Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ingeniería en Ciencias y Sistemas

APOYO DE LA TECNOLOGÍA EN EL MANEJO DE LAS FINANZAS PERSONALES

Ariel Francisco Montejo Dominguez
Asesorado por el Ing. Herman Igor Véliz Linares

Guatemala, abril de 2016

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

APOYO DE LA TECNOLOGÍA EN EL MANEJO DE LAS FINANZAS PERSONALES

TRABAJO DE GRADUACIÓN

PRESENTADO A LA JUNTA DIRECTIVA DE LA
FACULTAD DE INGENIERÍA
POR

ARIEL FRANCISCO MONTEJO DOMINGUEZ
ASESORADO POR EL ING. HERMAN IGOR VÉLIZ LINARES

AL CONFERÍRSELE EL TÍTULO DE

INGENIERO EN CIENCIAS Y SISTEMAS

GUATEMALA, ABRIL DE 2016

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

NÓMINA DE JUNTA DIRECTIVA

DECANO	Ing. Pedro Antonio Aguilar Polanco
VOCAL I	Ing. Angel Roberto Sic García
VOCAL II	Ing. Pablo Christian de León Rodríguez
VOCAL III	Inga. Elvia Miriam Ruballos Samayoa
VOCAL IV	Br. Raúl Eduardo Ticún Córdova
VOCAL V	Br. Henry Fernando Duarte García
SECRETARIA	Inga. Lesbia Magalí Herrera López

TRIBUNAL QUE PRACTICÓ EL EXAMEN GENERAL PRIVADO

DECANO	Ing. Pedro Antonio Aguilar Polanco
EXAMINADOR	Ing. César Augusto Fernández Cáceres
EXAMINADOR	Ing. Edgar Estuardo Santos Sutuj
EXAMINADOR	Ing. Pedro Pablo Hernández Ramírez
SECRETARIA	Inga. Lesbia Magalí Herrera López

HONORABLE TRIBUNAL EXAMINADOR

En cumplimiento con los preceptos que establece la ley de la Universidad de San Carlos de Guatemala, presento a su consideración mi trabajo de graduación titulado:

APOYO DE LA TECNOLOGÍA EN EL MANEJO DE LAS FINANZAS PERSONALES

Tema que me fuera asignado por la Dirección de la Escuela de Ingeniería en Ciencias y Sistemas, con fecha agosto de 2015.

Ariel Francisco Montejo Dominguez

Guatemala, 21 de Enero de 2016

Ingeniero
Marlon Pérez Turk
Director
Escuela de Ciencias y Sistemas
Facultad de Ingeniería

Respetable Ingeniero Pérez Turk

Por este medio hago de su conocimiento que he revisado el trabajo de graduación del estudiante Ariel Francisco Montejo Dominguez con el número de carné 201020716, que lleva por título "**APOYO DE LA TECNOLOGÍA EN EL MANEJO DE LAS FINANZAS PERSONALES**" el cual cumple con los objetivos trazados para su elaboración según el protocolo presentado.

Sin otro particular.

Atentamente.

Ing. Herman Igor Véliz Linares
COLEGIADO No. 4836

Herman Igor Véliz Linares
Ingeniero en Ciencias y Sistemas

Universidad San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ingeniería en Ciencias y Sistemas

Guatemala, 27 de Enero de 2016

Ingeniero
Marlon Antonio Pérez Türk
Director de la Escuela de Ingeniería
En Ciencias y Sistemas

Respetable Ingeniero Pérez:

Por este medio hago de su conocimiento que he revisado el trabajo de graduación del estudiante **ARIEL FRANCISCO MONTEJO DOMINGUEZ** con carné **201020716**, titulado: **"APOYO DE LA TECNOLOGÍA EN EL MANEJO DE LAS FINANZAS PERSONALES"**, y a mi criterio el mismo cumple con los objetivos propuestos para su desarrollo, según el protocolo.

Al agradecer su atención a la presente, aprovecho la oportunidad para suscribirme,

Atentamente,

Ing. Carlos Alfredo Azurdia
Coordinador de Privados
y Revisión de Trabajos de Graduación

ESCUELA DE INGENIERÍA EN CIENCIAS Y SISTEMAS

UNIVERSIDAD DE SAN CARLOS
DE GUATEMALA

FACULTAD DE INGENIERÌA
ESCUELA DE INGENIERÍA EN
CIENCIAS Y SISTEMAS
TEL: 24767644

El Director de la Escuela de Ingeniería en Ciencias y Sistemas de la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala, luego de conocer el dictamen del asesor con el visto bueno del revisor y del Licenciado en Letras, del trabajo de graduación ***"APOYO DE LA TECNOLOGÍA EN EL MANEJO DE LAS FINANZAS PERSONALES"****, realizado por el estudiante ARIEL FRANCISCO MONTEJO DOMINGUEZ, aprueba el presente trabajo y solicita la autorización del mismo.*

"ID Y ENSEÑAD A TODOS"

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
DIRECCION DE INGENIERIA EN CIENCIAS Y SISTEMAS

Ing. Marlon Antonio Pérez Türk
Director
Escuela de Ingeniería en Ciencias y Sistemas

Guatemala, 07 de abril de 2016

Universidad de San Carlos
de Guatemala

Facultad de Ingeniería
Decanato

Ref.DTG.D.157.2016

El Decano de la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala, luego de conocer la aprobación por parte del Director de la Escuela de Ingeniería en Ciencias y Sistemas, al trabajo de graduación titulado: **APOYO DE LA TECNOLOGÍA EN EL MANEJO DE LAS FINANZAS PERSONALES,** presentado por el estudiante universitario: **Ariel Francisco Montejo Domínguez,** y después de haber culminado las revisiones previas bajo la responsabilidad de las instancias correspondientes, se autoriza la impresión del mismo.

IMPRÍMASE.

Ing. Pedro Antonio Aguilar Polanco
Decano

Guatemala, abril de 2016

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
DECANO
FACULTAD DE INGENIERIA

/cc

ACTO QUE DEDICO A:

Mis padres	Benjamín Montejo y Luz América Dominguez, por su apoyo durante todo este tiempo.
Mis hermanos	José, Luz y Rocio Montejo Dominguez, por ser una importante influencia en mi vida.

AGRADECIMIENTOS A:

Universidad de San Carlos de Guatemala	Por brindarme un lugar para estudiar, con calidad y valores.
Facultad de Ingeniería	Por transmitirme el conocimiento necesario y formarme como ingeniero.
Mis padres	Benjamin Montejo y América Domiguez, por su apoyo incondicional.
Ing. Herman Véliz	Por brindarme su apoyo como asesor.

ÍNDICE GENERAL

ÍNDICE DE ILUSTRACIONES

FIGURAS

LISTA DE SÍMBOLOS

Símbolo	Significado
Gb	Gigabyte
Kb	Kilobyte
Mb	Megabyte
%	Porcentaje
Q	Quetzal

GLOSARIO

Framework	Conjunto de herramientas orientadas a un lenguaje de desarrollo que potencian su uso.
Mockup	Diagrama o prototipo gráfico de un sistema informático.
IP	Dirección de internet asignada a una conexión.
REST	Estilo de arquitectura de software utilizado para sistemas distribuidos.
Lenguaje	Gramática formal que describe procesos que pueden ser ejecutados por un sistema informático.
Wifi	Internet de acceso inalámbrico.
Host	Nombre definido para reemplazar una dirección IP.
Sistema operativo	Programa que administra el hardware y software de un sistema computacional.
Web service	Servicio de información que puede ser accedido por medio del internet.

RESUMEN

La administración personal, como tal, incluye un conjunto de prácticas que, si se realizan, se tiene una mayor probabilidad de alcanzar una mayor estabilidad financiera después de cierto periodo, comúnmente a fin de mes o de cada quincena. Las prácticas que se implementan pueden pasar de ser teorías a ser buenas prácticas como tal, entre las cuales están no gastar de más, ahorrar, tener un buen control de las deudas y saber medirse en qué es necesario gastar y en que no.

La construcción de una aplicación que apoye este proceso contempla las categorías anteriores, considerando la funcionalidad que representa cada una, ya que el manejo de la información no es igual en todas.

Luego del análisis de las funcionalidades a implementar, se crea el modelo de datos capaz de almacenar la lógica planteada en un principio, para luego pasar a la parte del diseño de la interfaz que debe procurar ser lo más sencilla posible y evitar que el usuario tenga que dar muchos pasos en ella, logrando así una mayor aceptación.

La última parte a tomar en cuenta es la presentación de la información, ya que es la parte que más le interesa al usuario, pues en esta se puede dar cuenta de todo en un simple resumen. En esta parte se consideran todos los reportes a desplegar, ya sean tablas o gráficos.

OBJETIVOS

General

Proporcionar una herramienta de software para ayudar a las personas con el control de sus gastos personales, atacando principalmente la falta de planeación y el gasto inconsciente.

Específicos

1. Facilitar el control de gastos.

2. Mejorar la cultura de ahorro.

3. Mejorar el control de ingresos.

4. Evidenciar la mejora en las finanzas, mediante la utilización de presupuestos.

INTRODUCCIÓN

En el mundo laboral existe una gran cantidad de jóvenes que entran a trabajar por primera vez por distintas razones. Normalmente a esta edad no se tiene mucha instrucción financiera y es muy común verlos gastar su dinero en artículos que antes no tenían, como un nuevo teléfono, ropa, salidas, entre otros; sin tomar en cuenta que desde temprana edad debe irse desarrollando la buena costumbre financiera, aprender a gastar lo necesario y un poco extra, si el sueldo lo permite, además, una de las cosas más importantes, ahorrar.

Sin embargo, esto se percibe en varias poblaciones laborales, no únicamente en los jóvenes, ya que se puede ver a gente mayor caer en las mismas malas prácticas, sin tener un control de su dinero, sin saber cuánto gastan, cuánto ganan o cuánto están ahorrando realmente.

Tomando en cuenta la creciente aceptación de la tecnología en las distintas poblaciones y que en la juventud está presente casi en un 100 %, realizar una aplicación que apoye, tanto en la concientización de los gastos como en el manejo de los presupuestos, con la que se pueda ayudar a mejorar la cultura financiera. Esto puede beneficiar a las personas, brindando la herramienta necesaria en la actualidad, que se ajusta a las necesidades y posibilidades existentes.

Por lo tanto, se presenta el proyecto para desarrollar una aplicación que ayude a las personas en los aspectos básicos de la administración personal, controlando gastos y creando presupuestos.

