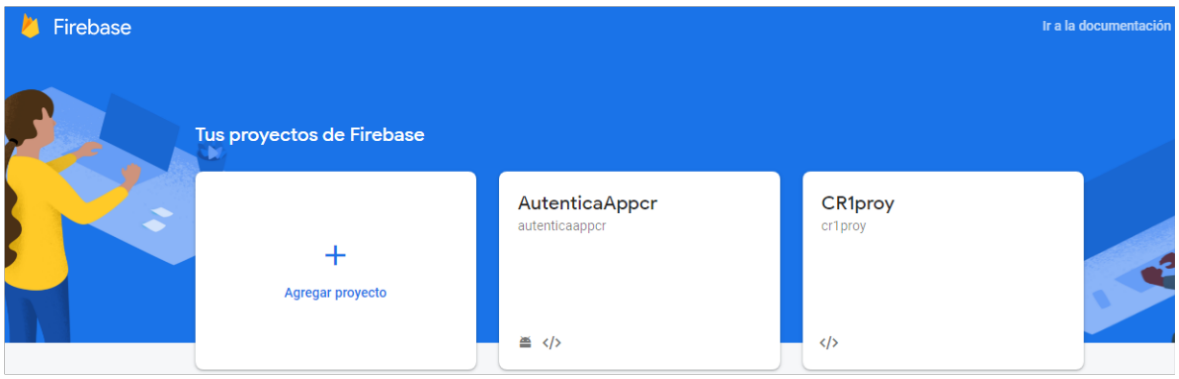


Figura 1.10 Interfaz Firebase Console [8].

Una vez creado el proyecto y haber registrado la aplicación, tendremos el código base generado automáticamente por Firebase, mismo que se representa en el Código 1.3 donde la línea 1 se importa Firebase, y las líneas del 2 al 9 se tiene las credenciales y llaves que

identifican el proyecto creado para hacer uso en la aplicación en desarrollo, previamente se debió haber instalado Firebase en el proyecto de React Native bajo el comando “npm install firebase".

```
1   import { initializeApp } from "firebase/app";

2   const firebaseConfig = {
3     apiKey: "AIzaSyBl6yKGwjQXxtDxC64_viU1SBr9SNAntKg",
4     authDomain: "democard-e3b41.firebaseapp.com",
5     projectId: "democard-e3b41",
6     storageBucket: "democard-e3b41.appspot.com",
7     messagingSenderId: "117404102176",
8     appId: "1:117404102176:web:b7aca1cf76c8b097e1d645"
9   };
10
11  const app = initializeApp(firebaseConfig);
```

Código 1.3 Identificadores para registro de la aplicación.

Finalmente se tendrá, disponible todos los servicios que ofrece Firebase y que sean necesarios para las aplicaciones que estén registradas en dicho proyecto.

1.3.6.4 VISUAL STUDIO CODE

Es un editor de código liviano disponible para Windows, macOS y Linux, provee soporte para JavaScript, TypeScript y Node.js, también cuenta con gran cantidad de extensiones para otros lenguajes y personalización de su entorno de trabajo [15].

La Figura 1.11 Muestra una captura de pantalla del ambiente de trabajo.

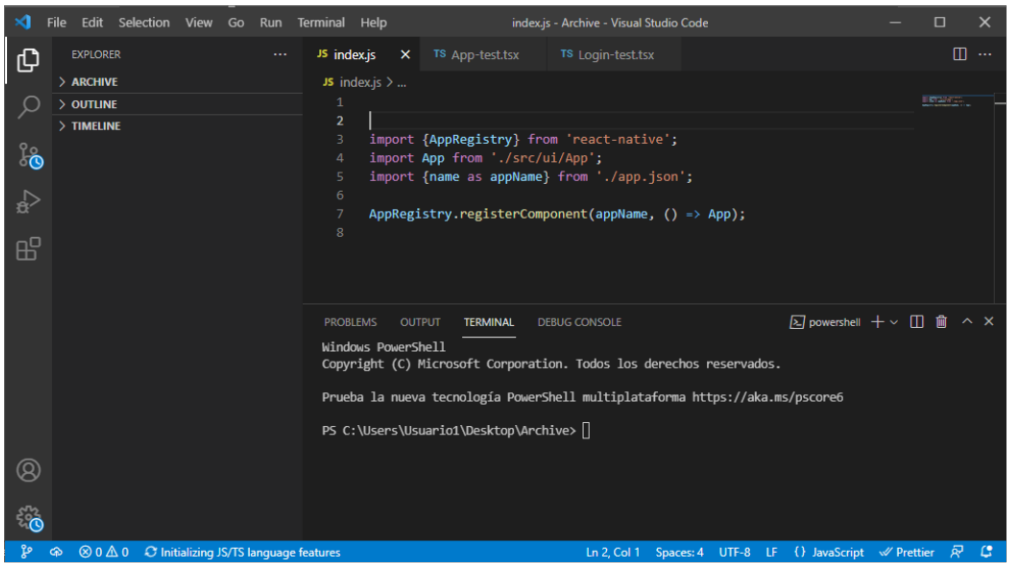


Figura 1.11 Ambiente de Visual Code [15].

1.3.6.5 ANDROID STUDIO

Android Studio es un entorno de desarrollo para aplicaciones Android, cuenta con varias herramientas para mejorar la productividad entre las principales se pueden nombrar: Integración con GitHub y plantillas de código en ejecución, Identificar problemas de rendimiento con uso de Lint, compatibilidad con C++ NDK, compatibilidad con Google Cloud Platform y un emulador AVD que se puede modificar sus características, en el caso del uso en conjunto con aplicaciones React native se requiere el SDK Android 10 (Q) [16].

En la Figura 1.12 se observa un AVD de Android Studio

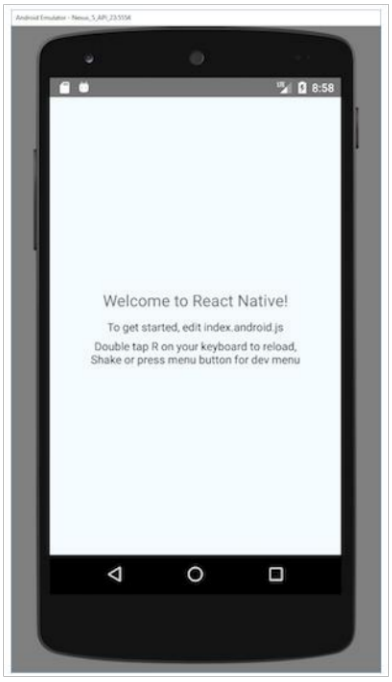


Figura 1.12 AVD de Android Studio [17].

1.3.6.6 ZOHO MAIL

Es una plataforma de correo empresarial, confiable y segura misma que está pensada para la comunicación de organizaciones, puede interactuar directamente con otras apps disponibles de Zoho como: Zoho CRM, Creator, Project y Docs. Para el uso de Zoho Mail con aplicaciones de terceros se debe hacer uso de Webhooks, uno entrante y uno saliente dependiendo de cuál sea el objetivo. Un Webhook entrante permite publicar en un URL elegido cuando se genera un evento puntual en la aplicación por terceros. Un Webhooks de salida permite configurar los correos electrónicos para que estos provoquen la activación de eventos en aplicaciones por terceros, con la herramienta de filtro disponible por Zoho Mail se puede escoger que correos detonarían dicha acción un ejemplo puede ser correos emitidos por una entidad bancaria [11].

2. METODOLOGÍA

En el presente capítulo se describe tanto el diseño como el desarrollo utilizado para la elaboración de la aplicación móvil, con el objetivo de crear una guía que facilite la información. El método de desarrollo ágil utilizado es Kanban, mismo que detalla las tareas a realizar en cada etapa de desarrollo.

Con el fin de obtener los requerimientos se generaron encuestas mismas que se realizaron a 10 personas, en base a esto se crearon las historias de usuario, las interfaces de usuario y la funcionalidad de la aplicación móvil.

2.1 TABLERO KANBAN

Las tareas a realizar para el desarrollo de la aplicación Android propuesta, se detallan en el tablero Kanban el cual se presenta en la tabla 2.1, mismas que encuentran enlistadas en orden con el fin que la construcción del proyecto se concrete en el menor tiempo posible y con los mínimos inconvenientes posibles.

Tabla 2.1. Tareas Tablero Kanban

	TAREAS TABLERO KANBAN
1	Analizar las respuestas obtenidas de las encuestas
2	Toma de requerimientos
3	Diseñar las interfaces de usuario
4	Realizar el diagrama de casos de uso
5	Realizar el diagrama de estructura
6	Diseñar la base de datos
7	Instalación de Visual Studio Code
8	Instalación de Node js
9	Instalación de Yarn
10	Instalación de React Native CLI
11	Crear Base de Datos en Firebase
12	Instalación de Firebase en la aplicación móvil
13	Configurar Zoho mail
14	Codificar Interfaces de usuario
15	Codificar funcionalidad
16	Pruebas de funcionamiento
17	Resultados

2.2 ENCUESTAS

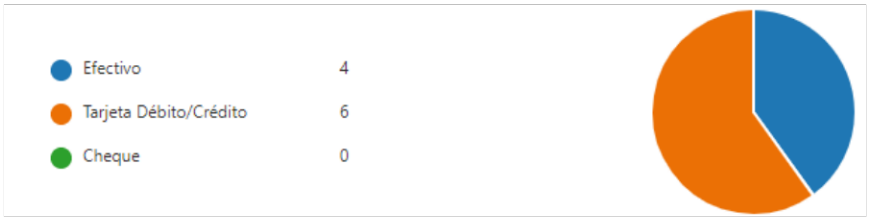
Con el fin de obtener las historias de usuario, los requerimientos funcionales y no funcionales se realizó una encuesta a diez personas tendiendo los siguientes resultados:

En la Figura 2.1 se presenta el resultado de las preguntas 1 y 2 en las cuales se puede observar que todos los encuestados manejan tarjeta de débito/crédito y que en su mayoría aseguran que este método de pago es que el que más utilizan.

Pregunta 1. ¿Cuenta usted con Tarjeta de Débito/Crédito?

(a)

Pregunta 2. ¿Cuál es el método de pago que más utiliza?



(b)

Figura 2.1 Resultados preguntas 1 (a) y pregunta 2 (b)

Con los resultados de la pregunta 3 mismas que se encuentran presentes en la Figura 2.2, se busca validar cual es el método de pago considerado más seguro dentro de los encuestados, siendo elegido el de tarjetas débito/crédito dando pie a que en el futuro cercano y mediano no se descontinuaría el uso de las mismas por cuestiones de seguridad.

Pregunta 3. ¿Qué método de pago cree usted que es más seguro?

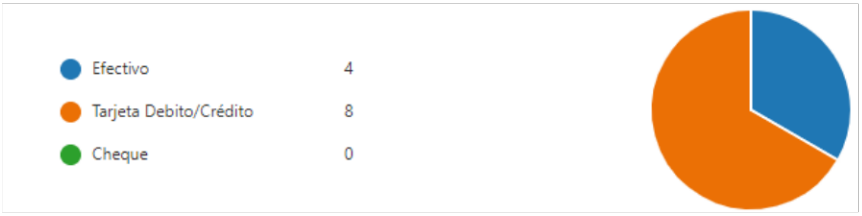


Figura 2.2 Resultados pregunta 3

La Figura 2.3 presenta los resultados de la pregunta 4 donde se puede observar que la mayoría de encuestados no está totalmente al pendiente de sus correos electrónicos que reciben al momento de hacer consumos.

Pregunta 4. ¿Cuán seguido usted revisa su correo electrónico por consumos realizado?

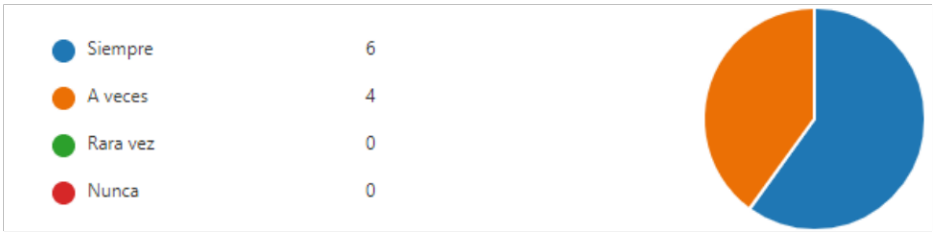
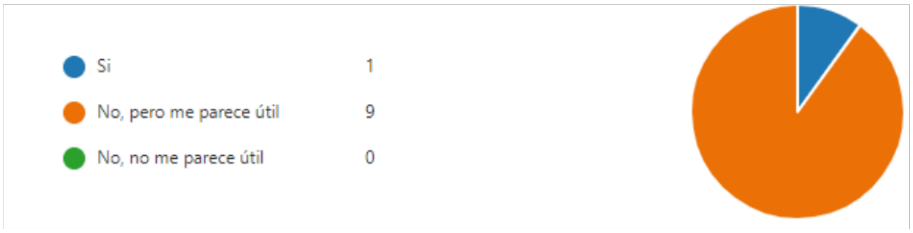


Figura 2.3 Resultados pregunta 4

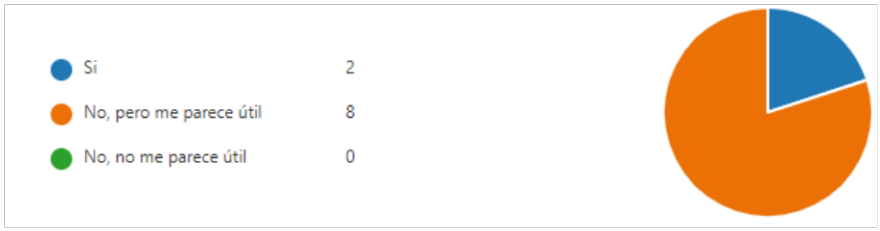
En los resultados de las preguntas 5 y 6 presentes en la Figura 2.4 se puede observar que la mayoría de usuarios no está consciente de cuanto es su consumo por categoría, lo cual genera es un indicativo de mala administración de finanzas personales mismo que se intenta mejorar con el prototipo propuesto.

