TECNOLÓGICO NACIONAL DE MÉXICO
INSTITUTO TECNOLÓGICO DE APIZACO

"ACTUALIZACIÓN TECNOLÓGICA Y REDISEÑO DE LA INTERFAZ DE USUARIO PARA WIPRECARGAS EN ANDROID"

TITULACIÓN INTEGRAL INFORME DE RESIDENCIA PROFESIONAL

PARA OBTENER EL TITULO PROFESIONAL DE:
INGENIERO EN TECNOLOGÍAS DE LA INFORMACIÓN Y COMUNICACIONES

PRESENTA:
KEVIN QUIROZ MORALES

NOMBRE DEL ASESOR:
ROBERTO ACOLTZI NAVA

Instituto Tecnológico de Apizaco

APIZACO, TLAX., JULIO 2024

Apizaco, Tlaxcala, 21/agosto /2024

OFICIO No: SC/OPV/0823/24

ASUNTO: Liberación de proyecto para titulación integral

CESAR REYNALDO RAMOS GÓMEZ
JEFE DEL DEPTO. DE DIVISIÓN DE ESTUDIOS PROFESIONALES
P R E S E N T E.

POR ESTE MEDIO LE INFORMO QUE HA SIDO LIBERADO EL SIGUIENTE PROYECTO PARA LA TITULACIÓN INTEGRAL.

NOMBRE DEL EGRESADO:	KEVIN QUIROZ MORALES
CARRERA:	INGENIERÍA EN TECNOLOGÍAS DE LA INFORMACIÓN Y COMUNICACIONES
NO. DE CONTROL:	19371057
NOMBRE DEL PROYECTO:	Actualización tecnológica y rediseño de la interfaz de usuario para Wiprecargas en Android
PRODUCTO U OPCIÓN:	TITULACIÓN INTEGRAL INFORME DE RESIDENCIA PROFESIONAL

AGRADEZCO DE ANTEMANO SU VALIOSO APOYO EN ESTA IMPORTANTE ACTIVIDAD PARA LA FORMACIÓN PROFESIONAL DE NUESTROS EGRESADOS.

A T E N T A M E N T E
Excelencia en Educación Tecnológica®
Pensar para Servir, Servir para Triunfar®

JOSÉ ANTONIO CUATEPOTZO VARELA
JEFE DEL DEPTO. DE SISTEMAS Y COMPUTACIÓN

SECRETARÍA DE EDUCACIÓN PÚBLICA
TECNOLÓGICO NACIONAL DE MÉXICO
INSTITUTO TECNOLÓGICO DE APIZACO
DEPARTAMENTO DE SISTEMAS Y COMPUTACIÓN

ROBERTO ACOLTZI NAVA
ASESOR INTERNO

c.c.p. División de Estudios Profesionales
c.c.p. Depto. de Servicios Escolares
c.c.p. Archivo
JAGV/mrol*

Apizaco, Tlaxcala, 23/septiembre/2024

D.E.P./458/2024
ASUNTO: AUTORIZACIÓN DE IMPRESIÓN

QUIROZ MORALES KEVIN
EGRESADO DE LA CARRERA DE INGENIERÍA
EN TECNOLOGÍAS DE LA INFORMACIÓN Y COMUNICACIÓNES
CON NÚMERO DE CONTROL 19371057
PRESENTE

Por este medio, me permito informar a usted que por aprobación de la comisión revisora asignada para valorar el trabajo que mediante la opción **TITULACIÓN INTEGRAL INFORME DE RESIDENCIAS PROFESIONALES** cuyo tema es: **ACTUALIZACIÓN TECNOLÓGICA Y REDISEÑO DE LA INTERFAZ DE USUARIO PARA WIPRECARGAS EN ANDROID** conforme a lo establecido en el procedimiento para la obtención del Título Profesional en el Instituto Tecnológico, la División de Estudios Profesionales a mi cargo le otorga:

AUTORIZACIÓN DE IMPRESIÓN

Debiendo entregar **2 USB** del mismo, apropiadamente encuadernados, a la brevedad, para presentar su acto de recepción profesional.

Sin otro particular por el momento me despido de Usted.

A T E N T A M E N T E
Excelencia en Educación Tecnológica®
Pensar para Servir, Servir para Triunfar®

CESAR REYNALDO RAMOS GÓMEZ
JEFE DE LA DIVISIÓN DE ESTUDIOS PROFESIONALES

SECRETARÍA DE EDUCACIÓN PÚBLICA
TECNOLÓGICO NACIONAL DE MÉXICO
INSTITUTO TECNOLÓGICO DE APIZACO
DIVISIÓN DE ESTUDIOS PROFESIONALES

Agradecimientos

En primer lugar, agradezco a Dios por guiarme y darme la fortaleza para superar los retos y alcanzar esta meta académica. También doy gracias a mis padres y familiares por su amor incondicional, apoyo emocional y sacrificios que hicieron posible mi educación. Su constante aliento fue mi motivación para superar los desafíos y alcanzar mis metas.

Agradezco profundamente a mis amigos, quienes estuvieron ahí en cada momento, brindándome su amistad, comprensión y alegría, haciendo más llevaderos los momentos difíciles y compartiendo los éxitos. Su presencia fue invaluable durante este trayecto.

Mi gratitud también se extiende a todos los profesores y mentores del Instituto Tecnológico de Apizaco, quienes no solo impartieron conocimientos, sino que también me guiaron, inspiraron y desafiaron intelectualmente. Sus enseñanzas han dejado una marca indeleble en mi formación profesional y personal.