1. REQUERIMIENTOS

Todo buen proyecto de software debe iniciar con un buen proceso de toma de requerimientos, asegurando así estar más cerca de lo que el usuario busca o necesita. Esto, traducido a la metodología TAM, puede verse como asegurar la utilidad percibida por el usuario, ya que con una buena toma de requerimientos se puede garantizar una aplicación que haga lo que el usuario necesita.

1.1. Requisitos funcionales

Los requisitos funcionales son las opciones solicitadas por los usuarios, es decir, lo que el usuario necesita que haga la aplicación. Estos requisitos están orientados a los puntos en donde se debe enfocar la administración de las finanzas personales, ya que estos aspectos son los principales a cubrir para tener un verdadero balance financiero a nivel individual.

Uno de los puntos en las finanzas personales es tener en cuenta tanto los gastos como los ingresos, ya que la ecuación más simple para estar solvente son los ingresos menos los egresos, fijos o variables. Pero agregado a esto, existen otras variables que deben considerarse para llegar con solvencia a fin de mes, cada una de estas debe traducirse en una funcionalidad del sistema, ya que esto agrega más valor para el usuario final.

1.1.1. Control de ingresos

Los ingresos pueden ser tanto fijos como variables, dependiendo de la forma en que el usuario sea remunerado en su trabajo.

Por lo tanto, establecer un sueldo inicial se vuelve complicado para las personas que no tienen un ingreso establecido. Si bien puede colocarse el mínimo esperado, como es recomendado, agregar otros ingresos durante el mes ayudará a tener un mejor control de las cuentas, no solo por tenerlo en cuenta para los gastos, sino para saber realmente cuanto se ha ganado. Por lo tanto, un módulo que permita agregar los ingresos que se tienen durante el mes se vuelve vital para el proceso.

1.1.2. Control de gastos

Muchas veces, los gastos suelen superar a los ingresos, esta es la causa principal de que se tengan problemas financieros. Una de las principales razones de esto es el gasto inconsciente, no se sabe si se está gastando de más, cuánto ya se ha gastado ese día o simplemente no se tiene en mente cuanto se debe gastar porque no se ha tomado en cuenta este tema. Teniendo esto en mente, el control de gastos intuitivamente debe ir incluido en la aplicación.

Existe otro tema a tratar con los gastos, este aparece cuando surge la duda de ¿era necesario gastar en esto? Al tener una buena cultura financiera, debe tomarse en cuenta la holgura del presupuesto, si hay un buen margen de ingresos pueden permitirse gastos que no sean necesarios o vitales, ya que estos no deben evitarse totalmente; si no existe tal margen, se deben limitar los gastos a lo estrictamente necesario. Como parte de este tema y a razón de un *feedback*, se puede incluir una sección de gastos necesarios y no necesarios, para que se sepa qué tan balanceados se está en este tema.

1.1.3. Presupuestos

No se puede hablar de administración de finanzas sin tener en cuenta el tema de presupuestos. Los presupuestos ayudan a distribuir los gastos, ya que no se gastará lo mismo en alimentación que en transporte o en medicamentos.

La forma de traducir los presupuestos a una funcionalidad de la aplicación se realiza con el uso de categorías o segmentos de gastos, con los que se pueden agrupar y definir cuánto es el máximo que se debe invertir en cada uno. A cada categoría debe poder asignársele una cantidad máxima de gastos por mes, así como una cantidad de alerta, que indique cuando se esté próximo a alcanzarla.

Un punto importante es, si los ingresos suben, ¿aumenta la cantidad destinada para cada categoría? Esto es una decisión personal, por lo cual debe poder asignarse el presupuesto por categoría, ya sea en cantidades netas o en porcentajes del total del presupuesto, para que cuando los ingresos suban no se deba ir a cada categoría a modificarlos, pues estos se adaptarán a la cantidad nueva gracias al porcentaje.

1.1.4. Deudas

Las deudas pueden no siempre contraerse por falta de solvencia, si se adquiere un préstamo, se compra algo a pagos o simplemente no se tenía efectivo en cierto momento y se tuvo que prestar. Las deudas como esta última pueden solventarse en un solo pago, por lo que se pueden incluir como gastos. El otro tipo de deudas, que son a mediano y largo plazo, en cambio, deben planificarse para poder tenerlas en cuenta dentro del presupuesto.

Por esto, en lugar de incluir las deudas como un gasto más, es conveniente tenerlas por separado, estableciendo distintas deudas, fechas de pago y montos, para que nunca se olvide pagar. Además, cada pago realizado puede incluirse en alguna de las categorías antes mencionadas, para que se clasifiquen como las demás salidas de dinero.

1.1.5. Ahorros

En una buena cultura financiera, se debe ahorrar un porcentaje de lo que se gana. Esto suele volverse difícil cuando no se tiene planeada una cantidad o simplemente no está dentro de los hábitos. Estos ahorros pueden ser una cantidad que se guarda en el banco o en cualquier otro lugar, pero está destinada a no gastarse hasta cierto momento, ya sea en unas vacaciones, un viaje o algo que se quiera comprar.

Tomando esto en cuenta, probablemente no se desea ahorrar una gran cantidad a inicio de mes o cuando hay ingresos, pero se puede ir ahorrando una pequeña parte cada cierto tiempo, hasta llegar a una cantidad establecida. Por lo tanto en la aplicación es funcional agregar una meta de ahorros, que indique qué tan lejos se está de la meta y recuerde que debemos ahorrar.

1.1.6. Balance general del mes

Teniendo administrado todo lo que puede surgir en el mes, es necesario saber si se cumplió con las metas, por lo que debe existir un balance general de fin de mes, a modo de resumen, que indique en qué se gastó realmente. Si se tuvieron los ingresos que se esperaban, se pagaron todas las deudas, se ahorró lo que se debía ahorrar y tener un panorama general del mes.

Este balance debe indicar que tanto se apegó a lo planificado, dependiendo de los límites que se establecieron para los gastos, con esto se podrá saber cómo van las finanzas.

1.1.7. Seguridad de la información

En la actualidad, el tema de la seguridad de la información es muy importante, dado el valor de la misma. Teniendo en cuenta que los datos que se manejarán son datos financieros, debe asegurarse que esta información sea accesible solo para el usuario que la ingresó. Por esta razón, la conexión de la aplicación que utiliza el usuario con el servidor donde es almacenada la información debe ser segura y evitar fugas de información.

1.2. Requisitos no funcionales

Son los requisitos que no se solicitan explícitamente por el usuario, pero que son necesarios para el buen desenvolvimiento de la aplicación. Entre estos se encuentran restricciones, tiempos de respuesta o especificaciones técnicas que intervendrán en el uso diario de la aplicación.

1.2.1. Accesibilidad

La información que se maneje en la aplicación debe poder migrarse de un dispositivo a otro, ya que, si por algún motivo se cambia de dispositivo, habría que ingresar todo nuevamente. Esto causaría una gran incomodidad a los usuarios, por lo que la información debe poder trasladarse entre dispositivos, a través de un servidor en el que se iniciará la sesión desde cada dispositivo.

1.2.2. Tiempo de respuesta

Debido a que la aplicación debe conectarse con un servidor, será necesario el uso del internet en todo momento. Por esto debe asegurarse que, aún en conexiones lentas, se tenga un tiempo de respuesta lo suficientemente rápido para no desesperar al usuario.

El tiempo planteado es de no más de 5 segundos por transacción, para que el usuario no tome la aplicación como lenta y pierda el interés en usarla. Por lo tanto, la aplicación debe optimizar la conexión con el servidor, la información que se envía y cómo se envía, para evitar que pueda ser lenta la comunicación.

1.2.3. Restricción técnica

La aplicación debe ser diseñada para el sistema operativo Android, a partir de la versión 4.0.0 en adelante, no existe otra restricción técnica en el desarrollo de la aplicación.

1.3. Requisitos de integración

Debido a que la aplicación no debe integrarse con ningún otro sistema ni empresa, no existen requisitos de integración que deban tomarse en cuenta durante el diseño y desarrollo de la misma.

2. DISEÑO

El diseño de la aplicación es la parte donde se debe definir cómo se construirá la aplicación, en esta parte se deben incluir todos los requerimientos definidos anteriormente, combinándolos con la tecnología que se utilizará.

En esta parte es muy importante tomar en cuenta que la aplicación debe ser fácil de utilizar para el usuario, complementando así la metodología TAM, para obtener una mayor probabilidad de aceptación del software. Aunque esta sea totalmente funcional a las necesidades del usuario, si no es fácil de usar, no habrá interés por parte del usuario de seguirla utilizando.

2.1. Metodología TAM

El modelo de aceptación tecnológica es un modelo en el que se intenta describir la forma en que los usuarios aceptan y usan una nueva tecnología, enfocado en distintos factores, los cuales dependen de la nueva tecnología en sí, así como factores culturales, ideológicos, demográficos, entre otros.

Esencialmente, son tres partes las que componen esta metodología: utilidad percibida, facilidad de uso y actitud hacia el uso.

2.1.1. Utilidad percibida

Cuando una nueva tecnología es creada, debe centrarse en resolver una necesidad, entendiendo esto como cualquier forma de mejorar la vida de las personas.

Así, serán las mismas personas las que quieran hacer uso de esta nueva tecnología, de esta forma se puede tener un mayor grado de aceptación al crear una nueva tecnología. Esto puede realizarse a través de muchas formas, estudios de mercado, *focus groups*, encuestas u otro medio para conocer la opinión de un grupo de personas.

2.1.2. Facilidad de uso

Cuando una necesidad es ubicada, pueden o no existir varias soluciones para la misma. En este caso, la mayor ventaja la obtendrá la que sea más fácil de utilizar. Para tener esta ventaja, el análisis y el diseño de la nueva tecnología debe tomarse con dedicación, para crear una solución sencilla, fácil de utilizar y que sea amigable con el usuario final.