Pregunta 5. ¿Conoce usted cuanto gasta mensualmente en comida?



(a)

Pregunta 6. ¿Conoce usted cuanto gasta mensualmente en vestimenta?



(b)

Figura 2.4 Resultados preguntas 5 (a) y pregunta 6 (b)

La Figura 2.5 que presenta el resultado de la pregunta 7 pretende conocer si el cliente le interesa un sistema de administración de sus consumos ordenado por categorías, en su totalidad de los encuestados les atrae la propuesta planteada.

Pregunta 7. ¿Le gustaría contar con información ordenada en categorías de sus consumos?

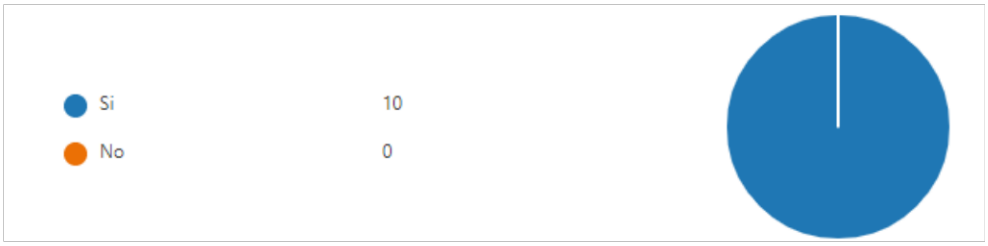
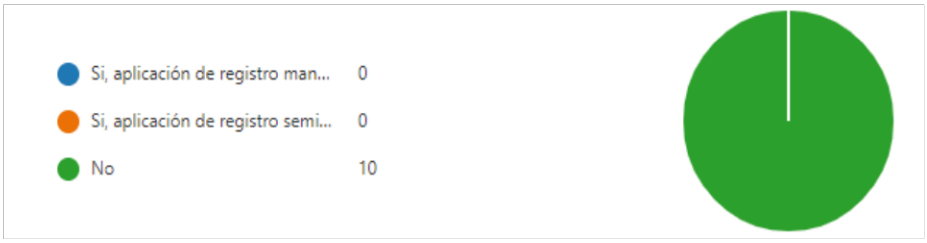


Figura 2.5 Resultados pregunta 7

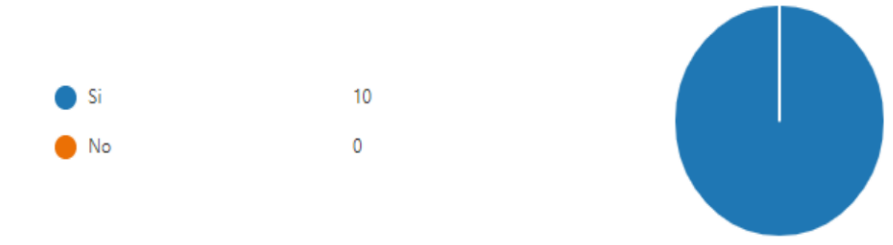
Finalmente con las preguntas 8 y 9 se pretende conocer si los encuestados estarian dispuesto a probar una aplicación que les brinde informacion acerca de sus consumos lo cual ayudaria con el manejo de sus finanzas personales para estar más concientes sobre su capacidad de endeudamiento, donde los resultados brindado muestran que si les interesa dicha aplicación. La Figura 2. 6 muestra las respuestas de las preguntas 8 (a) y de la pregunta 9 (b)

Pregunta 8. ¿Cuenta usted con una aplicación móvil para administrar sus finanzas personales?



(a)

Pregunta 9. ¿Usted usaría una aplicación móvil para organizar la información de sus consumos, esto basado en los correos electrónicos emitidos posteriormente?



(b)

Figura 2.6 Resultados preguntas 8 (a) y pregunta 9 (b).

2.3 HISTORIAS DE USUARIO

En base a las encuestas y de los resultados obtenidos se puede deducir las historias de usuario mismas que se presentan en la Tabla 2.2 que servirán de guía para el desarrollo del prototipo, definir funcionalidades y módulos a emplear.

Tabla 2.2. Historias de usuario

H.U.	Título	Descripción
1	Registrar al usuario	Para poder hacer uso de la aplicación móvil el usuario debe autenticarse con su correo electrónico de Gmail.
2	Ingresar a la pantalla inicial del aplicativo	El cliente debe configurar por seguridad desde su Gmail el reenvío de correos.
3	Observar sus consumos	Cliente requiere obtener sus consumos con detalles.
4	Agregar, actualizar y eliminar consumos	El cliente puede agregar actualizar o eliminar consumos de forma manual en caso de que lo crea pertinente.
5	Mostrar Consumos por categoría	Cliente requiere conocer sus consumos por categorías.
6	Mostrar mensaje de alerta	Cliente puede activar un mensaje que sirva de alerta en caso de que sobrepase un valor deseado en cierta categoría.

2.3.1 REQUERIMIENTOS FUNCIONALES

- Autenticar al usuario mediante su correo de Gmail.
- Registrar al usuario con su información personal
- Reenviar los correos del cliente desde su correo Gmail
- Filtrar los correos del cliente
- Discriminar información requerida de los correos electrónicos.
- Obtener y almacenar en Firebase la información relevante
- Permitir al usuario agregar, actualizar o eliminar consumos de forma manual.
- Permitir al usuario conocer sus consumos por categorías.
- Permitir al usuario ingresar un valor “x” que sirva como activador de un mensaje de alerta cuando se lo haya sobrepasado en cierta categoría elegida.

2.3.2 REQUERIMIENTOS NO FUNCIONALES

- Desarrollar un prototipo de aplicación móvil Android haciendo uso del framework React Native
- Alojar la base de datos en Firebase
- Para actualizar los datos el usuario debe estar conectado a internet.
- El cliente debe tener registrado en su entidad bancaria el mismo correo Gmail con el que se registra en la aplicación.

2.4 ARQUITECTURA DEL PROTOTIPO

La arquitectura del prototipo hace uso de BaaS de Google Firebase para el desarrollo, se compone de una aplicación Android la cual utiliza como servicios backend los provistos por Firebase en la nube, entre los que hace uso es el de autenticación del cliente, Cloud Functions y almacenamiento de datos.

Para la lectura de los correos se utiliza Zoho Mail con Webhooks, debido a que Gmail por seguridad modifico sus reglamentos y no permite la lectura de mails, por lo que es necesario el reenvió de correos hacia Zoho Mail, mismo que permite filtrado y cuenta con una API que permite leerlos. En la Figura 2.7 se presenta la arquitectura del prototipo pudiéndose observar el ciclo del funcionamiento del mismo.

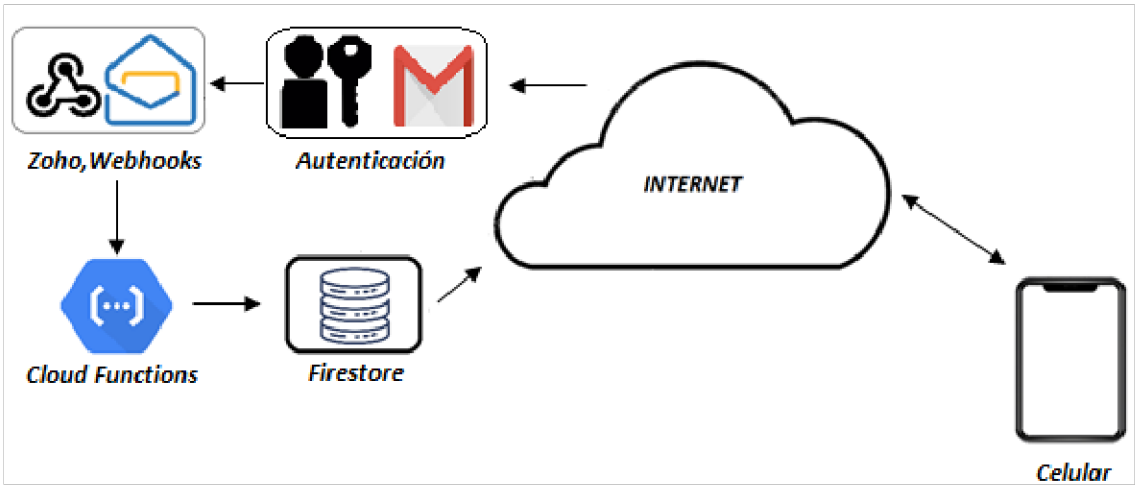


Figura 2.7 Arquitectura de prototipo

2.4.1 CASOS DE USO

En el presente proyecto se cuenta con los siguientes actores quienes interactuaran con el prototipo teniendo distintas opciones de elección. También se presenta una imagen representando cada situación.

- Usuario no registrado
- Usuario registrado sin confirmación de reenvío de correo
- Usuario registrado con confirmación de reenvío de correos

Para cada actor se representa su diagrama del caso de uso correspondiente, el usuario no registrado solo cuenta con la opción de autenticarse con su cuenta de Gmail, el diagrama se presenta en la Figura 2.8

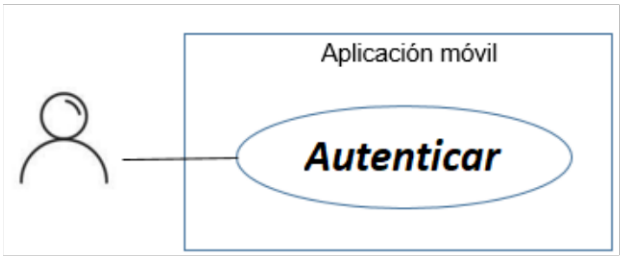


Figura 2.8 Caso de Uso, usuario sin registrar

El usuario registrado que no haya confirmado el reenvío de correos se le presentara una guía de cómo hacerlo y no avanzara hasta que el proceso finalice con éxito, en la Figura 2.9 se presenta su diagrama de caso de uso.

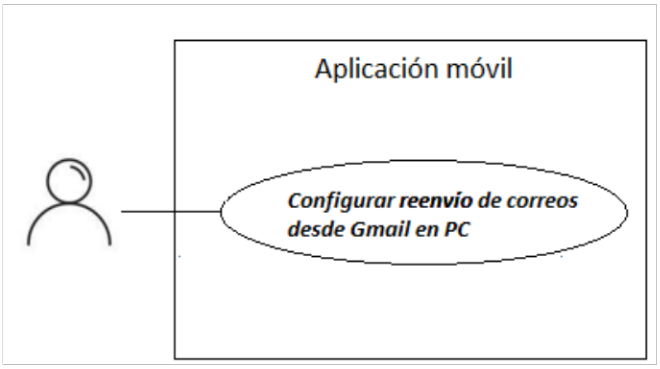


Figura 2.9 Usuario registrado sin confirmación de reenvío de correo

Una vez que el usuario haya finalizado con los pasos previos el cliente podrá ingresar a la pantalla principal, observar sus consumos realizados, agregar actualizar o eliminar consumos manualmente, observar consumos por categorías y finalmente añadir alertas, en la Figura 2.10 se presenta el diagrama de uso del cliente registrado y con configuración validada.

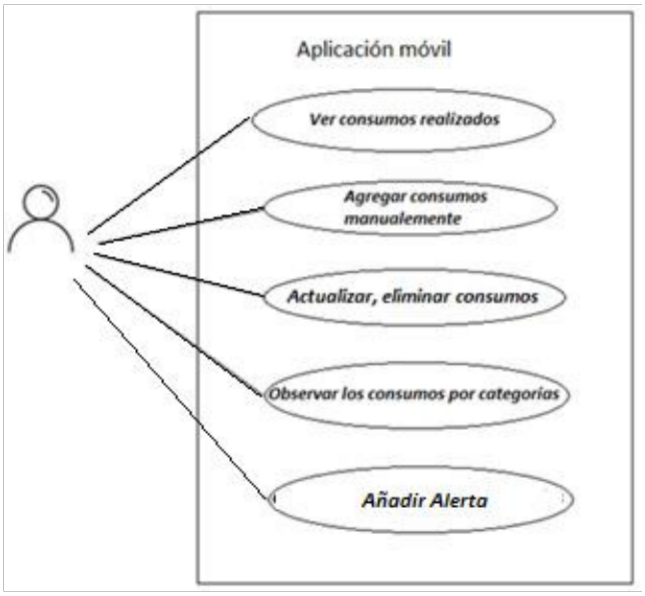
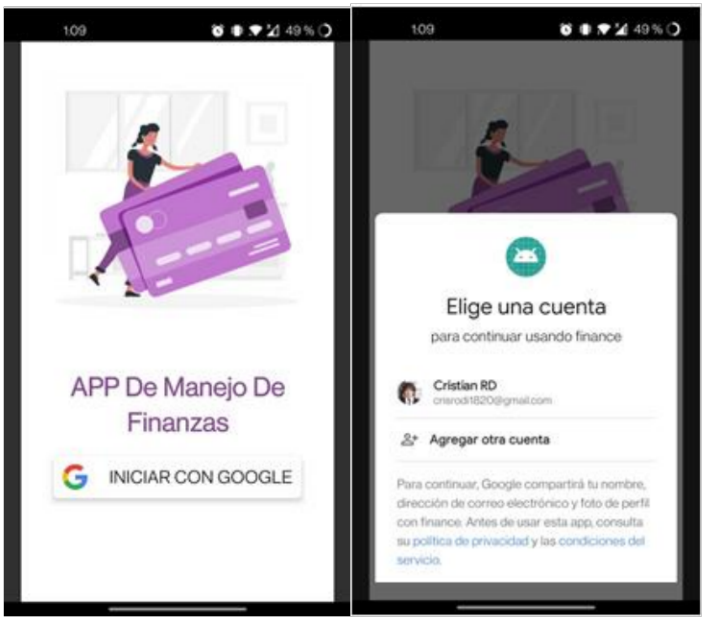


Figura 2.10 Usuario registrado con confirmación de reenvío de correos

2.5 DISEÑO DE LA INTERFAZ DEL PROTOTIPO

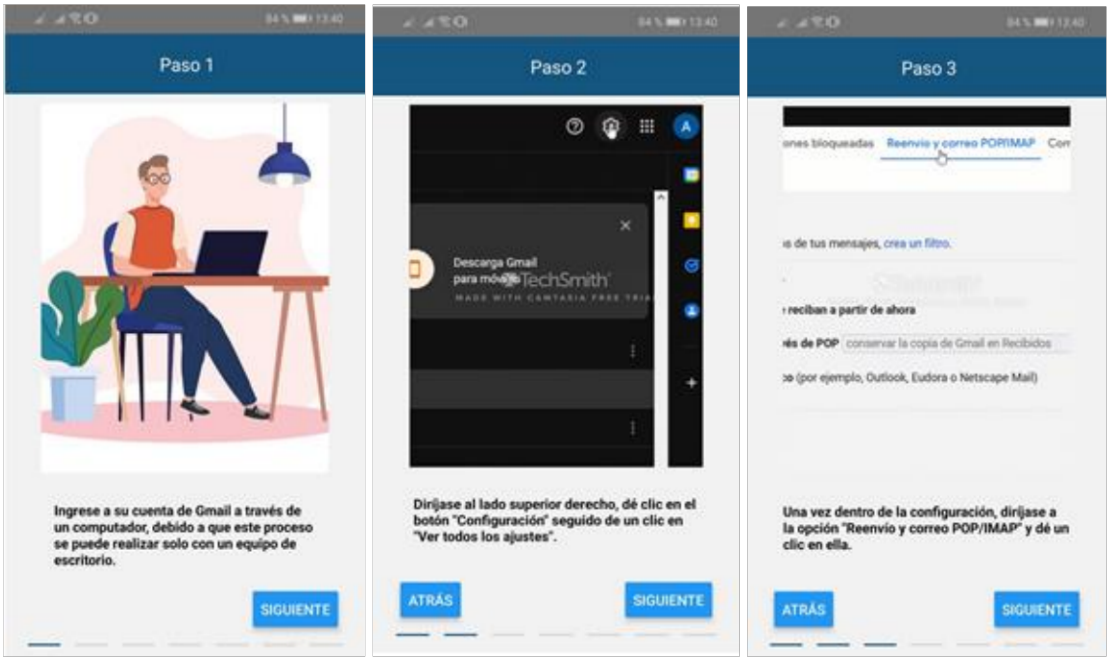
A continuación, se presenta las interfaces de usuario y detalle de su función, en base a los casos de uso y los requerimientos recolectados previamente, en la Figura 2.11 se observa una captura de la pantalla inicial (a) y la página de autenticación del usuario con su cuenta de Gmail (b).



(a) (b)

Figura 2.11 Interfaz de inicio (a) y autenticación de usuario (b)

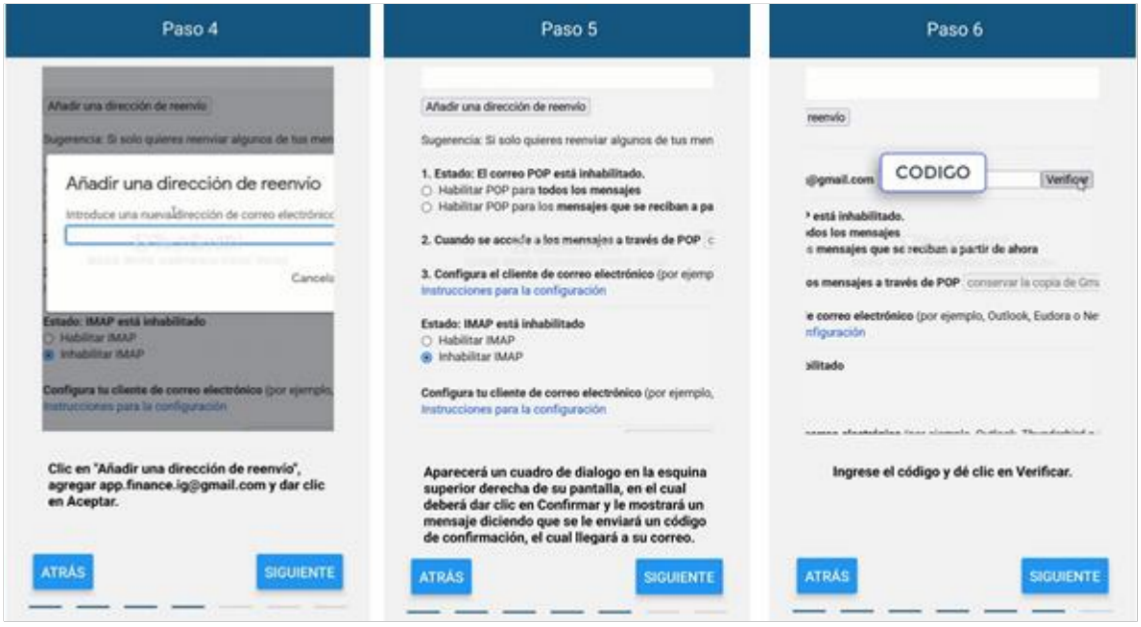
Una vez autenticado el cliente se presentan las interfaces siguientes, mismas que reflejan una guía y los pasos a seguir para la configuración del reenvío de correos hacia Zoho mail. En la Figura 2.12 se presentan los 3 primeros pasos a seguir en donde se debe acceder desde una computadora (a) ir a configuraciones acceder a todos los ajustes (b) e ir a la opción de reenvío y correo(c).



(a) (b) (c)

Figura 2.12 Interfaces de los 3 primeros pasos a seguir

En la Figura 2.13 se presentas los ultimos 3 pasos a seguir para finalizar con la configuracion desde Gmail, mismos que son el de añadir la cuenta de correo “app.finance.ig@gmail.com” (a), comprobar el correo ingresado y validar la recepcion del código necesario en la bandeja de entrada (b) y finalmente ingresari dicho codigo (c)



(a) (b) (c)

Figura 2.13 Interfaces de los últimos pasos a seguir.

Una vez completado los pasos a seguir finalmente pulsando en la opción “He Terminado” mostrándose captura de pantalla en la Figura 2.14 se comprobará si se ejecutó de forma correcta e ingresara a la aplicación caso contrario no avanzara teniéndose por parte del cliente revisar configuración.

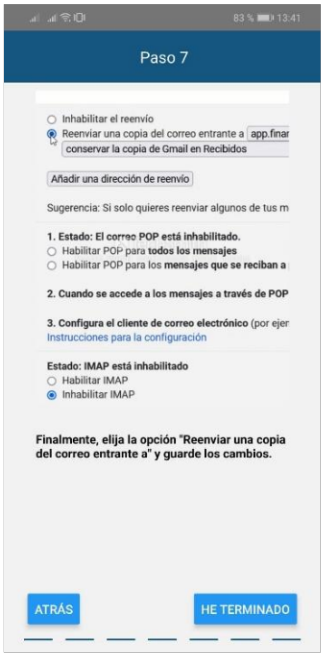
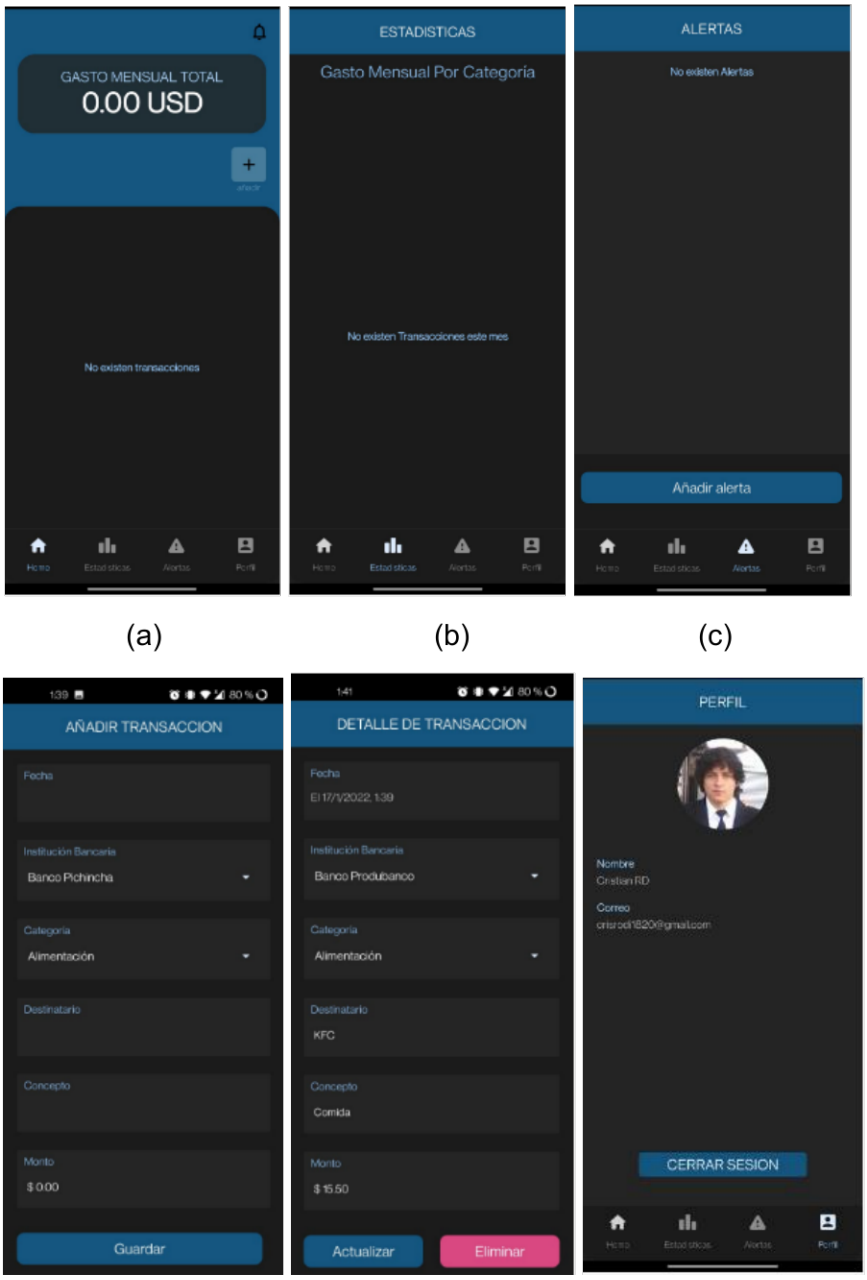


Figura 2.14 Pantalla de comprobación de configuración.

Con el usuario autenticado y la configuración necesaria correcta en Gmail, el cliente observara las diferentes pantallas que se presentan en la Figura 2.15, donde en la imagen (a) se tiene la pantalla inicial, en la que se presentan los consumos generados y el gasto mensual total, la imagen (b) cuenta con el gasto mensual por categoría mediante una gráfica informativa de dichos consumos, la imagen (c) presenta la interfaz en la cual el cliente tiene la opción de ingresar un mensaje de alerta, esto indicando el valor que no desea superar en consumo mensual referenciado alguna categoría de su interés. La interfaz en donde se agregan consumos de forma manual se presenta en la imagen (d) mientras que al momento de obtener los detalles o editar uno se presenta en la imagen (e) y la imagen (f) es la interfaz del perfil del usuario registrado.



(a) (b) (c)

(d) (e) (f)

Figura 2.15 Capturas de las Interfaces de usuario.

Diseño e implementación de la Base de Datos

Para construir la base de datos se hace uso de Firestore servicio que ofrece Firebase, la cual es una base de datos No SQL que almacena los datos en documentos mismos que se organizan en colecciones, por lo que no se cuenta con filas ni columnas, permitiendo que al momento de agregar datos no haya limitaciones si alguno no fue previamente definido.

En la Figura 2.16 presenta una captura de las colecciones utilizados para el proyecto.

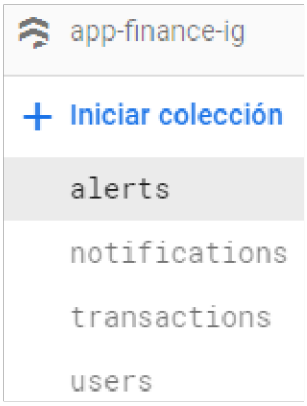


Figura 2.16 Colecciones Base de Datos Firestore

La colección “users” almacena la información relevante de los usuarios autenticados con su cuenta de Gmail en la cual llegaran los correos electrónicos con los consumos realizados, la Figura 2.17 presenta la colección “users” captura desde Firebase.