2.1.3. Actitud hacia el uso

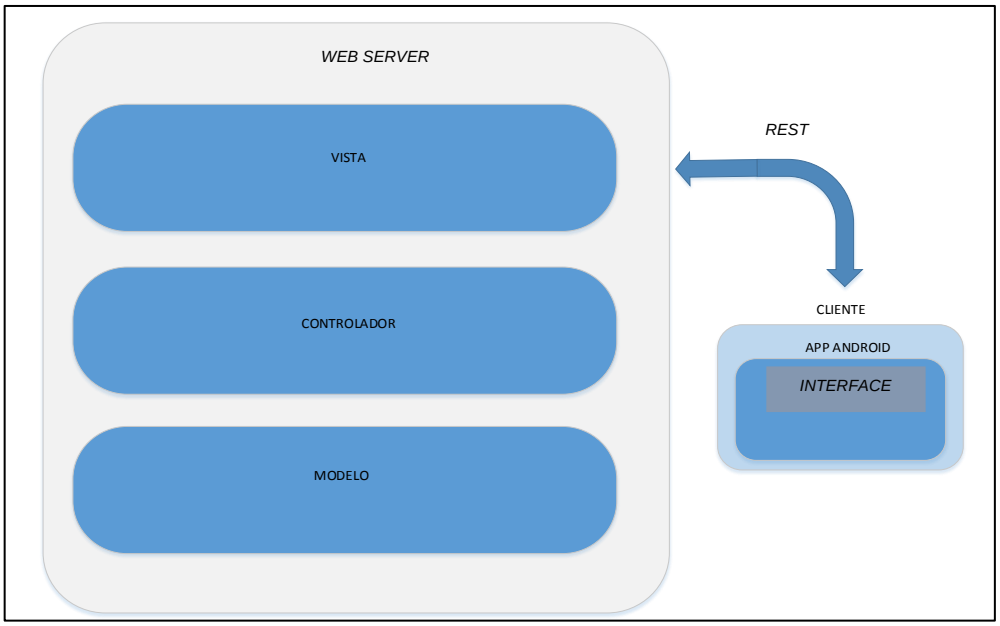
En la actitud hacia el uso, se encuentran los diversos factores que varían de persona a persona, dependiendo de la cultura, región, edad, religión, entre otros. Por lo tanto, las tecnologías deben adaptarse, en la manera que sea posible, a los grupos que se identifiquen para tener una mejor adaptación y aceptación de la tecnología.

2.2. Arquitectura del software

La arquitectura propuesta para la solución de software combina dos arquitecturas bastante utilizadas, las cuales hacen que la aplicación sea potente, segura y escalable, por lo que es capaz de crecer en términos de cantidad de usuarios y en funciones de la aplicación.

Estas arquitecturas son una combinación de la ya conocida cliente-servidor con la más reciente MVC (modelo vista controlador), están comunicadas a través de una API REST cifrada que provee la comunicación y seguridad necesarias para que la aplicación pueda cumplir sus funciones de la mejor manera.

Figura 1. **Arquitectura general**



Fuente: elaboración propia.

2.2.1. Cliente-servidor

La arquitectura cliente-servidor divide el funcionamiento de la aplicación en dos secciones, una en donde se desarrolla toda la lógica del negocio y se almacenan los datos. En este caso, el servidor donde se almacenan y manejan los datos, se recibe y envía la información desde y hacia cada uno de los clientes.

El servidor está alojado en un servidor web, al cual se accede a través de internet. El cliente es cada dispositivo móvil utilizado por los usuarios, la conexión a este se hace a través de REST, lo que indica que el servidor no tiene estados según las conexiones anteriores que se han realizado y en cada conexión hay que autenticarse e indicar la identidad.

En esta parte es donde se vuelve importante la seguridad, ya que la identidad del usuario debe ser validada cada vez por el servidor, por lo que el cliente implementa credenciales y encriptación para evitar fugas de información.

Del lado del cliente, los datos obtenidos desde el servidor se presentan en la vista, pudiendo el usuario interactuar con la información. Esta también se encarga de recoger los datos que el usuario provee, almacenándolos temporalmente en memoria y posteriormente enviándolos al servidor para su almacenamiento permanente.

2.2.2. MVC

El MVC, o modelo vista controlador de reciente auge, permite separar las partes de la aplicación en varias capas para mejorar la seguridad y la interacción entre estas, además de proveer una estructura bastante ordenada, que permite separar la interfaz, la lógica del negocio y el modelo de datos.

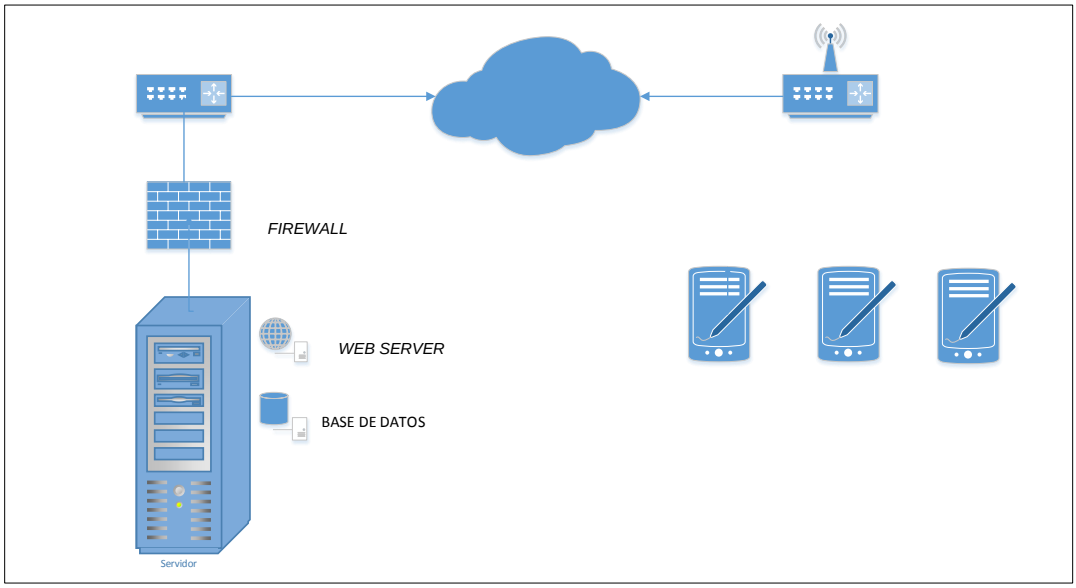
Esta arquitectura está únicamente del lado del servidor web, en donde se aloja la base de datos, la cual interactuara con el modelo. La capa de los controladores funge como puente entre la capa de vista y el modelo, ya que recibe información y solicitudes desde la vista, y, apoyándose en el modelo, procesa la información según las acciones definidas por la lógica del negocio.

La vista es la encargada de interactuar con las peticiones de la API REST, ya que esta recibe las peticiones y parámetros, y los envía al controlador, el que posteriormente da una respuesta con parámetros de salida a través de la vista nuevamente. Esta no está en contacto directo con el modelo, solo puede acceder a él a través de las acciones definidas por los controladores.

2.2.3. Diagrama de despliegue

La aplicación está soportada por una capa de software, que ya fue descrita anteriormente. Además, se cuenta con el soporte del hardware y la distribución de este que debe existir para tener un servicio funcional y que la aplicación cumpla su cometido. En este caso, el usuario solo necesita de un dispositivo móvil y acceso a internet, el resto es utilizado por parte del servidor y es totalmente invisible al usuario.

Figura 2. **Diagrama de despliegue**



Fuente: elaboración propia.

Como se observa en la figura 2, el usuario se conecta desde su dispositivo móvil a una red de internet cualquiera, ya sea desde su proveedor de datos o a una red wifi, a través de esta conexión, soportada por toda la infraestructura de red existente para el internet (la cual se obvió en la figura 2), se permite la comunicación con el servidor, en el cual están alojados los distintos servicios, como el servidor web o el servidor de base de datos.

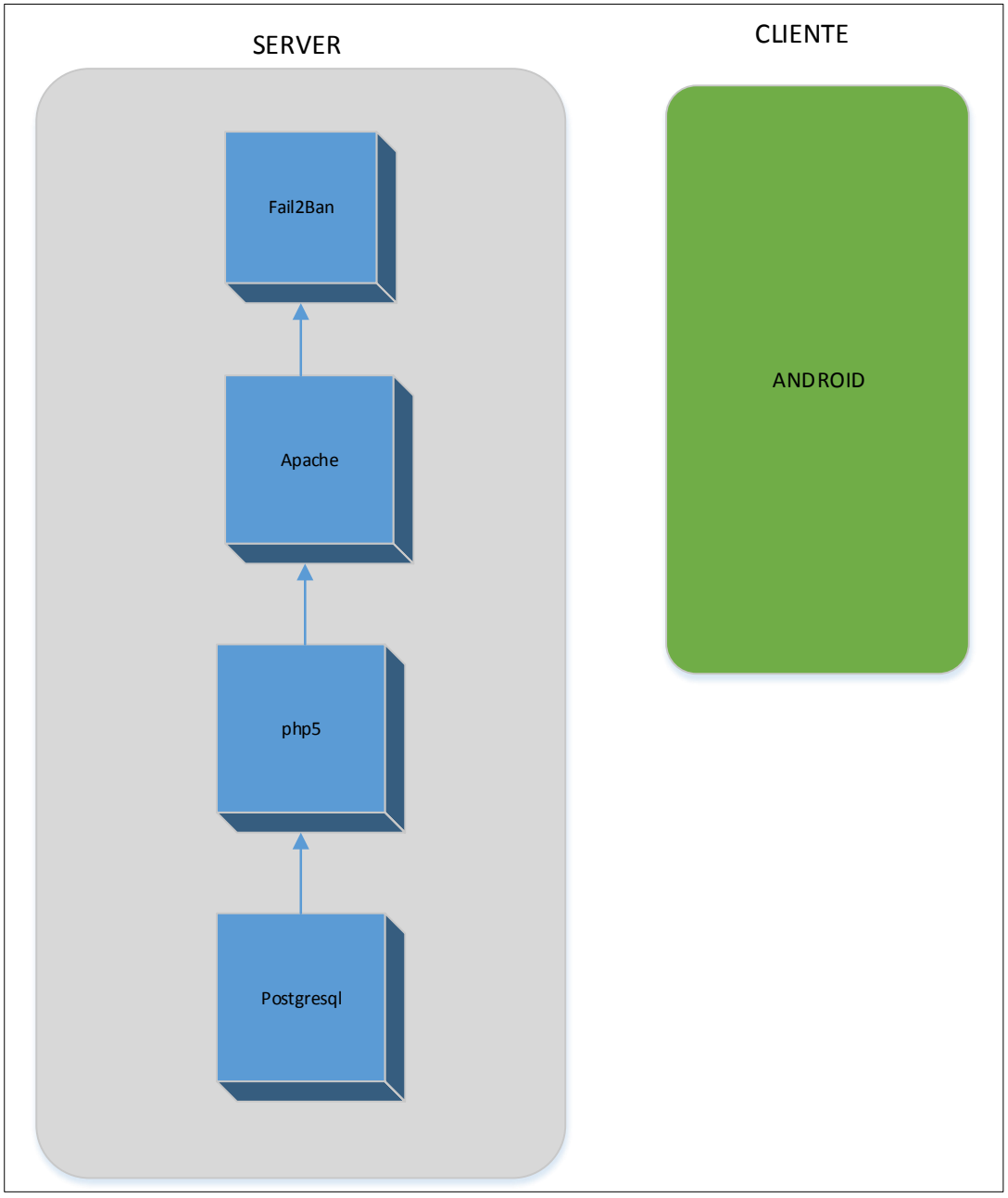
La mayor parte de la arquitectura física necesitada consiste en la red de internet, la cual es independiente tanto para el usuario como para el servidor, por lo que para el usuario le basta con tener conexión a internet.