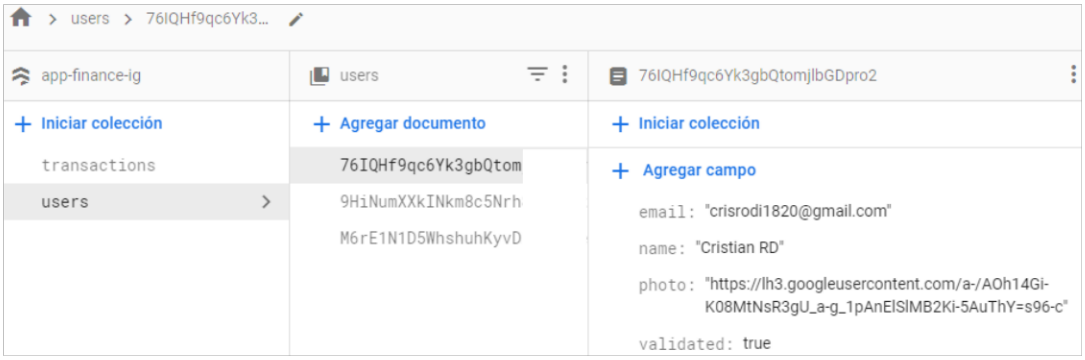


Figura 2.17 Captura de pantalla de Firestore colección usuarios

La Figura 2.18 presenta la colección “transactions” en donde se almacena los datos importantes de la información de consumos del usuario y la referencia del usuario

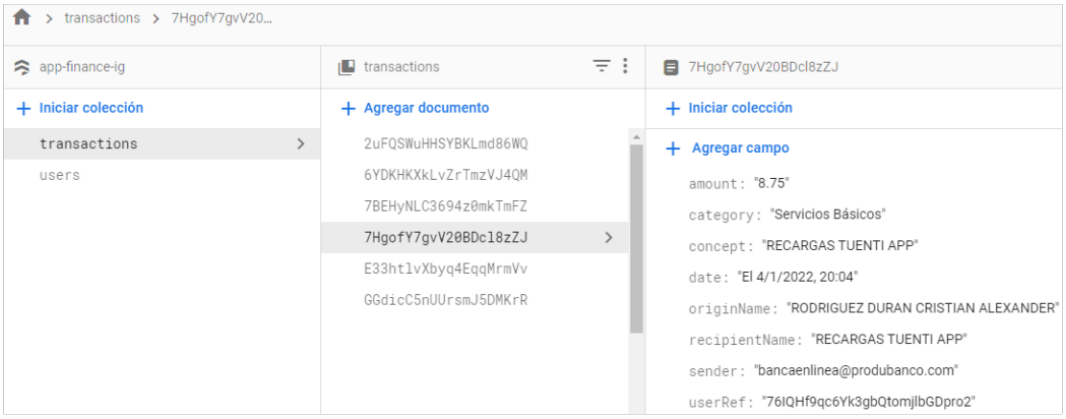


Figura 2.18 Captura de pantalla de Firestore colección transacciones

En la Figura 2.19 se presenta la colección “alerts” dentro de la cual se observa la estructura del documento con los datos necesarios para la creación del valor referencia ingresado por el cliente el cual consta de valor máximo y categoría.

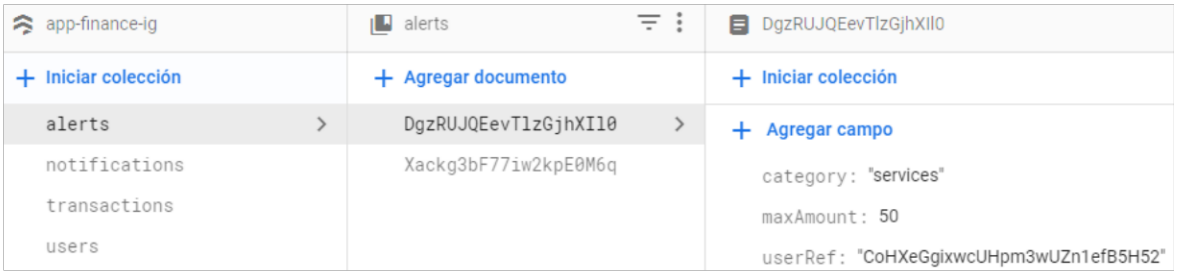


Figura 2.19 Captura de pantalla de Firestore colección alerts

En caso de que el valor ingresado en el documento de la colección “alerts” sea superado en la categoría elegida por el cliente se almacena un mensaje de alerta dentro del documento de la colección “notifications” misma que se puede observar en la Figura 2.20

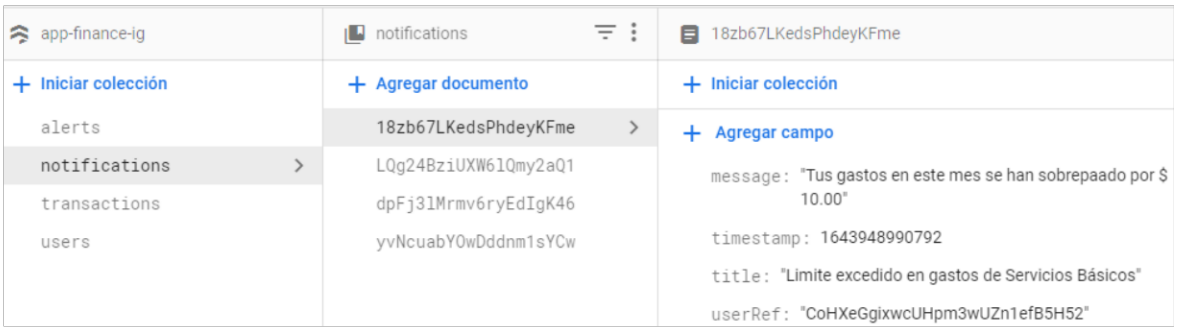


Figura 2.20 Captura de pantalla de Firestore colección notifications

2.6 IMPLEMENTACIÓN

Para la implementación de las interfaces de usuario se utilizó los componentes brindados por React Native y React Navigation este último necesario para la navegación entre pantallas, para la implementación del funcionamiento las herramientas utilizadas adicional a las antes mencionadas son Redux, Firebase y Zoho.

Los componentes y métodos más utilizados para la implementación de interfaces son:

- View: Componente fundamental, el cual funciona como un contenedor de otros componentes y puede estar anidado.

- Image: Componente que permite mostrar imágenes.
- Text: Componente con el que se puede mostrar texto.
- TextInput: Componente para el ingreso de texto a través del teclado del usuario.
- Button: Componente para utilizar un botón el cual generara una acción.
- OnPress: Método que se ejecuta al pulsarse el componente en el cual este definido.

2.6.1 IMPLEMENTACIÓN INTERFAZ

El Código 2.1 presenta la implementación de la interfaz de la pantalla inicial misma que se muestra en la Figura 2.11 (a), dicho segmento de código se detalla lo siguiente:

En las líneas 1 y 2 se importan la librearía “React” desde la librearía “react” y también los componentes necesarios desde “react-native”, desde la línea 4 a la línea 6 se importan el logo, gift y estilos desde los directorios instanciados, de la línea 8 a la 26 se hace la construcción visible de la interfaz, donde la línea 14 presenta el gift superior de la pantalla, de la línea 16 a la 24 se crea el botón de autenticación.

```
1  import React from 'react';
2  import {Image, Text, View, TouchableOpacity} from 'react-native';
3  import GoogleLogoicon from '../../components/Icons/GoogleLogo';
4  import HeroImage from '../../assets/img/plain-credit-card.gif';
5  import styles from './styles';
6
7  interface LayoutProps {
8    onPressLogin: () => Promise<void>;
9  }
10 const Layout = ({onPressLogin}: LayoutProps) => {
11   return (
12     <View style={styles.container}>
13       <Image source={HeroImage} style={styles.image} />
14       <Text style={styles.title}>APP De Manejo De Finanzas</Text>
15       <TouchableOpacity
16         style={styles.button}
17         onPress={onPressLogin}
18         testID="button">
19         <View style={styles.buttonContent}>
20           <GoogleLogoicon />
21           <Text style={styles.buttonText}>INICIAR CON GOOGLE</Text>
22         </View>
23       </TouchableOpacity>
24     </View>
25   );
26 };
27 export default Layout;
```

Código 2.1. Implementación de la interfaz de la pantalla inicial.

Para la navegación entre pantallas se usó el estilo de navegación “Tab navigaton” importado de la librearía “react navigation”, en la Figura 2.21 se presenta la captura de pantalla del estilo de navegación del prototipo.

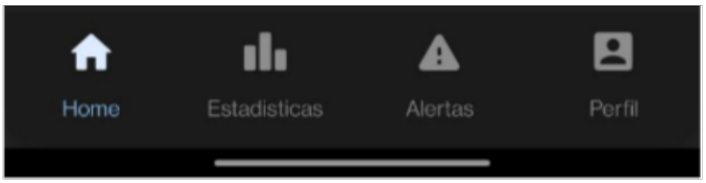


Figura 2.21 Estilo de navegación Tab Navigation

El Código 2.2 presenta una parte importante de la implementación de la navegación entre pantallas de la Figura 2.19. Dentro de las líneas importantes se puede detallar la línea 2, en donde se importa el estilo de navegación desde la librería “react-navigation” desde la línea 3 a la línea 6 se importan las interfaces ya construidas de cada pantalla procedentes de sus respectivos directorios, de la línea 7 a la 27 se crea el contenedor del Tab Navigation y de la línea 16 a la 25 se cuenta con cada pantalla como opción de navegación.

```
1  import React from 'react';
2  import {createBottomTabNavigator} from '@react-navigation/bottom-tabs';
3  import Home from '../../screens/Home';
4  import Statistics from '../../screens/Statistics';
5  .
6  .
7  const Tab = createBottomTabNavigator();
8  const TabNavigator = () => (
9    <Tab.Navigator
10     initialRouteName="home"
11     screenOptions={{
12       headerShown: false,
13       tabBarLabelStyle: styles.tabLabel,
14       tabBarStyle: styles.tabBar,
15       tabBarActiveTintColor: '#145680', }}>
16     <Tab.Screen
17       options={{tabBarIcon: ({color}) => <HomeIcon color={color} />}}
18       name="Home"
19       component={Home}/>
20     <Tab.Screen
21       options={{tabBarIcon: ({color}) => <ReportsIcon color={color} />}}
22       name="Estadisticas"
23       component={Statistics}/>
24     .
25     . />
26   </Tab.Navigator>
27 );
28 export default TabNavigator;
```

Código 2.2. Segmento de código Navegación entre Pantallas.

2.6.2 IMPLEMENTACIÓN AUTENTICACIÓN

El Código 2.3 presenta un segmento correspondiente a la autenticación del usuario en la interfaz inicial, la línea del 1 al 5 se importa los métodos utilizados para autenticación y almacenamiento, de la línea 7 a la 15 se validan datos de entrada y de tener autenticación exitosa se almacenan los datos del cliente para finalmente retornarlos en las líneas 17 a la 20.

```
1  import {
2    AuthenticationService,
3    ExternalStorageService,
4    UserStorageService,
5  } from './ports';
6
7  export const useAuthenticate = (
8    storage: UserStorageService,
9    extStorage: ExternalStorageService,
10 ) => {
11   const authenticate = async (auth: AuthenticationService) => {
12     const userAuth = await auth.auth();
13     const userInfo = await extStorage.getUser(userAuth);
14     storage.updateUser(userInfo);
15   };
16
17   return {
18     user: storage.user,
19     authenticate,
20   };
21 };
```

Código 2.3. Segmento Registro de Usuario.

2.6.3 IMPLEMENTACIÓN REENVÍO Y LECTURA DE CORREOS

Con los correos en el servidor de Zoho procedentes desde la cuenta de Gmail del cliente, gracias a la configuración del reenvío, se hace uso de Zoho Mail y de Webhooks para filtrado y envió de los correos deseados hacia Firebase referenciando el proyecto correspondiente, para lo cual es necesario apuntar con un webhook el endpoint (url) del proyecto mencionado, en la Figura 2.22 se observa los diferentes Webhooks utilizados, en donde el primero “incomingResendEmail” corresponde al reenvío del código necesario para la configuración inicial en Gmail por parte del cliente, la función del segundo webhook “IncomingConfirmationEmail” es de validar si se configuro de forma correcta el reenvío de correos por parte del cliente y el ultimo “IncomingTransaccionEmail” El reenvío de correos en formato JSON hacía el URL del proyecto creado en Firebase.

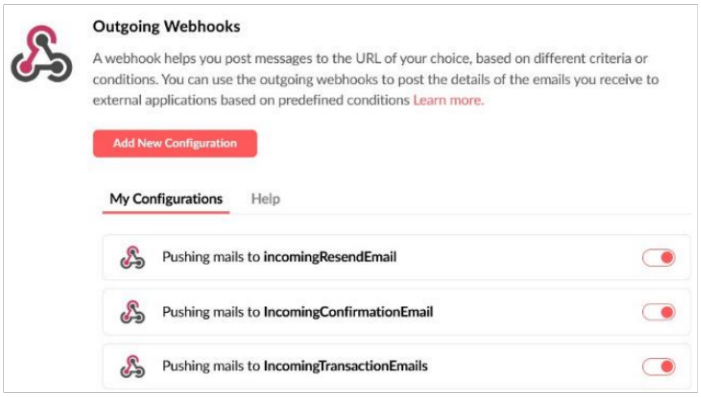


Figura 2.22 Webhooks utilizados

Dentro de Zoho se configuran los Webhooks antes mencionados, en la Figura 2.23 se presenta la configuración del webhook “incomingResendEmail”, en donde se ubica el URL del proyecto referenciando a la función cargada y se configura las condiciones de filtrado, este webhook se encarga de enviar el código que se hace referencia en la Figura 2.13 imagen (c)

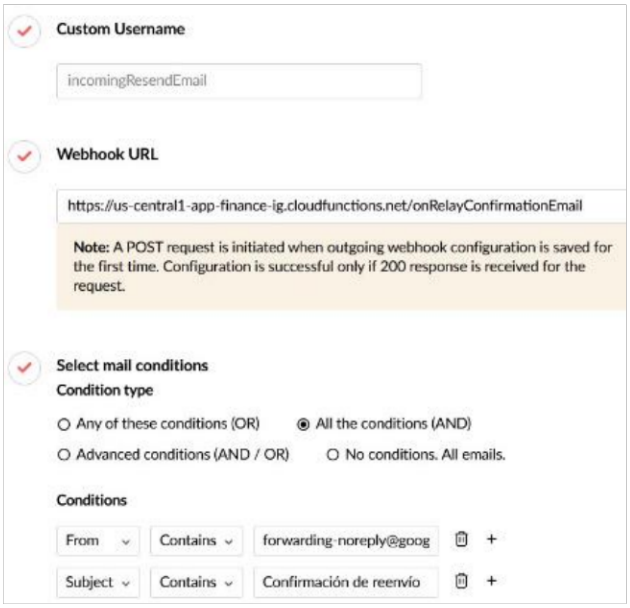


Figura 2.23 Captura configuración webhook “incomingResendEmail”

La Figura 2.24 se presenta la configuración del webhook “IncomingConfirmationEmail” mismo que se encarga de confirmar que este correctamente configurado el reenvío de correos en la cuenta de Gmail del usuario, este webhook se activa en la acción que se presenta en la Figura 2.14.

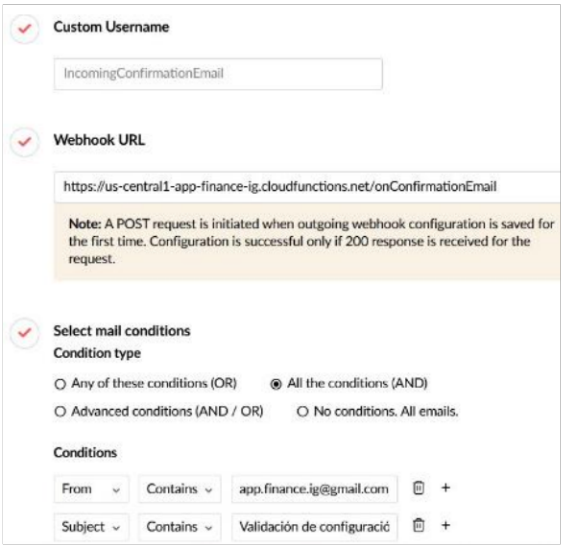


Figura 2.24 Captura configuración webhook “IncomingConfirmationEmail”

Finalmente, la Figura 2.25 presenta la configuración del webhook “IncomingTransaccionEmail” que se encarga del reenvío de los correos filtrados hacia el URL (endpoint) de la función que extrae los datos la cual esta carga en la nube dentro del proyecto en Firebase.

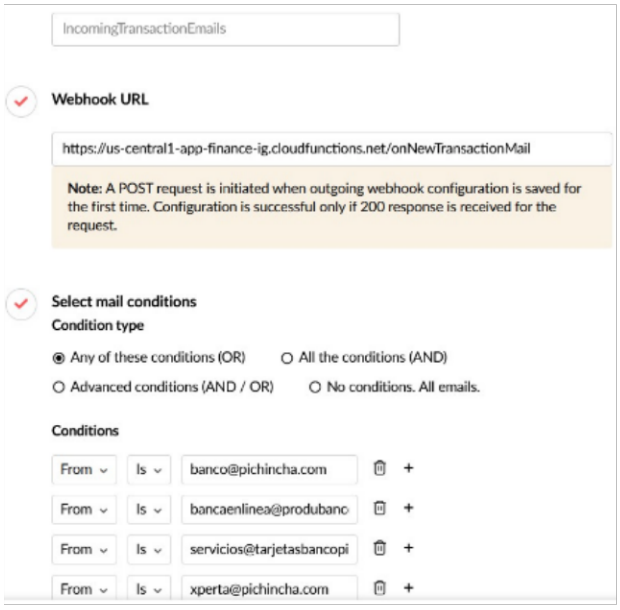


Figura 2.25 Captura configuración webhook “IncomingConfirmationEmail”

Con los Webhooks configurados en Zoho estos envían datos en JSON a las funciones correspondientes cargadas en la nube.

El Código 2.4 presenta un fragmento del código utilizado para la lectura de correos electrónicos mediante la API facilitada por Zoho (línea 3), en la línea 1 se importan las funciones cargadas en Firebase, línea 4 se importa la base de datos Firestore, de la línea 7 a la 11 se obtienen los identificadores tanto del correo, proyecto y usuario procedente,

```
1  import {https,config,auth,firestore as firestoreFunc,} from "firebase-
2  functions"
3  import sgMail from "@sendgrid/mail"
4  import Firestore from "@google-cloud/firestore"
5  .
6  .
7  const SENDGRID_API_KEY = config().sendgrid?.api?.key
8  const ORIGIN_EMAIL = config().origin?.mail
9  const PROJECTID = "app-finance-ig"
10 const COLLECTION_NAME = "users"
11 sgMail.setApiKey(SENDGRID_API_KEY)
12
13 const firestore = new Firestore.Firestore({
14   projectId: PROJECTID,
15   timestampsInSnapshots: true,
16 })
17
18 exports.sendEmail = https.onRequest(async (req, res) => {
19   const {to, subject, html, text} = req.body
20   try {
21     await sgMail.send({
22       from: ORIGIN_EMAIL,
23       to,
24       subject,
25       html,
26       text,
27     })
28     res.send(buildResponse({message: "Email sent successfully"}))
29   } catch (error) {
30     res.status(500).send(buildResponse({error: error as Error}))
31   }
32 })
```

Código 2.4. Lectura de correos.

El Código 2.5 presenta el segmento con el que se extraen los datos necesarios e importantes del correo electrónico emitido por la entidad bancaria, dentro del cual se hace uso de los siguientes métodos:

- Email.substring(inicio,fin): El presente método selecciona una porción del texto referenciado desde el índice “inicio” hasta el índice “fin” mismos que se instancian en el método inicialmente, el índice “fin” se puede omitir y en tal caso seleccionara desde el índice “inicio” hasta el final del texto.

- Email.indexOf(“referencia”): Dicho método devuelve el índice de la primera letra en donde este ubicado la palabra o valor “referencia” dentro del texto.
- Email.trim().- Este método elimina los espacios presentes en los extremos del texto al cual se lo referencia.

La línea 1 corresponde al asunto del mail el cual se usa como filtro, de la línea 2 a la 8 se extrae el nombre del titular de la cuenta, en las líneas 10 hasta la 13 se obtiene el valor del consumo, el nombre del local o beneficiario se extrae desde la línea 15 hasta la 19, finalmente se retorna las variables con la información lista desde la línea 21 a la 25.

```
1  case "Consumo Tarjeta Visa Débito": {
2        const originNameStrStart =
3          emailContent.indexOf("<br />",
4  emailContent.indexOf("Estimado/a")) + 6
5        const originNameStrEnd = emailContent.indexOf(
6          "</font>",originNameStrStart, )
7        const originName = emailContent
8          .substring(originNameStrStart, originNameStrEnd).trim()
9
10       const amountStrStart = emailContent.indexOf("USD ") + 4
11       const amountStrEnd = emailContent.indexOf(" en ",amountStrStart)
12       const amount = emailContent.substring(amountStrStart,
13 amountStrEnd).trim()
14
15       const recipientNameStrStart = amountStrEnd + 4
16       const recipientNameStrEnd = emailContent.indexOf(
17         ".<br />", recipientNameStrStart, )
18       const recipientName = emailContent
19         .substring(recipientNameStrStart, recipientNameStrEnd).trim()
20
21       return {
22         originName,
23         recipientName,
24         amount,
25         concept: recipientName,
26       }}
```

Código 2.5. Extracción de información del correo.

En el Código 2.6 se presenta el segmento de código utilizado para agregar un nuevo consumo, la función presente se encuentra cargada en Google Cloud esto sincronizado con Firebase, en donde se observa desde la línea 3 a la 11 los datos del nuevo consumo.

En las líneas de la 13 a la 18 se llama a la colección “transaction” de Firestore referenciando al usuario que se va a agregar el consumo, finalmente en la línea 20 a la 22 se agrega el consumo a la base de datos.

```
1  exports.addTransaction = https.onCall(async (data, context) => {
2    const transaction = {
3      recipientName: data.recipientName,
4      amount: data.amount,
5      concept: data.concept,
6      date: data.date,
7      category: data.category,
8      originName: data.originName,
9      sender: data.sender,
10     userRef: context.auth?.uid,
11     timestamp: getTimestamp(data.date)
12   }
13   const id = context.auth?.uid || ""
14   const userRef = firestore.collection(COLLECTION_NAME).doc(id)
15   const userData = await userRef.get()
16   const customCategories = getCustomCategories(transaction, userData)
17   if(customCategories){
18     await userRef.update({customCategories: customCategories})
19   }
20   const transactionsRef = firestore.collection("transactions")
21   const res = await transactionsRef.add({...transaction})
22   return {id: res.id}
23 })
```

Código 2.6. Agregar nuevo consumo.

La actualización y eliminación de un consumo se presenta en el Código 2.7 donde se puede observar la exportación de la función en la línea 1, de la línea 2 a la 6 se crea la constante que va a contener la información del consumo a modificar, desde la línea 8 a la 11 se extraen los datos del usuario involucrado, en las líneas 12 y 13 se valida la categoría del consumo, la actualización del consumo se efectúa en las líneas 14 a la 16 y finalmente de la línea 18 a la 20 es el segmento que elimina el consumo seleccionado.

```
1  exports.updateTransaction = https.onCall(async (data, context) => {
2    const transaction = {
3      recipientName: data.recipientName,
4      amount: data.amount,
5      . }
6    const id = context.auth?.uid || ""
7    const userRef = firestore.collection(COLLECTION_NAME).doc(id)
8    const userData = await userRef.get()
9    const customCategories = getCustomCategories(transaction, userData)
10   if(customCategories){
11     await userRef.update({customCategories: customCategories}) }
12   const transactionsRef =
13 firestore.collection("transactions").doc(data.id)
14   return await transactionsRef.update(transaction)})
15
16 exports.deleteTransaction = https.onCall(async id => {
17   const transactionsRef = firestore.collection("transactions").doc(id)
18   return await transactionsRef.delete()})
```

Código 2.7. Actualizar y Eliminar un consumo.

El Código 2.8 presenta la sección de la funcionalidad de añadir, actualizar y eliminar alertas de lo cual se requiere el monto máximo y la categoría a la que se va a llevar control, desde la línea 1 a la 10 se presenta el código de añadir alerta, de la línea 12 a la 20 se cuenta con función para modificar la alerta, finalmente de la línea 22 a la 25 se presenta la función de eliminar la alerta registrada.

```
1  exports.addAlert = https.onCall(async (data, context) => {
2    const alert = {
3      maxAmount: data.maxAmount,
4      category: data.category,
5      userRef: context.auth?.uid,
6    }
7    const alertsRef = firestore.collection("alerts")
8    const res = await alertsRef.add({...alert})
9    return {id: res.id}
10 })
11
12 exports.updateAlert = https.onCall(async (data, context) => {
13   const alert = {
14     maxAmount: data.maxAmount,
15     category: data.category,
16     userRef: context.auth?.uid,
17   }
18   const alertsRef = firestore.collection("alerts").doc(data.id)
19   return await alertsRef.update({...alert})
20 })
21
22 exports.deleteAlert = https.onCall(async id => {
23   const alertsRef = firestore.collection("alerts").doc(id)
24   return await alertsRef.delete()
25 })
```

Código 2.8. Agregar, actualizar y eliminar alertas

.

3. RESULTADOS, CONCLUSIONES Y RECOMENDACIONES

3.1 RESULTADOS

3.1.1 PRUEBAS

En la presente sección se presentan las pruebas de funcionamiento de cada apartado, dando un resumen del proceso utilizado y como se llevó a cabo para obtener los resultados deseados y planificados en la toma de requerimientos.

3.1.1.1 Prueba de autenticación y reenvío de correos.

Firebase provee el servicio de autenticación y se utiliza el método de Gmail, la Figura 3.1 muestra los usuarios autenticados en el proyecto del prototipo, la seguridad, y validación de credenciales correctas se encarga directamente Gmail, dando una respuesta booleana al proyecto, si esta es positiva también otorga la información del usuario autenticado.

Figura 3.1 Captura de pantalla de Firestore autenticación

En la Figura 3.2 se presenta el código generado al momento de configurar el reenvío de correos desde PC con la cuenta de Gmail que el usuario se autentico previamente.

Figura 3.2 Recepción de código para reenvío de correos

Una vez confirmado el código de reenvío que llega al correo del cliente, se valida si toda la configuración de este proceso fue realizada correctamente por lo que el cliente recibe un correo informando aquello, la Figura 3.3 presenta el correo de validación de configuración de reenvío.