2.3. Tecnologías a utilizar

Luego de conocer cuál será la arquitectura de la aplicación y de saber cómo estará distribuida físicamente, es necesario conocer qué tecnologías se utilizarán para llevar a cabo el desarrollo de la aplicación. Para seleccionar estas tecnologías se tomaron en cuenta los requerimientos y se analizó cual es la mejor solución, adaptando las tecnologías con la arquitectura.

De esta forma se obtiene una aplicación de fácil mantenimiento, pues esta combinación permitió seleccionar tanto lenguajes como *frameworks* de uso actual, lo que posibilita tener mucha documentación y bastante mantenimiento en las herramientas que se utilizarán, pues estas son constantemente actualizadas.

Figura 3. **Tecnologías a utilizar**



Fuente: elaboración propia.

2.3.1. Fail2Ban

Consiste en un *script* escrito en Python, encargado de escanear los *logs* de acceso de Apache. Este sirve para proteger el servidor de ataques DOS, que en la actualidad son muy comunes de escuchar. Estos pueden ocasionar la caída del servidor y, por ende, la caída de los servicios, lo cual ocasiona inconvenientes a los usuarios provocando que la aplicación deje de parecerles estable y decidan no utilizarla.

Los ataques DOS consisten en hacer una gran cantidad de solicitudes a páginas inexistentes, obteniendo errores de páginas no encontradas. Se genera una cantidad de solicitudes tal que el servidor web llega a su límite y se detiene, denegando el servicio a los usuarios legítimos.

Con este software se escanean constantemente los *logs* de Apache, que es el servidor web utilizado en este caso, detecta las IP que han incurrido en este tipo de solicitudes a páginas inexistentes recurrentemente y les niega el servicio a través de listas de acceso.

De esta forma, estas IP quedan bloqueadas temporalmente, por lo que no saturan el servidor web y de esta forma se protegen del ataque.

2.3.2. Apache *web server*

Apache es uno de los *web servers* más populares. Este servidor se ha popularizado por su potencia y además por ser multiplataforma, de fácil instalación y con un conjunto de herramientas que dan soporte a una gran cantidad de opciones que se deseen tener al momento de publicar una aplicación. Soporta el uso de los *virtual hosts*, que son utilizados para el

desarrollo de esta aplicación como requerimiento del *framework* de desarrollo utilizado.

Los *virtual hosts* permiten tener distintos nombres de *host* en uno mismo. Además, el *framework* de desarrollo de la aplicación lo necesita para su correcto funcionamiento.

2.3.3. Php5 Zend Framework

Php es uno de los más potentes lenguajes para desarrollo web, además de uno de los más populares. Con su uso básico se pueden crear fácilmente páginas dinámicas, combinándolo con un *framework* se pueden obtener beneficios más grandes, ya que con librerías de funciones y una amplia gama de opciones que están listas para utilizar, se ahorra mucho tiempo de desarrollo, porque solo se debe dedicar tiempo a la construcción de la aplicación y a alguna función que no se encuentre dentro de las librerías proporcionadas por los *frameworks*.

Zend Framework es un popular *framework* de desarrollo de aplicaciones basado en MVC, provee toda la base de funciones para mantener separados vistas, controladores y modelos. Provee una compatibilidad bastante amplia con las bases de datos, por lo que desde este se pueden manejar de forma sencilla las mismas con las librerías ya creadas, proporcionando únicamente los datos de conexión.

Para el desarrollo de la aplicación se utilizó Zend Framework, para la implementación de los *web services*, los cuales serán consumidos por los clientes desde los dispositivos móviles. Desde estos se reciben y contestan las peticiones, se maneja la lógica de los cálculos y se controla el almacenamiento de datos, todo esto implementando una capa de seguridad general donde se valida la identidad de los usuarios, para evitar fugas de información.

2.3.4. Postgresql

El motor de base de datos seleccionado es Postgresql. Esta es una base de datos que se ha popularizado mucho últimamente, debido a su alto rendimiento, que es *open source* y que cuenta con una gran comunidad la cual constantemente está trabajando en la mejora de la misma; gracias a esto también cuenta con una muy amplia documentación.

Esta base de datos relacional provee una gran confiabilidad y seguridad de los datos, construyendo un modelo de datos apropiado se puede obtener la mejor funcionalidad de la misma, logrando así que la aplicación funcione de una manera más óptima, ya que normalmente la base de datos suele ser uno de los eslabones de la ruta crítica en cuanto a rendimiento.

Para trabajar en conjunto con php, debe agregarse el paquete php5-pgsql, que es el controlador de Postgres para php, así estos podrán comunicarse y se podrá establecer la conexión.

2.3.5. Android

Es actualmente el sistema operativo más popular para celulares, cubriendo la mayoría del mercado, superando a otros sistemas como Windows Phone, lo que hace del campo del desarrollo de aplicaciones para este sistema una buena opción para desenvolverse, ya que cuenta con millones de usuarios.

Por este motivo, se seleccionó este sistema para el inicio del desarrollo de la aplicación de control de gastos, ya que brinda una buena base de usuarios para las versiones, iniciales pudiendo así obtener retroalimentación sobre el uso de la misma, además, se alcanzará a más personas según sea la aceptación que tenga la aplicación.

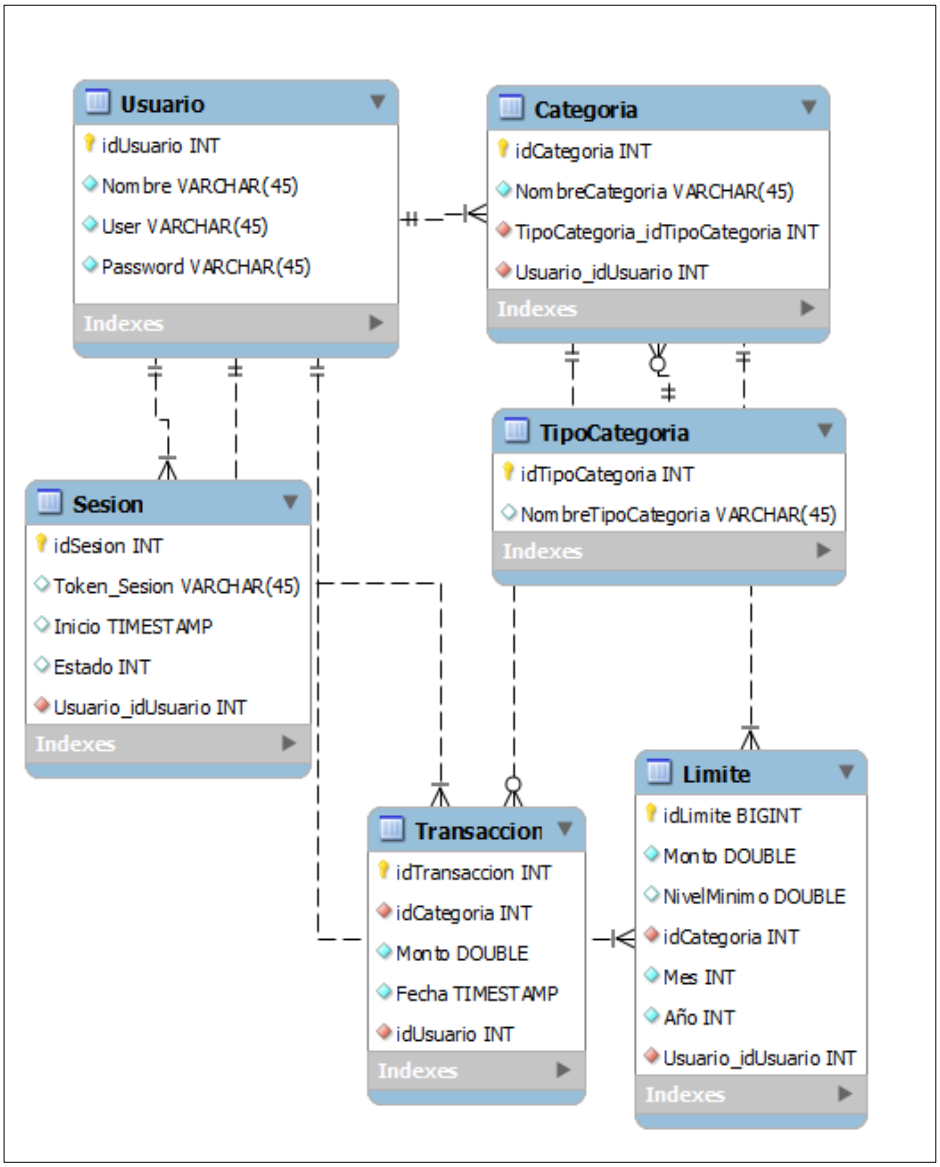
Se estableció una versión mínima del sistema operativo de 4.0.0, para tener funciones que no están presentes en otras versiones y que son necesarias para el buen desempeño de la aplicación. La versión 4.x.x es de las últimas versiones del sistema operativo, muy popular y de las más usadas por los proveedores de celulares, ya que la versión anterior 2.x.x ha quedado obsoleta y se tiene más posibilidad de usuarios con la versión más reciente.

2.4. Modelo de datos

Se diseñó un modelo de datos relacional, utilizando la normalización de la base de datos para tener un esquema óptimo y fácilmente escalable, para futuras herramientas que se puedan diseñar. Para el diseño de la base de datos se utilizó la herramienta Workbench, que sirve para crear modelos entidad relación y luego se exportan a un lenguaje de base de datos específico para crear la base de datos por medio del *script* generado.

Para este modelo se utilizaron las tablas relacionadas con el alcance preliminar de aplicación, es decir, se creó solo lo funcionalmente necesario para que el programa se desempeñe dentro de los alcances establecidos.