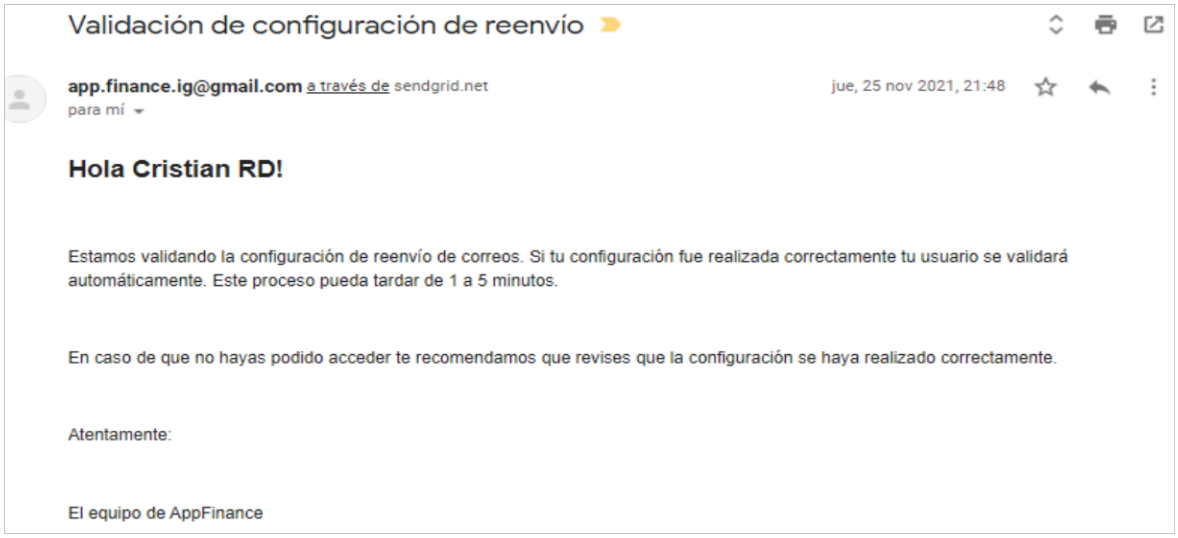
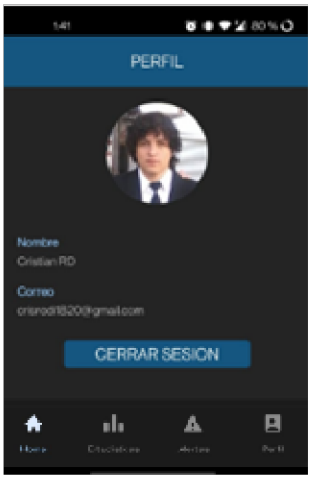


Figura 3.3 Correo de validación de configuración de reenvío

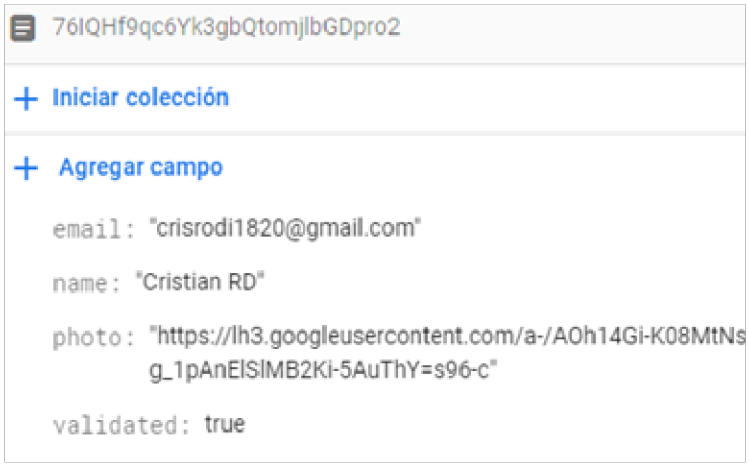
3.1.1.2 Registro del usuario

La presente prueba tiene como finalidad validar que el usuario una vez finalizado el proceso de configuración del reenvío de correos, se encuentre registrado con su información obtenida de la cuenta de Gmail, misma que será visible en la aplicación móvil en el módulo “Perfil”. La Figura 3.4 muestra la captura de pantalla del perfil del usuario (a) y del almacenamiento en Firebase (b).

El correo electrónico con el que el usuario se haya registrado debe ser el mismo en el cual recibe los correos de las entidades bancarias por consumos.



(a)



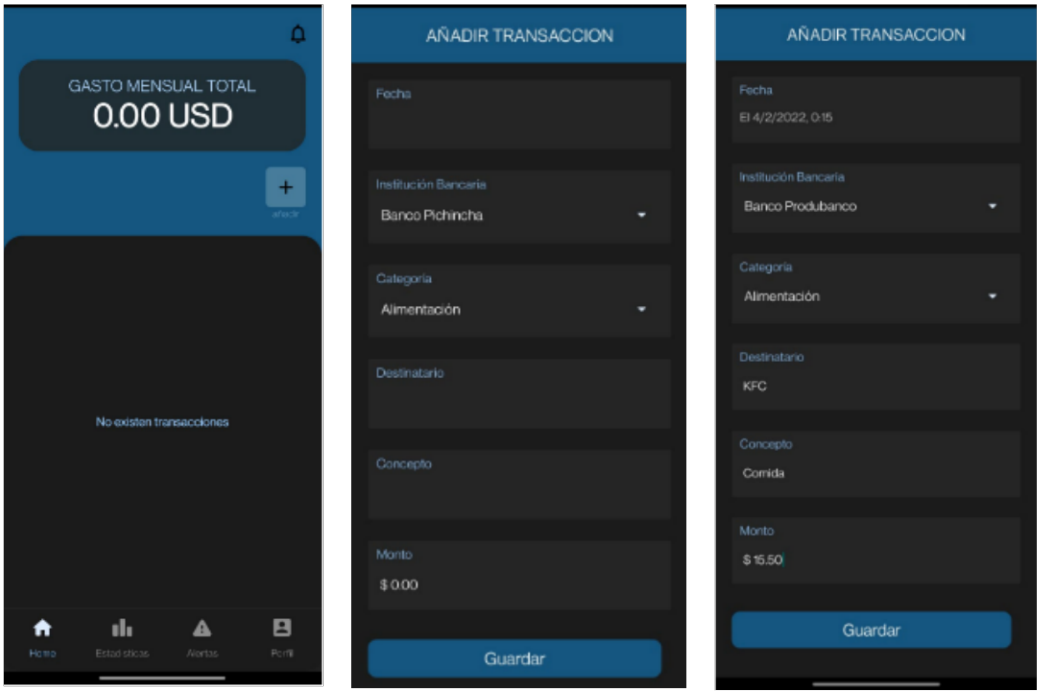
(b)

Figura 3.4 Registro de usuario Perfil desde app móvil (a) registro en Firebase (b)

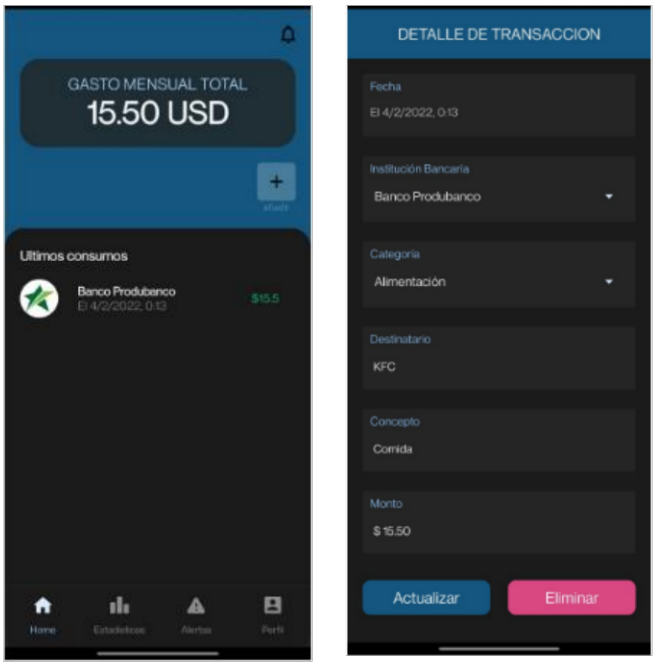
3.1.1.3 Prueba agregar, actualizar o eliminar consumos manualmente

Las pruebas de agregar, actualizar y eliminar consumos manualmente se presentan en la Figura 3.5. Una vez el cliente haya finalizado con la configuración del reenvío accederá a las funcionalidades del prototipo, en la imagen (a) se reflejan los consumos generados y también se encuentra disponible el botón “añadir” desplegándose un formulario para el ingreso de un consumo nuevo tal como se observa en la imagen (b), la imagen (c) es una presenta el formulario completo con los datos que el cliente desee registrarlo, una vez que haya finalizado el formulario y se haya guardado se refleja en pantalla como se valida en la imagen (d) finalmente si ingresamos a algún consumo registrado como se observa en la imagen (e) contamos con la opción de actualizar o eliminar.

Cada consumo agregado se suma al balance total, mismo que se observa en la parte superior de la imagen (a) para tener presente cuanto es el total del consumo que se ha ido realizando, este varía dependiendo de la modificación de dichos consumos o su eliminación.



(a) (b) (c)



(d) (e)

Figura 3.5 Capturas de pantalla de añadir un consumo manualmente

Los datos guardados en la aplicación se almacenan en la base de datos lo cual se presenta en la Figura 3.6 en donde se puede observar que el consumo esta enlazado con el usuario correspondiente bajo el campo “userRef”.

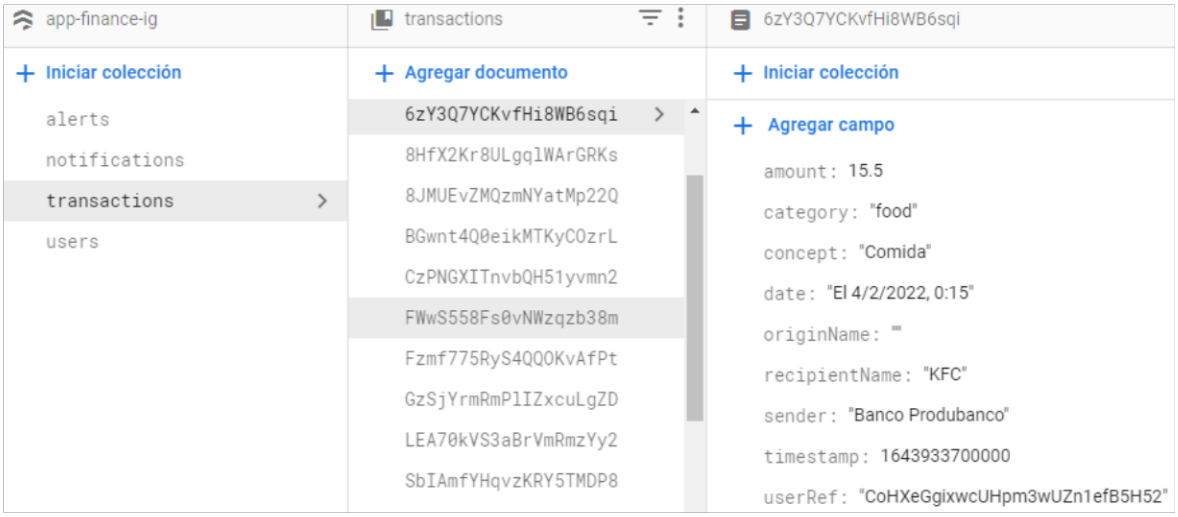


Figura 3.6 Almacenamiento de consumo en Firestore

3.1.1.4 Prueba obtención de datos desde correo electrónico

Una de las problemáticas del presente trabajo de integración es el de realizar pruebas de consumos con los correos electrónicos de la entidad bancaria, ya que cada que se desea probar su funcionamiento era necesario consumir con tarjeta para recibir el correo correspondiente y validar que se refleje en el prototipo, como alternativa para aquello se hizo uso de la herramienta Postman, que permite realizar peticiones HTTP hacia un endpoint, en este caso se utiliza el método POST para simular él envió de correos de una entidad bancaria hacia el endpoint (url) del proyecto alojado en Firebase.

Previo al uso de Postman es necesario recolectar la información que se va a enviar al endpoint, existen diferentes formatos de correos emitidos por una misma entidad bancaria que reflejan consumos, en la Figura 3.7 se presenta un ejemplo de correo de consumo con tarjeta de débito de Produbanco.

---------- Forwarded message ---------
De: **Produbanco enlínea** <bancaenlinea@produbanco.com>
Date: lun., 12 de octubre de 2020 00:33
Subject: Consumo Tarjeta Visa Débito
To: <jorgevallejo62@gmail.com>

Produbanco
Grupo Promerica

Estimado/a
VALLEJO DURAN JORGE LUIS

Fecha y Hora de Proceso: 12/Octubre/2020 0:33

Transacción: Registro de Consumo

Te Informamos que se acaba de registrar un consumo con tu tarjeta Visa Débito Produbanco debitado de la cuenta ANA XXXXXX42420 por el valor de USD 3.95 en PAYPAL *STEAM GAMES 042588996.

Si no realizaste esta transacción favor comunícate a nuestro Call Center al 1700 123 123 o al (02) 2996 200.

Este es un servicio de Produbanco enlínea.

La información y adjuntos contenidos en este mensaje electrónico son confidenciales y reservados; por tanto no pueden ser

Figura 3.7 Consumo Notificado por entidad bancaria

El Código 3.1 presenta la respuesta del correo mencionado en la Figura 3.7, una vez que ya se haya procesado su información con los servicios backend de Firebase, dicha información se extrae y se obtienen los datos en formato JSON. Las líneas principales del Código 3.1 se detallan a continuación:

Línea 2 informa el remitente del correo (entidad bancaria), la línea 3 se presenta el asunto del correo mismo que varía al igual que su formato este es de utilidad para hacer

el filtrado, en la línea 4 se referencia al destinatario del mail por último desde la línea 5 se tiene el cuerpo del correo en formato HTML con toda la información del consumo generado, en el cual se puede observar marcado el valor la fecha el beneficiario.

```
1  {
2      "sender": "bancaenlinea@produbanco.com",
3      "subject": "Consumo Tarjeta Visa Débito",
4      "toAddress": "<crisrodi1820@gmail.com>",
5      "html": "<div><div dir=ltr><img alt=Produbanco - Grupo Promer
6  ica src=https://www.produbanco.com/produnet/imagen/logo-
7  enlinea.png /> <div class=x_1707579826gmail_quote><div dir=ltr><d
8  iv class=x
9  .
10 .
11 ALIGN :  justify; ><font size=2 face=Tahoma></font></p><p></p><p>
12 <font size=2 face=Tahoma>Estimado/a <br />VALLEJO DURAN  JORGE LU
13 IS</font></p><p><font size=2 face=Tahoma>Fecha y Hora de Proceso:
14  12/Octubre/2020  0:33</font></p><p><font size=2 face=Tahoma>Tran
15 sacci&oacute;n: Registro de Consumo</font></p><p><font size=2 fac
16 e=Tahoma>Te Informamos que se acaba de registrar un consumo con t
17 u tarjeta Visa D&eacute;bito Produbanco debitado de la cuenta ANA
18  XXXXXX42420 por el valor de USD 3.95 en PAYPAL *STEAM GAMES 0425
19 88996.<br /> <br />Si no realizaste esta transacci&oacute;n
20 favor comun&
21 nea.</font></p></td></tr></tbody><tbody></tbody></table><p></p><t
22 able width
23 .
24 }
25
26
```

Código 3.1. Datos de consumo, formato JSON

Una vez obtenido el objeto JSON con el formato del correo del consumo, se configura Postman para la simulación de envío de correo bancario, por lo que se crea un archivo de nueva petición, se escoge el método POST, en la opción “Body” se escoge “raw” y formato JSON. En el método POST se coloca el URL del servicio cargado en Firebase, esto se presenta en la Figura 3.8.