Figura 4. **Modelo de datos**



Fuente: elaboración propia.

2.4.1. Usuario

Esta tabla contiene la información básica del usuario, solo se solicitaron los datos básicos como inicio de la misma. Los datos recolectados son el nombre, el usuario y la contraseña para ingresar.

2.4.2. Sesión

En esta tabla se almacenan los datos de las nuevas solicitudes del usuario, al ingresar una nueva solicitud el sistema verifica si cuenta con una sesión, de lo contrario se generará una nueva. Esta sesión se identifica con un *token* que se obtiene de forma aleatoria para el usuario, este se envía de vuelta al usuario para poder identificarlo en cada solicitud que realice. Cada *token* cuenta con un tiempo de vida, el cual es controlado por el servidor. Al cumplirse este tiempo de vida, el sistema automáticamente gestiona un nuevo *token*, dando inicio a un nuevo ciclo idéntico al anterior. La técnica de utilización de tiempos de vida es ampliamente utilizada en sistemas de seguridad, pues brinda una sólida protección frente a intrusos.

Al realizar el manejo de la seguridad mediante un *token*, se hace necesaria la utilización de un código que permita la comunicación entre el servidor y la aplicación. Este código le permite a la aplicación identificar las distintas respuestas del servidor; cada respuesta está asociada a un identificador numérico que es conocido por ambas partes. Al recibir un nuevo código de respuesta la aplicación es capaz de decidir qué acción debe realizar.

2.4.3. Tipo de categoría

Como se observa en los requerimientos, se necesitan categorías, no solo para catalogar los gastos, sino los otros tipos de destinos del dinero como ingresos, ahorros o deudas. Las categorías pueden pertenecer a algún área distinta, por lo que esta tabla contiene la descripción de estos tipos, para asociarlas al área a la que pertenecen.

2.4.4. Categoría

Las categorías son utilizadas para catalogar en cada área los gastos, esto para tener un mejor detalle de en qué se ha invertido el dinero. Para esto, cada categoría es asociada a un área, como se describió anteriormente, también están asociados a un usuario en específico, ya que cada usuario puede personalizar sus propias categorías. Estas categorías se muestran solo para el usuario que las creó.

2.4.5. Límite

Para el manejo de los presupuestos se debe contar con límites establecidos de gastos que deberán ser respetados durante el periodo, en este caso el mes. Estos límites son establecidos por mes y año, asociados a una categoría que, a su vez, es propiedad de un usuario. Así, cada usuario tiene sus propios límites, estos son aplicables solo a las categorías de gastos, por lo que en la asignación de los límites solo deben aparecer las categorías de dicha área.

2.4.6. Transacción

En esta tabla se almacenan todos los movimientos que el usuario registra, cada transacción va asociada a una categoría, un usuario y un límite. El monto gastado de cada categoría se calcula a partir de todas las transacciones que se hayan hecho durante el mes del periodo, estos son los que se comparan con el límite de las mismas.

Cada transacción cuenta con un detalle donde se puede especificar la razón del gasto, para tener un detalle en específico, esto también aplica a las deudas y los ingresos, ya que en estos también se puede especificar la razón de la transacción.

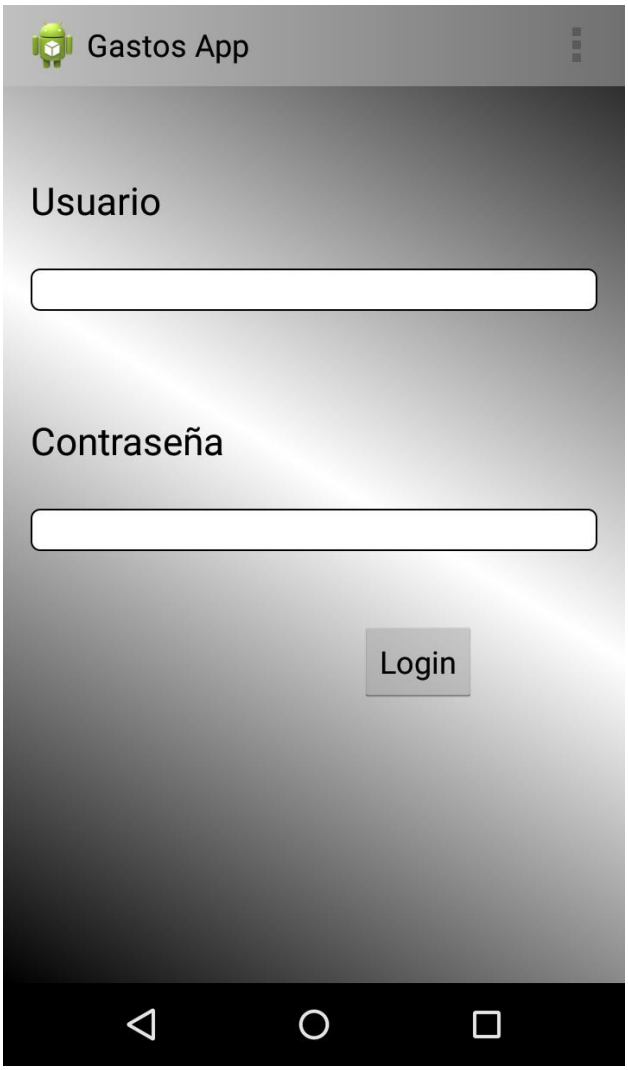
2.5. *Mockups*

Son diseños básicos y de baja calidad que se utilizan para tener un canal de comunicación claro con el cliente, pues permiten transmitir ideas de forma sencilla.

2.5.1. *Login*

La pantalla de inicio cuenta con un campo para el nombre de usuario y uno para la contraseña, esta es la pantalla inicial de la aplicación, no se puede pasar de ella sin autenticarse.

Figura 5. ***Login***

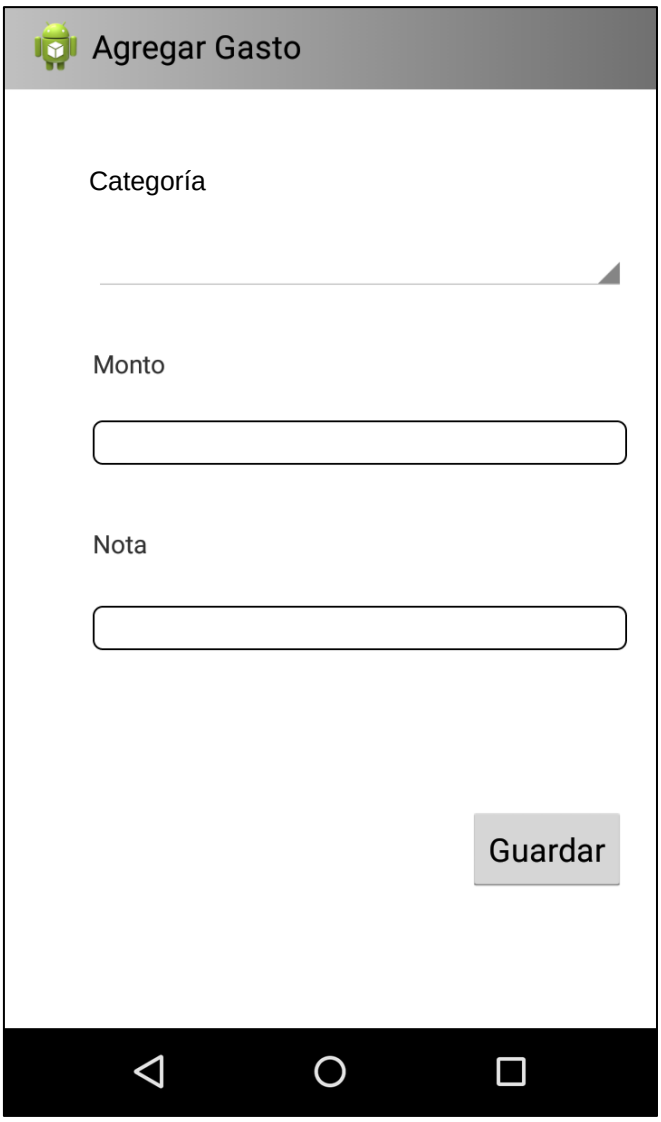


Fuente: elaboración propia.

2.5.2. Agregar gasto

Luego de autenticarse, el usuario es dirigido a la pantalla inicial de la aplicación. Esta pantalla es la de agregar gasto, ya que es el uso más común que se le dará a la misma.

Figura 6. **Crear gasto**

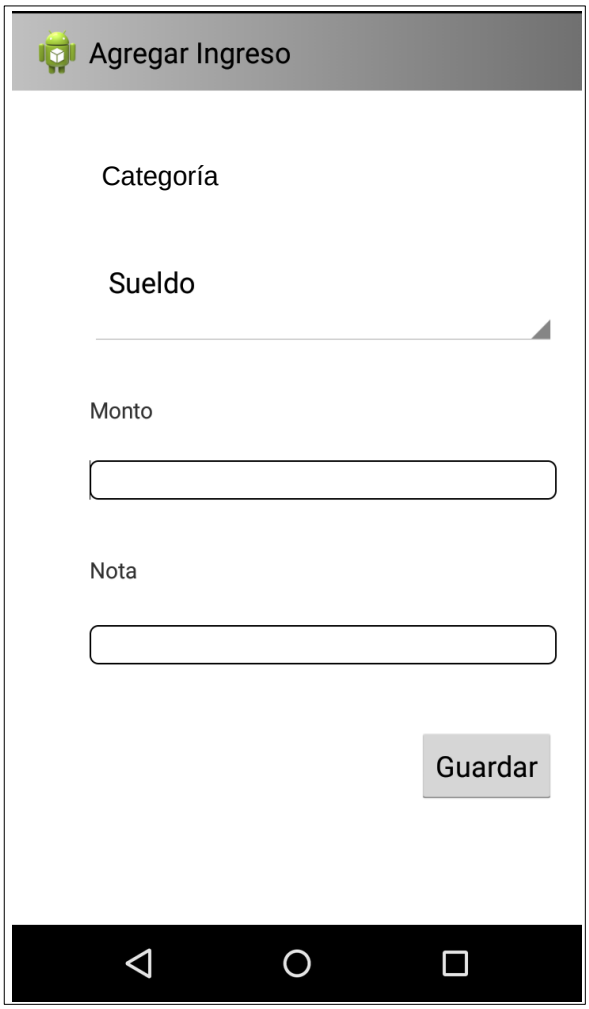


Fuente: elaboración propia.

2.5.3. Crear ingreso

Así como se pueden crear gastos, se pueden crear ingresos, la vista es prácticamente la misma, pero se guarda como un área distinta.

Figura 7. **Crear ingreso**

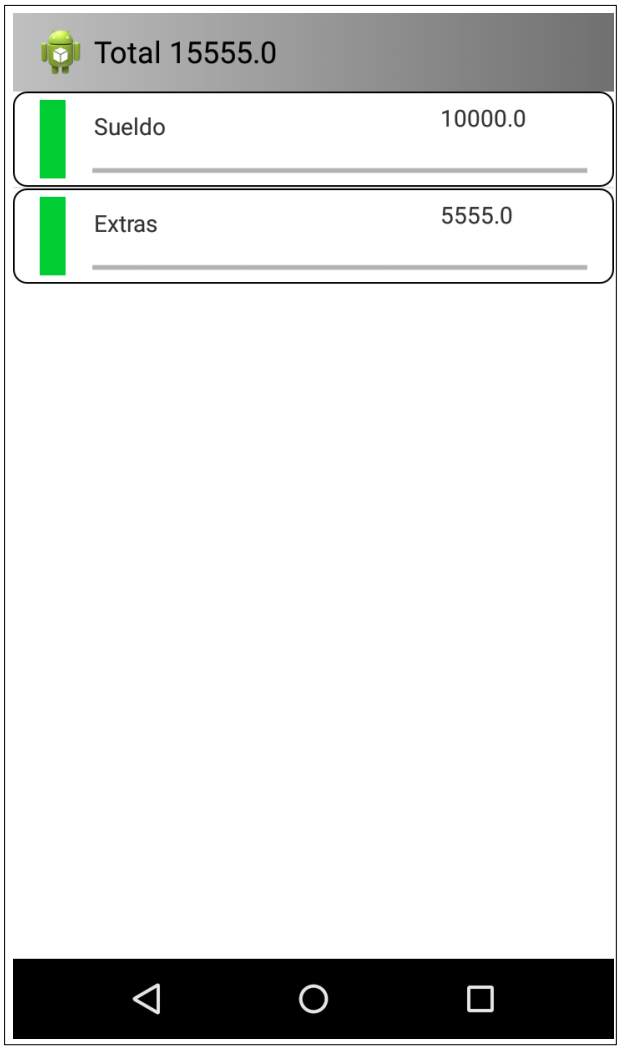


Fuente: elaboración propia.

2.5.4. Ingresos

En esta pantalla se observa el resumen por categoría de todas las transacciones realizadas, también se observa el total general de todos los ingresos registrados. Cada categoría puede ser presionada para ver el detalle de transacciones de la misma.

Figura 8. **Ingresos**



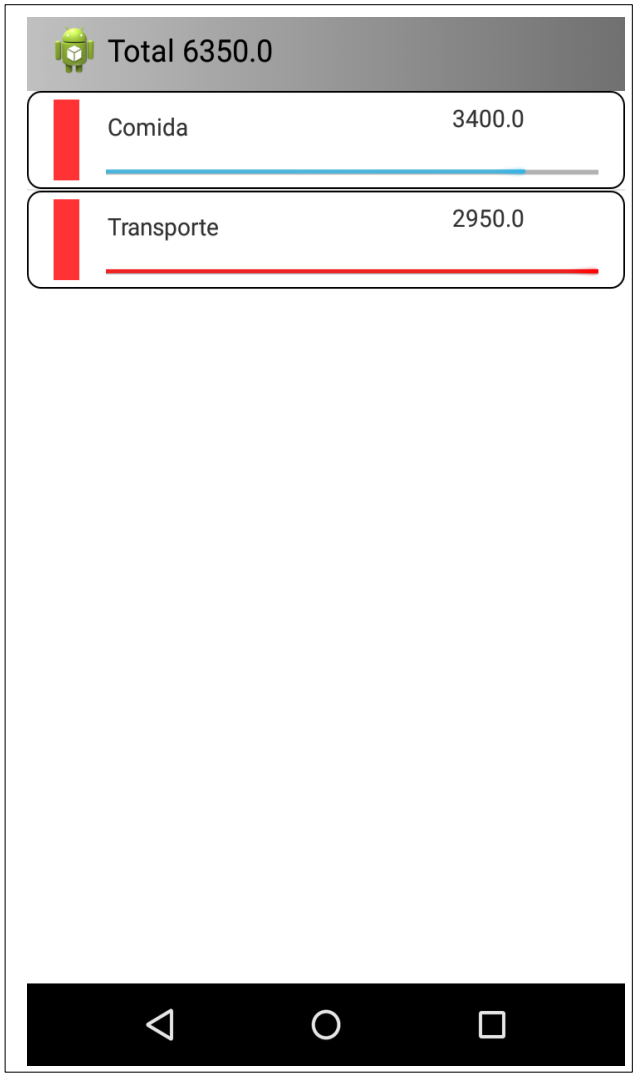
Fuente: elaboración propia.

2.5.5. Egresos

Al igual que la anterior, en esta pantalla se observa el resumen de los gastos, agrupados por categoría y el total general, al presionar sobre cada categoría puede observarse su detalle. Cada categoría cuenta en el detalle con

el total gastado, el límite establecido y cuánto queda de margen para llegar a dicho límite.

Figura 9. **Gastos**



Fuente: elaboración propia.

2.5.6. Presupuesto

En esta pantalla pueden definirse los límites por cada categoría de gastos, para llevar el control de cuánto dinero se ha gastado en realidad. Cada presupuesto corresponde a una categoría por mes, por lo que estos deberán establecerse cada mes nuevamente.

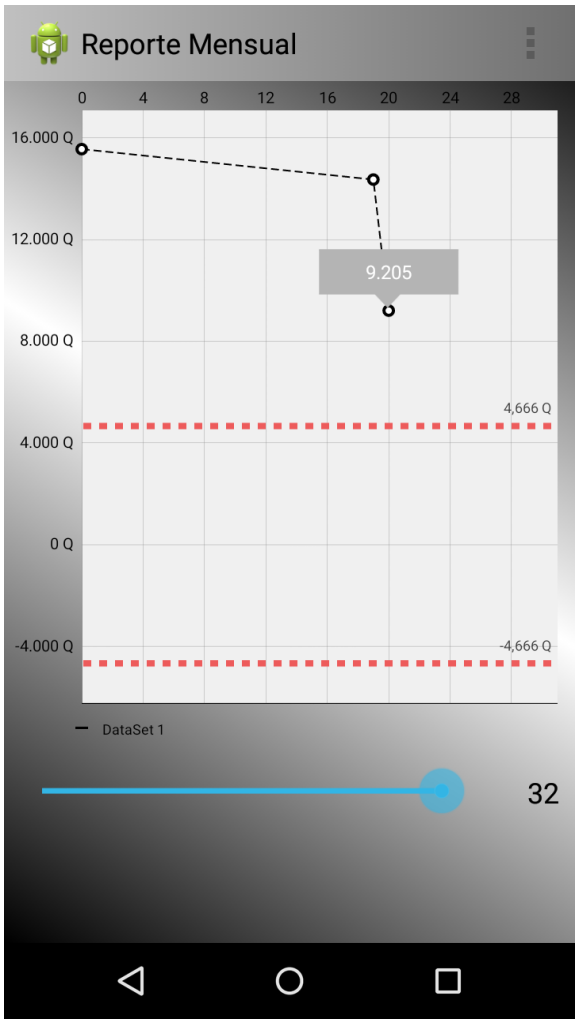
Figura 10. **Presupuestos**

Fuente: elaboración propia.

2.5.7. Reporte lineal por mes

En esta pantalla se observa cómo han ido surgiendo los gastos según el avance del mes, descontando del total disponible los gastos que se van dando por día. Esta gráfica cuenta con una línea de alerta, la cual indica el 30 % del total para tener un control del estado financiero.

Figura 11. **Reporte lineal por mes**



Fuente: elaboración propia.

2.5.8. Menú

A excepción del reporte, donde solo se puede regresar con el botón *back*, las demás pantallas son manejadas exclusivamente por el menú lateral, este está oculto todo el tiempo y se muestra con un gesto en la pantalla, deslizando el dedo desde la orilla izquierda hacia el centro, donde se encontrarán las distintas opciones para navegar.

Figura 12. **Menú**

Fuente: elaboración propia.

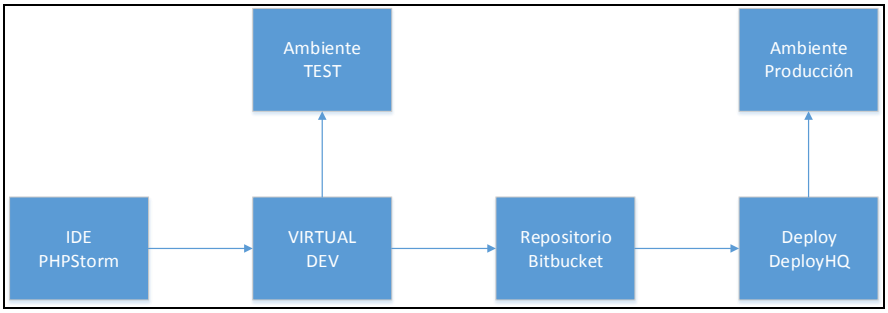
3. DESARROLLO

El desarrollo de la aplicación se llevó a cabo en dos ambientes con herramientas distintas, ya que ambos lenguajes tienen sus IDE y su metodología de trabajo, por lo que fue necesario utilizar dos ambientes de desarrollo.

3.1.1. Servidor

Para este se utilizó, como se describió anteriormente el lenguaje php, se utilizaron herramientas dedicadas al manejo del mismo, para facilitar tanto el desarrollo, las pruebas y la publicación.

Figura 13. **Ambiente de desarrollo del servidor**



Fuente: elaboración propia.

3.1.2. IDE

PHPStorm es un IDE especializado en el lenguaje php, sin embargo, tiene soporte integrado para más lenguajes para interactuar mejor, como html, xml,

javascript, entre otros. Provee una herramienta muy amplia que puede considerarse un entorno de desarrollo en sí, permitiendo trabajar el control de versiones, *deployment* automático, *autocomplete* de distintos lenguajes y una interfaz bastante sencilla de utilizar.

3.1.3. Virtual Dev

Para la realización de las pruebas, se creó un ambiente con el sistema operativo y las mismas especificaciones que el servidor de producción, en una máquina virtual montada en VMWare. En esta se instalaron las mismas herramientas que en el servidor de producción y se realizaron las pruebas iniciales durante la primera etapa de desarrollo del servidor, comprobando la funcionalidad a través de pruebas sin la integración de sistema móvil.