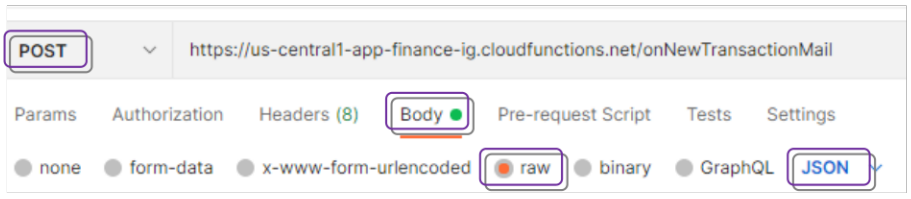


Figura 3.8 Método POST hacia URL del servicio de Firebase

El objeto JASON presentado en el Código 3,1 es copiado en el “Body” del documento creado previamente en Postman tal como se observa en la Figura 3.9, en dicho documento se puede modificar para cuestión de pruebas el sitio de consumo, su valor, y el destinatario en donde se coloca el correo del cliente antes registrado, dependiendo del correo del destinatario ingresado se enviará la simulación del correo a su perfil.

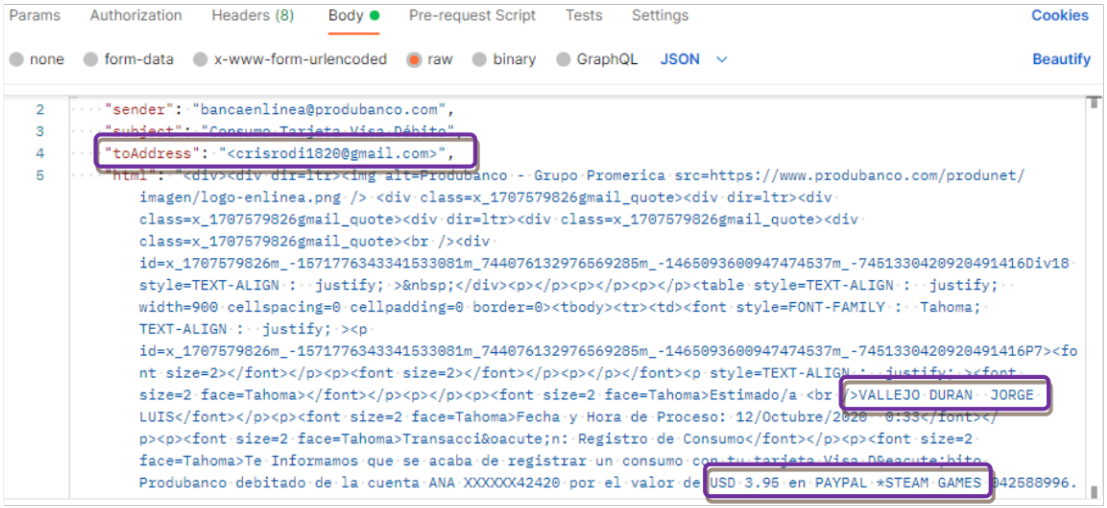


Figura 3.9 Captura de Postman con el objeto JASON de un correo de consumo

Una vez realizado el paso anterior se envía el objeto JSON y en la herramienta Postman refleja el mensaje presentado en la Figura 3.10, tal mensaje confirma que la acción se ejecutó de forma exitosa.

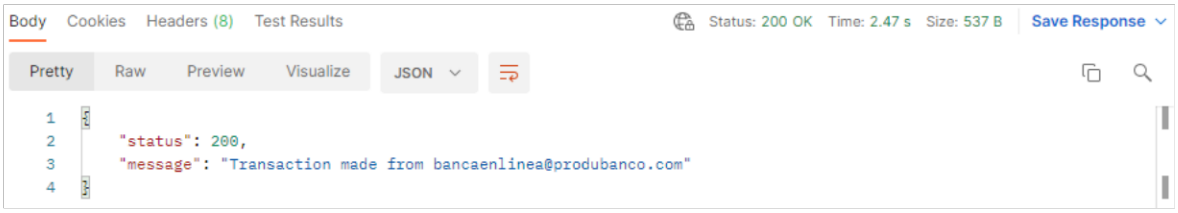
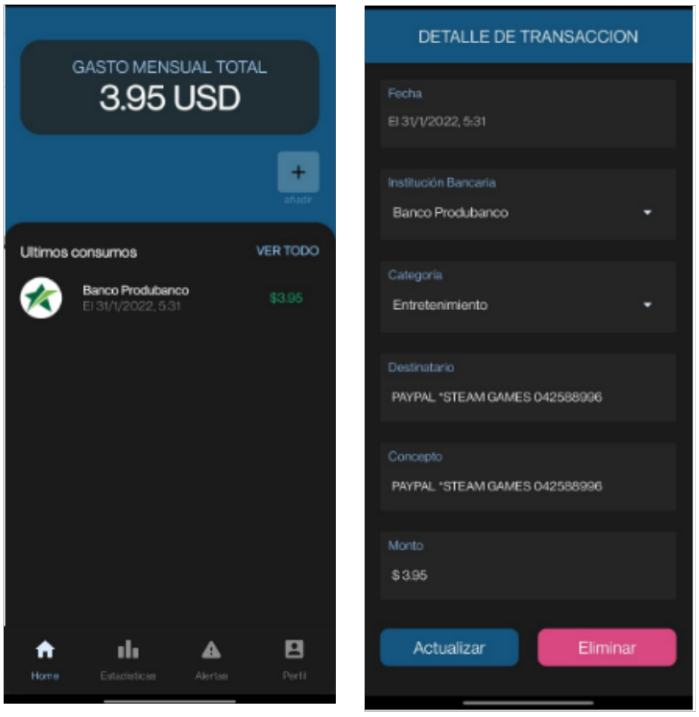


Figura 3.10 Confirmación de envío de datos desde Postman

Con el correo simulado ya enviado, en el prototipo se actualiza y refleja el consumo de dicho mail, la Figura 3.11 muestra el consumo emitido desde Postman donde se aprecia la imagen (a) que presenta el consumo con su valor, en la imagen (b) se presenta los datos de dicho consumo y en caso de que se desee también se puede modificar de forma manual.



(a) (b)

Figura 3.11 Comprobación de ingreso de consumos y toma de valores.

En la Figura 3.12 se presenta una captura de pantalla de la colección y documento del servicio Firestore en el cual refleja el consumo que se simulo vía Postman almacenando la información relevante.

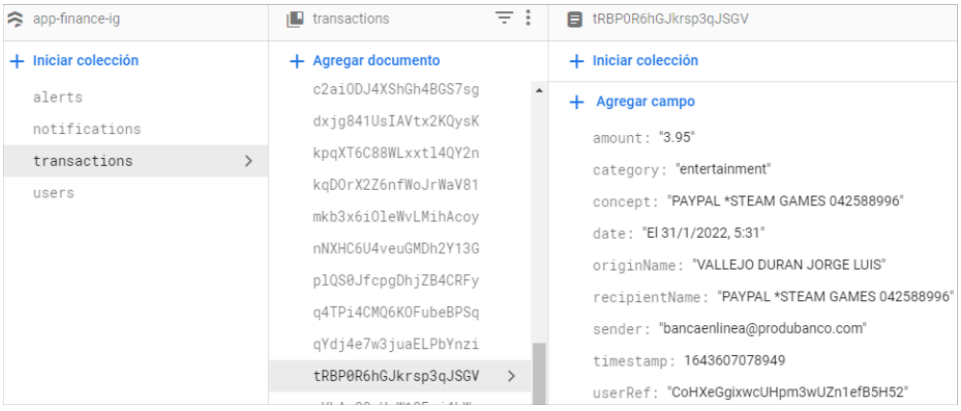
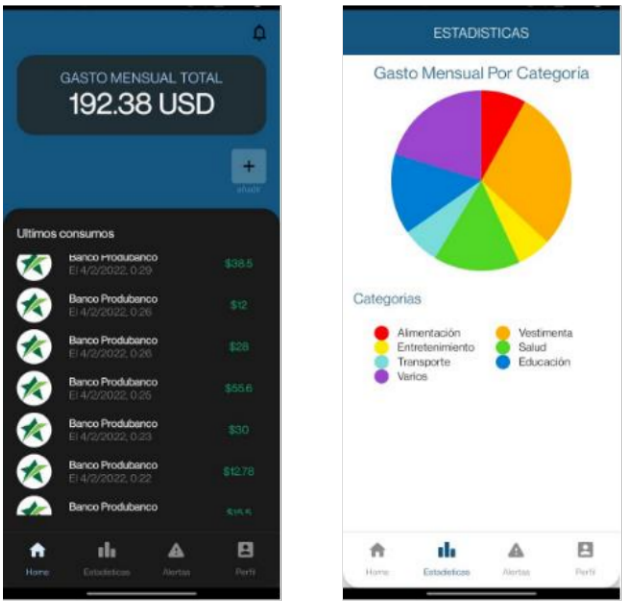


Figura 3.12 Datos almacenados en Firestore.

3.1.1.1 Prueba de consumos por categorías (Gráfica Estadística)

Para las pruebas de las estadísticas de consumo se ingreso varios gastos de forma manual de diferentes categorías esto se presenta en la imagen (a) de la Figura 3.13 con dichos

consumos también se verifica que el gasto total mensual se actualiza con el valor de cada consumo ingresado, en la imagen (b) se observa una imagen informativa la cual contiene los gastos mensuales por categorías de forma proporcional



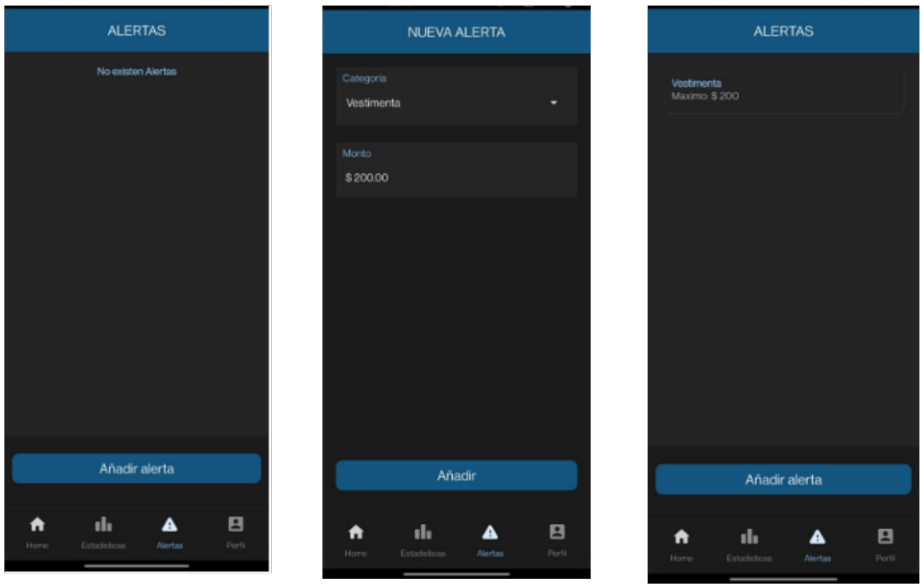
(a) (b)

Figura 3.13 Datos almacenados en Firestore.

3.1.1.2 Prueba Ingreso de Alerta

El prototipo permite ingresar añadir una alerta, la cual solicita dos datos (valor máximo y categoría a la que se desea dar seguimiento) el funcionamiento de esta opción se basa en los consumos mensuales de dicha categoría donde el valor total de estos se compara con el ingresado por el cliente en la alerta y de confirmar que se rebaso dicho valor se crea un mensaje de notificación indicando aquello y por cuanto rebaso dicho consumo.

En la Figura 3.14 se presenta la creación de la alerta en donde la imagen (a) presenta la interfaz inicial, la imagen (b) el ingreso de los datos para crear la alerta y la imagen (c) indica la alerta ya registrada en el prototipo.



(a) (b) (c)

Figura 3.14 Creación de alerta dentro del prototipo

Una vez creada la alerta se verifica la creación de esta en la base de datos de Firestore en la colección “alerts” donde se encuentra el valor máximo, categoría y el código del usuario que la ingreso. La Figura 3.15 muestra el almacenamiento de la alerta en Firestore.