Utilizar un sistema como este permite realizar pruebas más versátiles al software, ya que se cuenta con un ambiente totalmente aislado y separado del de producción, por lo que no existe ningún riesgo de utilización de recursos o de alguna configuración que se deba modificar en el sistema, o algún nuevo componente que deba ser instalado, además de ser más accesible y tener el entorno de pruebas de forma local.

Como se mencionó en el punto anterior, el IDE cuenta con la opción de *deployment* automático, esto consiste en que, mientras se está trabajando en el sistema operativo *host*, al guardar el avance que se lleve, este se publicará automáticamente en la máquina virtual, ya que da la opción de mapear la dirección en donde se quiere que se publique el código. De esta forma se puede automatizar la forma de trabajo, ya que solo con guardar, se puede ir comprobando los avances que se realicen.

3.1.4. Repositorio

Para el control de versiones se utilizó GIT, que es una de las opciones más populares y potentes actualmente; como repositorio se utilizó BitBucket, que es una herramienta gratuita y una alternativa muy recomendada. La combinación de estas, integrándola al ambiente de PhpStorm, permite mantener el control de las versiones como un proceso bastante sencillo y muy fácil de manejar.

La utilización de un control de versiones, actualmente es una práctica que se vuelve necesaria, ya que mantiene seguro el código en un repositorio en internet, también permite tener un historial de lo que se ha realizado, dejando restaurar en cualquier punto donde se necesite. Además, el manejo de *branches* o ramas, permite modificar o agregar nuevas funcionalidades al sistema sin necesidad de alterar el código de producción. Para el desarrollo de esta aplicación se utilizaron dos ramas, una de desarrollo (dev) y una para producción (master), de esta forma se aseguró que mientras se realizan nuevas partes del desarrollo, los usuarios no se verán afectados en lo absoluto.

3.1.5. DeployHQ

Esta es una herramienta en la nube que permite automatizar el proceso de publicación del software a producción. Esta herramienta, a la cual se accede por medio de una página web, permite conectarse a los repositorios y servidores de producción. Mediante un poco de configuración, se pueden definir los distintos proyectos que se necesiten (1 si es la versión gratuita) y asociarlos con un repositorio. Cada vez que se desee publicar una nueva versión, mejora o la corrección de algún error, solo se debe ingresar a la página y esta realizará todo el manejo. Si en algún caso se diera algún error durante la publicación,

este sistema es capaz de revertir todo el proceso hasta el estado original, antes de iniciar la publicación.

Una de las herramientas más útiles de este sistema es que puede ser activado por *hooks*, además de emitir los propios, así se pueden programar acciones a realizar antes y después de una publicación, como la actualización de la base de datos con las modificaciones realizadas.

3.1.6. Ambiente de producción

Debe ser un servidor que cuente con acceso a internet, por lo cual, para este proyecto se eligió un Droplet de Digital Ocean, el cual provee una máquina virtual en la nube con una IP pública, con la que se puede tener acceso a internet y el control completo sobre lo que se instala y sobre los recursos que se quieren manejar, según el plan de pago que se elija.

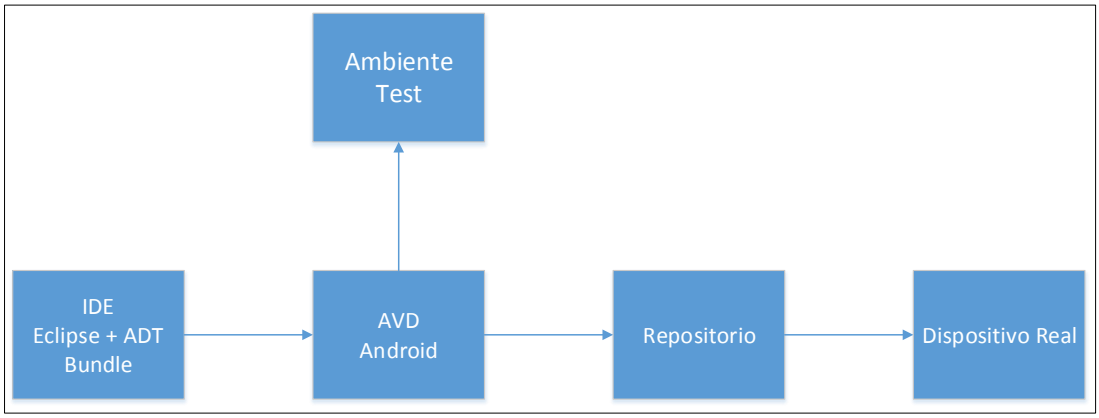
Este sistema, también conocido como PaaS (Plataform as a Service), provee una herramienta muy útil y fácil de usar, ya que se pueden elegir todas las herramientas que se desean usar, contando con una IP pública y acceso a internet las 24 horas. Tiene limitaciones en cuanto a recursos y ancho de banda de salida y entrada, pero esto puedo mejorarse según el plan de pago que se elija.

3.2. Android

Para la otra parte de la aplicación, que es la desarrollada en Android, se utilizó un ambiente de programación distinto al mencionado anteriormente,

siempre manteniendo las buenas prácticas de programación como el control de versiones y el manejo de los distintos ambientes para producción y desarrollo.

Figura 14. **Ambiente de desarrollo Android**



Fuente: elaboración propia.

3.2.1. Eclipse

Es uno de los IDE más utilizados para la programación en Android, puede descargarse en su versión individual e instalar manualmente las herramientas para desarrollo Android, o puede descargarse la versión existente que contiene todo preinstalado, solo es necesario descargarlo y se usa inmediatamente.

Este IDE viene equipado con todas las herramientas necesarias para desarrollar una aplicación en android, cuenta también con el emulador para tener un dispositivo virtual para las pruebas respectivas del sistema. Una de las herramientas más útiles es LogCat, una sencilla consola que permite ver el *log* de los eventos que suceden durante la ejecución del programa en el dispositivo, así se identifican cuáles fueron los errores.

3.2.2. AVD

El Android Virtual Device es el emulador en el cual se pueden probar las aplicaciones sin contar con un dispositivo Android real. Este permite ejecutar las aplicaciones como si se estuviera usando un dispositivo normal, al crearlo se puede elegir entre distintos tipos de dispositivos y distintas resoluciones de pantalla, para que se adapte de la mejor manera posible al requerimiento que se necesita desarrollar.

En este dispositivo virtual se establecieron las primeras pruebas del sistema, como los aspectos gráficos, transiciones entre pantallas y demás, dejando de lado los *web services*.

3.2.3. Repositorio

Para esta parte de la aplicación se utilizaron las mismas herramientas que en la anterior, pero en distinto repositorio. El código se versionó con GIT y se almacenó en Bitbucket, al igual que con el IDE anterior, Eclipse permite una sencilla integración con el manejo de versiones, manejo de *branches* y permite mantener las versiones locales y remotas.

3.2.4. Dispositivo móvil

Para las pruebas del sistema a nivel de producción, se utilizó un dispositivo real, ya que este debía conectarse a la aplicación almacenada en el servidor, Eclipse permite probar la aplicación en un dispositivo móvil, al igual que lo hace en el dispositivo virtual. Esto nos permite mantener los *logs* de errores, posibilidad de *debugs* y una mayor velocidad de despliegue de la

aplicación en comparación con el dispositivo virtual, lo cual ahorra mucho tiempo. Además, se tienen disponibles todas las herramientas del dispositivo real.

Las pruebas en el dispositivo incluyeron el consumo de los *web services*, la prueba del ambiente gráfico y la funcionalidad total del sistema, ya que en este punto se esperaba que todo funcionara normalmente, después de series de correcciones se pudo obtener un sistema estable.

4. RESULTADOS DE ACEPTACIÓN

Después de terminado el desarrollo de la aplicación es importante medir qué tan aceptado es el software, siguiendo la metodología TAM, la cual indica varios factores a tomar en cuenta:

- Actitud hacia el uso
- Facilidad de uso percibida
- Utilidad percibida

Tomando en cuenta estos factores, se diseñó una aplicación que pretende adaptarse a los mismos para tener un software que, además de funcional, sea aceptado por los usuarios y que se convierta en una de sus opciones para el control de sus finanzas personales.

Para medir la aceptación real que se podría tener, se realizó una encuesta aplicada a una muestra de la población, en la cual se intentan medir los factores anteriormente descritos, además de los datos demográficos de las personas, para identificar las poblaciones que podrían utilizarlo. Esta encuesta califica puntualmente la facilidad del uso, la utilidad y la disposición que tendrían los usuarios al utilizar un método como este, una aplicación móvil. Para mantener sus finanzas personales, en comparación con métodos tradicionales, o no llevar un registro de las finanzas como tal.

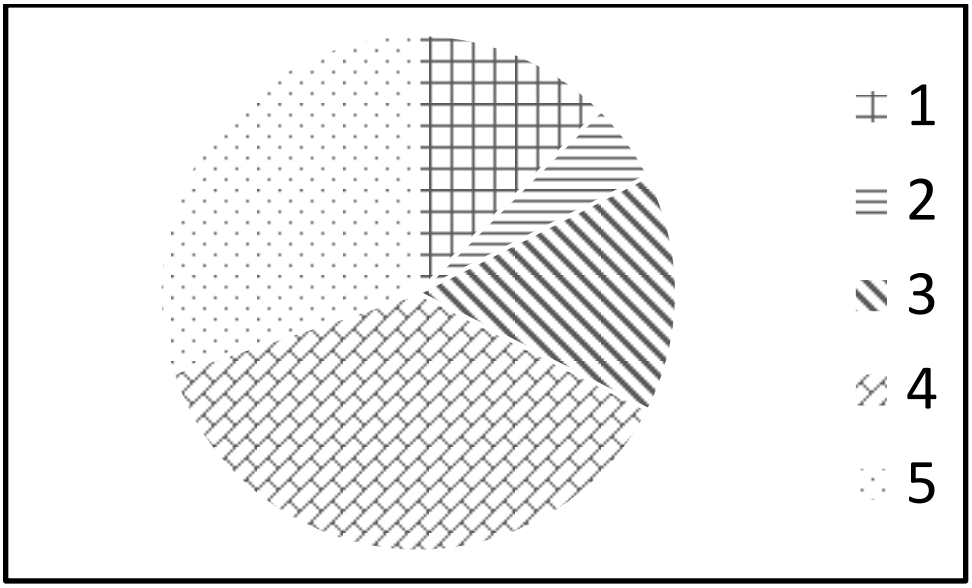
En total se entrevistó a 40 personas, las cuales hicieron uso de la aplicación y dieron su opinión acerca de la misma, el modelo de la encuesta puede verse en los apéndices de este trabajo de graduación. Se utilizó una

escala de 1-5, donde 1 representa la ponderación más baja y 5 la más alta, obteniendo así el grado de aceptación que tiene el software.

4.1. Facilidad de uso percibida

Se midió la facilidad de uso percibida, es decir, la forma en que los usuarios pueden desenvolverse en la aplicación. Para esto se pidió a los usuarios que usaran las distintas opciones que ofrece la aplicación y que dieran su opinión sobre qué tan sencillo fue el uso de la misma. Los resultados obtenidos se reflejan en la siguiente gráfica.

Figura 15. **Facilidad de uso percibida**



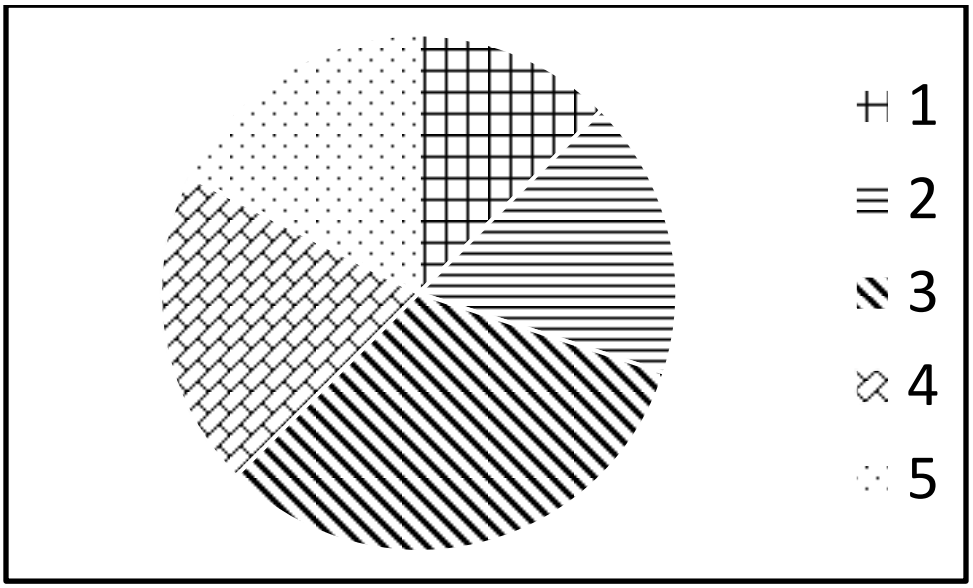
Fuente: elaboración propia.

Como se observa, el mayor porcentaje se encuentra en el número 4, por lo que se puede deducir que la aplicación fue percibida como bastante simple de usar, aunque se podría mejorar en varios aspectos para llegar al número 5.

4.2. Funcionalidad

Además de ser sencilla de usar, la aplicación tiene que ser útil al usuario, es decir, debe aportarle valor para que este considere usarla. De la misma forma, se consultó a los encuestados que tan útil es la aplicación para ellos o que tanto podría ayudarles en su vida diaria, los resultados obtenidos se encuentran en la siguiente imagen.

Figura 16. **Funcionalidad**



Fuente: elaboración propia.

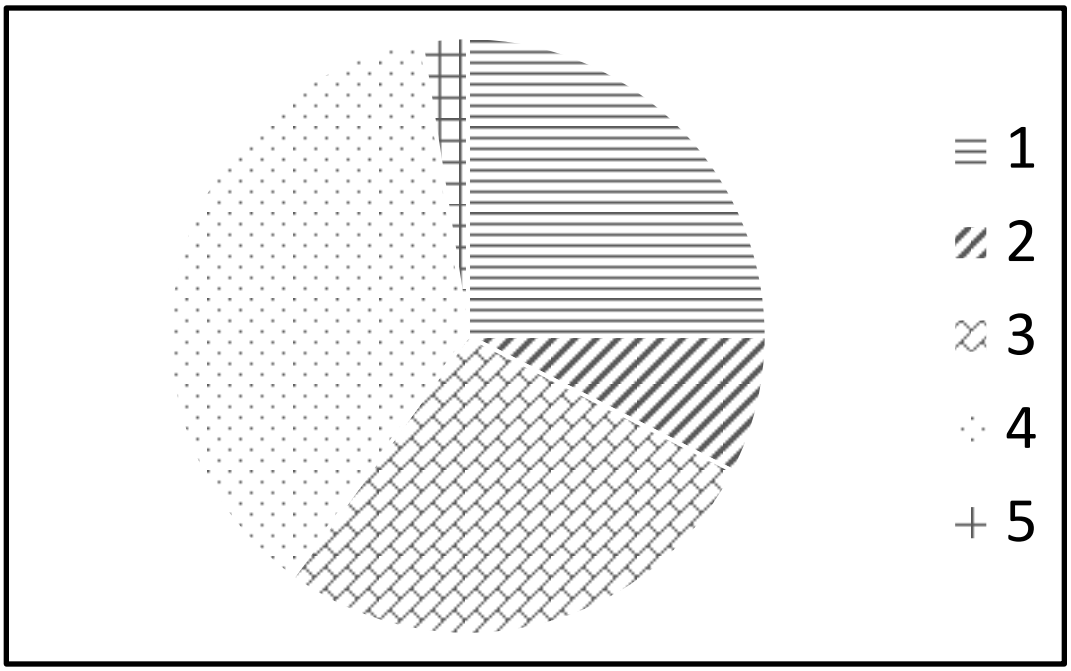
Como se observa, el número 3 es el de mayor incidencia, por lo que se deduce que a los usuarios la aplicación les pareció de utilidad, pero no tanto, ya

sea por las funciones con las que esta cuenta o porque simplemente no llevan el control de las finanzas.

4.3. Actitud hacia el uso

El último punto a tomar en cuenta para la metodología TAM, es la actitud hacia el uso, qué tanto estarían dispuestos los usuarios a utilizar un software de este tipo. Los resultados se reflejan en la siguiente gráfica.

Figura 17. **Actitud hacia el uso**



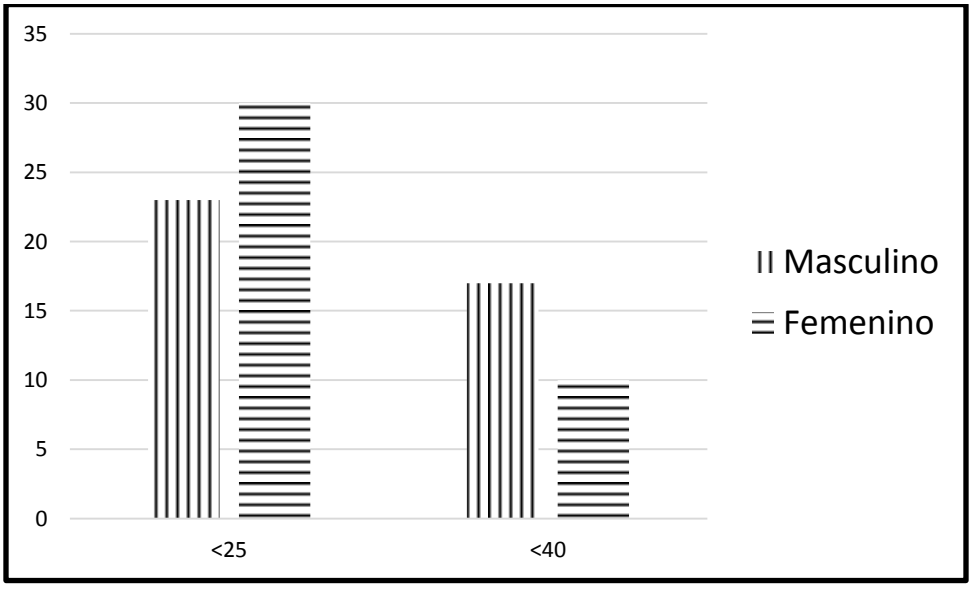
Fuente: elaboración propia.

Como se observa, una gran cantidad de personas indicó que no utilizaría un software de este tipo, ya que no lleva control de sus finanzas o simplemente no utilizaría un software para hacerlo, Los otros dos porcentajes mayoritarios se encuentran en los números 3 y 4, lo cual indica que los encuestados podrían considerar utilizar un software para las finanzas personales.

4.4. Datos demográficos

Se obtuvieron los datos demográficos de género y edad durante la encuesta, para conocer más sobre las poblaciones de usuarios, los datos recolectados son los siguientes.

Figura 18. **Datos demográficos**



Fuente: elaboración propia.

CONCLUSIONES

1. La sencillez en el diseño de la aplicación agrega un gran valor al usuario final.

2. Tener un planificador de presupuestos y ahorros mejora las buenas prácticas de la administración personal.

3. Una aplicación móvil que registre los gastos ayuda a aumentar la percepción de los gastos innecesarios.

4. Según los datos obtenidos en las encuestas, los usuarios perciben con buena aceptación el sistema, solo debe mejorarse en la parte de funcionalidades.

5. Llevar un control de ingresos como egresos, ayuda en la realización de un buen presupuesto, ya que con estos datos se tiene una estimación más real de las cantidades a tomar en cuenta para realizar un buen balance financiero.

RECOMENDACIONES

1. Basar el diseño de las aplicaciones en un flujo que permita una alta facilidad de uso para el usuario.

2. Planificar y controlar los gastos para tener una mejor administración de las finanzas personales.

3. Establecer una cultura de control de ingresos como de egresos, lo cual permitirá que se mantengan los planes o presupuestos que se establecieron.

BIBLIOGRAFÍA

1. GITMAN, Lawrence. *Principios de administración financiera*. 10a. ed. México: Pearson Education, 2003. 676 p. ISBN: 970-26-0428-1.

2. TYSON, Eric. *Finanzas personales para dummies*. 5a. ed. EUA: Wiley Publishing Inc, 2008. 510 p. ISBN: 978-0-470-22712-1.

APÉNDICES

1. Propuesta de encuesta para medición de la aceptación del software.

Encuesta de aceptación de software	
Facilidad de uso (1-5)	
Funcionalidad (1-5)	
Utilizaría este software o uno de este tipo (1-5)	
Edad	
Género	

Fuente: elaboración propia.