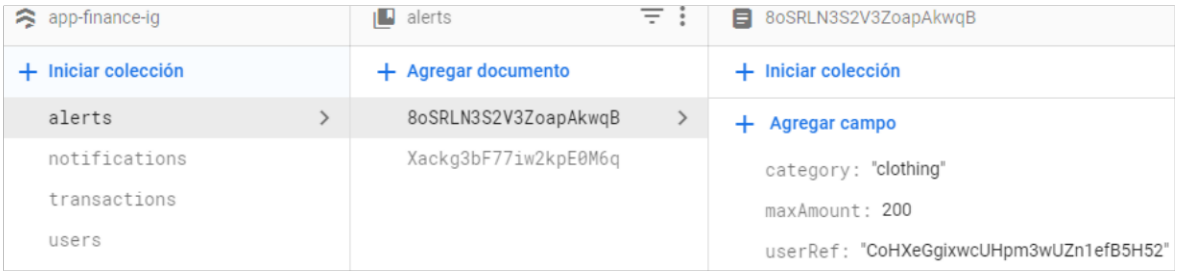
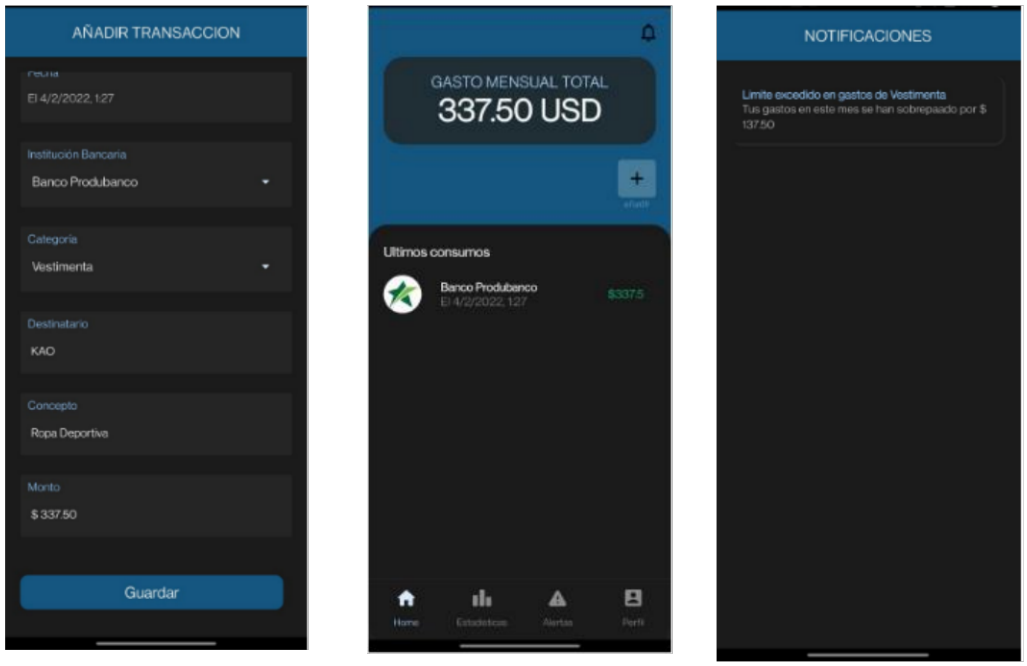


Figura 3.15 Almacenamiento de la Alerta en Firestore

Para comprobar que se actiave el mensaje de notificacion se crea un consumo que sobrepase el valor antes almacenado mismo que se observa en la imagen (a) de la Figura 3.16, en la imagen (b) se observa el consumo ingresado y el valor de este lo cual genera el mensaje de la imagen (c) que indica que se excedio el valor ingresado.



(a) (b) (c)

Figura 3.16 Prueba de mensaje de notificación

El mensaje generado en la imagen (c) de la Figura 3.16 se comprueba que se registró en la base de datos dentro de la colección “notifications” esto se presenta en la Figura 3.17.

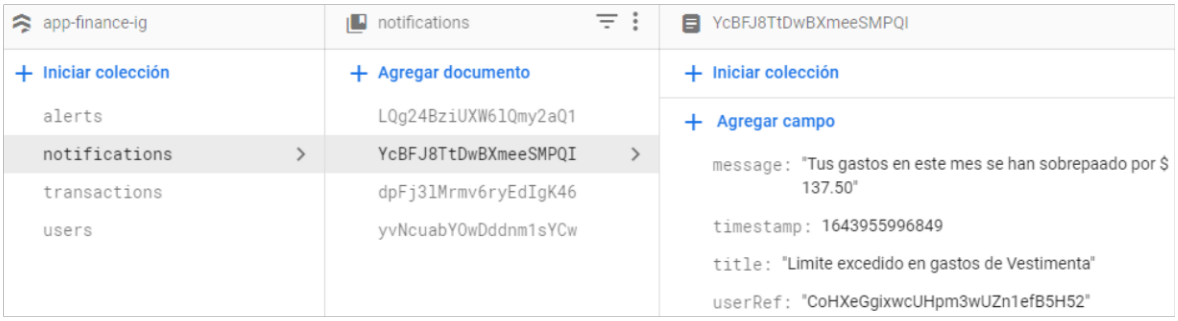


Figura 3.17 Almacenamiento de notificación en Firestore

3.2 PRESENTACION DE RESULTADOS

Para la obtención de resultados y validar el funcionamiento del prototipo, se realizó una encuesta a 10 usuarios teniendo los siguientes resultados:

Pregunta 1: ¿Ingresó con su cuenta de Gmail?

10 (100.0%): sí
- (0.0%): no

El resulto confirma la funcionalidad de autenticación con cuenta de Gmail en todos los usuarios que probaron el prototipo.

Pregunta 2: ¿Qué tan claro y sencillo le resultó seguir la guía de configuración de reenvío de correos?

4 (40.0%): Muy Sencillo
4 (40.0%): Sencillo
2 (20.0%): Regular

En los resultados de la pregunta 2 se observa que el 20% de los encuestados les resulto regular seguir la guía de configuración de reenvió de correo electrónico debido a que se requiere un ordenador para realizarlo.

Pregunta 3: ¿Puede observar el perfil de su cuenta en la aplicación con su información personal?

10 (100.0%): sí
- (0.0%): no

Todos los encuestados validaron que en el prototipo se creó su perfil personal.

Pregunta 4: ¿En el manejo de la aplicación, pudo realizar acciones como: agregar, actualizar y eliminar consumos?

10 (100.0%): sí
- (0.0%): no

Los encuestados confirmaron que el prototipo permite agregar, actualizar y eliminar consumos de forma manual.

Pregunta 5: ¿Refleja sus consumos en el prototipo? *

9 (90.0%): sí
1 (10.0%): no

La mayoría de los participantes en la encuesta confirman que reflejaron consumos en la aplicación con sus detalles, siendo el 10% el caso de excepción debido a que el correo con el que se autentico en la aplicación no es el mismo del que tiene registrado en su entidad bancaria.

Pregunta 6: ¿Al momento de contar con consumos registrados puede visualizarlo por categorías en la gráfica informativa? *

10 (100.0%): sí
- (0.0%): no

Los resultados de la pregunta 6 permite conocer que todos los participantes pudieron observar sus consumos por categorías presentados en una gráfica informativa.

Pregunta 7: ¿El prototipo le permite añadir una alerta y puede visualizarla al momento de sobrepasar el límite del consumo seleccionado? *

8 (80.0%): sí
2 (20.0%): no

El 80% de los encuestados comprobó el manejo de alertas, y el mensaje que este refleja en el apartado de notificaciones como información del valor excedido.

Pregunta 8: ¿Qué tan útil le parece esta aplicación? *

5 (50.0%): Muy Bueno
3 (30.0%): Bueno
2 (20.0%): Regular

Los resultados obtenidos en esta pregunta reflejan una aceptación de la mayoría de los encuestados a la aplicación presentada, mientras que en el porcentaje regular prefieren como método de pago otra alternativa.

3.3 CONCLUSIONES

- La optimización de recursos en el ámbito personal es un factor de gran importancia en la estabilidad económica de cada individuo. El manejo de finanzas mediante el desarrollo de una aplicación Android mejora la administración evitando un desequilibrio en el capital.

- Los índices de seguridad cuando se utiliza como método de pago el efectivo ha generado que los consumidores usen medios digitales con el fin de garantizar confianza en sus pagos. Sin embargo, esto no genera conciencia en los usuarios del consumo de sus productos. El desarrollo del prototipo presentado en el proyecto de titulación que crea una organización de los egresos generados en diversas secciones como salud, vestimenta, educación, que garantizan una mejor distribución monetaria que podrían reducir las deudas por falta de dinero.

- El framework React Native facilita la organización de software. Además, tiene compatibilidad con JavaScript y TypeScript que agiliza el desarrollo de aplicaciones móviles que lo categoriza como una tecnología multiplataforma, como consecuencia de los beneficios mencionados se seleccionó para el desarrollo de la aplicación presente en el trabajo de integración.

- Las características importantes de la aplicación son ser escalable y segura debido a que se implementó autenticación, los datos son almacenados en una base de datos NoSQL y sus funciones se encuentran cargadas en la nube. Estas propiedades pudieron ser insertadas de manera sencilla ya que se empleó Firebase.

- El formato JSON permite el intercambio de datos que acelera los procesos. El uso de Zoho nace como una opción por las políticas de seguridad proveniente de Gmail que no permite la lectura de los correos electrónicos con el uso de su API, para lo cual se realiza un reenvío de los correos hacia Zoho que facilita la lectura de los correos.

- La necesidad de información sobre los consumos de los usuarios permite conocer la aceptación de las personas ante una aplicación de finanzas cuyos resultados se observaron en las encuestas, que fueron la iniciativa para crear este proyecto de titulación.

3.4 RECOMENDACIONES

- Para el desarrollo de aplicaciones móviles iniciales con React Native se sugiere el uso de la herramienta Expo la cual cuenta con módulos ya implementados y facilita la compilación y pruebas con su aplicación Expo Go o directamente vía Web.
- El presente prototipo fue diseñado con el fin de utilizarlo para la lectura de correos procedentes de una entidad bancaria en este caso Produbanco, pero se recomienda agregar más formatos de otras entidades.
- Antes de utilizar React Native como framework de desarrollo para aplicaciones móviles se sugiere tener conocimientos básicos de JavaScript, HTML, CSS. Ya que esto facilitaría el entendimiento y agilitaría procesos de desarrollo.
- Se recomienda implementar más alternativas de autenticación a la actual (Gmail) implementada con el fin de que se propague con mayor facilidad y sea mas cómoda para el usuario.

4. REFERENCIAS BIBLIOGRÁFICAS

[1] «Agencia EFE,» 7 Diciembre 2018. [En línea]. Available: https://www.efe.com/efe/america/tecnologia/las-personas-conectadas-a-internet-ya-son-mayoria-en-el-mundo/20000036-3836112. [Último acceso: Diciembre 2019].

[2] P. A. MONSERRAT, «El País,» 8 Enero 2016. [En línea]. Available: https://elpais.com/economia/2016/01/07/actualidad/1452159171_118239.html. [Último acceso: Septiembre 2019].

[3] C. Gackenheimer, Introduction-to-React, New York: Springer Science+Business Media New York, 2015.

[4] V. Novick, React Native - Building Mobile Apps with JavaScript, Packt, 2017.

[5] «Redux,» Product Pains, 2016. [En línea]. Available: http://es.redux.js.org/?q=#. [Último acceso: Noviembre 2021].

[6] «Product Pains,» 2016. [En línea]. Available: https://es.redux.js.org/docs/introduccion/tres-principios.html. [Último acceso: Noviembre 2021].

[7] C. Khawas y P. Shah, «Application of Firebase in Android App Development-A Study,» *International Journal of Computer Applications,* pp. 49-53, Junio 2018.

[8] Google Developers, «Firebase,» Google Developers, 2018. [En línea]. Available: https://firebase.google.com/docs. [Último acceso: Noviembre 2021].

[9] Google Developers, «Firebase,» Octubre 2021. [En línea]. Available: https://firebase.google.com/docs/firestore. [Último acceso: Noviembre 2021].

[10] Google Developers, «Firebase,» Octubre 2021. [En línea]. Available: https://firebase.google.com/docs/functions. [Último acceso: Noviembre 2021].

[11] Google Developers, «Firebase,» Octubre 2021. [En línea]. Available: https://firebase.google.com/docs/auth. [Último acceso: Noviembre 2021].

[12] Affde, «Affde,» 2021. [En línea]. Available: https://www.affde.com/es/what-is-a-webhook.html. [Último acceso: Noviembre 2021].

[13] Zoho Corporation Pvt. Ltd, «Zoho,» 2021. [En línea]. Available: https://www.zoho.com/mail/help/dev-platform/webhook.html. [Último acceso: Noviembre 2021].

[14] OpenJS Foundation, «nodejs,» [En línea]. Available: https://nodejs.dev/learn. [Último acceso: Noviembre 2021].

[15] OpenJS Foundation, «nodejs,» [En línea]. Available: https://nodejs.org/en/download/. [Último acceso: Noviembre 2021].

[16] Facebook, Inc., «React Native,» 2021. [En línea]. Available: https://reactnative.dev/docs/typescript#getting-started-with-typescript. [Último acceso: Noviembre 2021].

[17] Microsoft, «Visual Studio Code,» 2021. [En línea]. Available: https://code.visualstudio.com/docs. [Último acceso: Noviembre 2021].

[18] Google Developers, «Android Developers,» 2021. [En línea]. Available: https://developer.android.com/studio/intro?hl=es-419. [Último acceso: Noviembre 2021].

[19] Facebook, Inc., «React Native,» 2021. [En línea]. Available: https://reactnative.dev/docs/environment-setup. [Último acceso: Noviembre 2021].

[20] L. Carvajal, Metodología de la Investgación Científica. Curso general y aplicado, 28 ed., Santiago de Cali: U.S.C., 2006, p. 139.