Además, deseo agradecer a la empresa BAMGY por proporcionarme la oportunidad de realizar mi residencia profesional. Agradezco a todos los miembros del equipo por su colaboración, orientación y por compartir su experiencia y conocimientos conmigo. Esta experiencia práctica fue fundamental para mi desarrollo profesional y me permitió aplicar lo aprendido en un entorno real.

Por último, quiero reconocer al Instituto Tecnológico de Apizaco por su infraestructura, recursos académicos y por crear un entorno propicio para el aprendizaje y la investigación.

Resumen

Se renovo completamente la interfaz de usuario de la aplicación para mejorar su atractivo visual y modernizar su apariencia, en la cual se generarán cambios significativos en el diseño de la app de Wiprecargas se cambiaran aspectos como colores, tipografías, iconografías y elementos visuales mejorando así la apariencia de la app al igual se implementarán cambios disposición de los elementos de la interfaz de usuario implementando iconos e imágenes de alta calidad para ser más llamativos al usuario. Todos estos cambios van a proporcionar que la aplicación sea mucho más fácil de usar y más intuitiva al usuario ya que la app contara con una versión más reciente con las mejoras anteriormente mencionadas.

La creación de la aplicación de Wiprecargas ha sido un proceso cuidadosamente planificado y ejecutado utilizando dos herramientas fundamentales que han contribuido de manera significativa a su desarrollo son Android Studio y Git. Estas plataformas han desempeñado un papel esencial en la materialización de nuestra visión, proporcionando un entorno de desarrollo sólido y un sistema de control de versiones eficiente.

ÍNDICE GENERAL

ÍNDICE DE ILUSTRACIONES

1 INTRODUCCIÓN

En el mundo actual, la interfaz de usuario (UI) de una aplicación juega un papel crucial en la percepción y experiencia del usuario. La apariencia visual y la facilidad de uso pueden determinar el éxito o fracaso de una aplicación en el mercado. Dicho esto, el proyecto se enfoca en la renovación completa de la interfaz de usuario de Wiprecargas con el objetivo de mejorar su atractivo visual y modernizar su apariencia.

La app Wiprecargas ha sido una herramienta valiosa para sus usuarios, pero su diseño actual requiere una actualización significativa para mantenerse competitiva y alineada con las tendencias modernas de diseño. Este proyecto de renovación abordará varios aspectos clave del diseño de la aplicación, incluyendo la selección de paletas de colores más modernas y atractivas que no solo mejoren la estética de la aplicación, sino que también proporcionen una mejor experiencia visual para los usuarios. La actualización de las tipografías se enfocará en la legibilidad y el estilo, seleccionando fuentes que sean coherentes con la nueva identidad visual de la aplicación. Se implementarán iconos e imágenes de alta calidad, diseñados para ser más llamativos y fáciles de interpretar por los usuarios. Además, la reorganización de los elementos en la pantalla buscará una navegación más intuitiva y fluida, facilitando el acceso a las funciones principales de la aplicación.

Estas mejoras no solo harán que la aplicación sea más atractiva, sino que también aumentarán su usabilidad y accesibilidad, proporcionando una experiencia de usuario más intuitiva y agradable.

El desarrollo de la aplicación Wiprecargas ha sido un proceso cuidadosamente planificado y ejecutado, utilizando herramientas fundamentales como Android Studio y Git. Android Studio ha sido crucial para el desarrollo y la implementación de los cambios de UI, mientras que Git ha permitido un control de versiones eficiente y la colaboración en equipo.

2 Generalidades de la Empresa

- **Empresa:** Bamgy S. A. de C. V.
- **Dirección:** Avenida 5 de mayo #804, colonia centro, Apizaco Tlaxcala.

  C.P 90000
- **Teléfono:** 2411870014

Ubicación Geográfica de la Empresa:

Ilustración 1 Ubicación Geográfica de la empresa.

Logotipo de la empresa:

Ilustración 2 Logotipo de la empresa Bamgy S.A de C.V.

Misión: Somos una empresa facilitadora en pagos electrónicos, integra y transparente comprometida con cuidar los intereses de nuestros aliados.

Visión: Formar la mayor red de afiliados de tiendas de conveniencia para competir en conjunto con cualquier cadena comercial.

Valores:

- Colaboración.

- Confianza.
- Mejora continua.

Giro: Telecomunicaciones.

Descripción del área de realización:

Sistemas: En el área de Sistemas se enfoca en la creación y mejora de las interfaces visuales de las aplicaciones. El objetivo es proporcionar una experiencia de usuario óptima, alineada con las tendencias y estándares actuales para que el usuario tenga una mayor experiencia al utilizar nuestras plataformas.

3 Problemas a resolver

La interfaz de usuario de Wiprecargas enfrenta varios problemas significativos que afectan negativamente la experiencia del usuario y, en consecuencia, la eficiencia y popularidad de la aplicación. La apariencia visual de Wiprecargas ha permanecido sin cambios significativos desde su lanzamiento. Los colores, tipografías e iconografías actuales no reflejan las tendencias modernas de diseño, lo que hace que la aplicación se vea anticuada y poco atractiva para los usuarios contemporáneos. Esto puede resultar en una percepción negativa de la calidad y profesionalismo del servicio ofrecido.

La disposición actual de los elementos en la interfaz de usuario no es intuitiva ni eficiente, lo que dificulta la navegación y el uso de la aplicación. Los usuarios pueden encontrar complicado acceder a las funciones principales y completar tareas comunes, lo que lleva

a una frustración y posible abandono de la aplicación. La falta de usabilidad es uno de los problemas críticos. La aplicación no facilita una interacción sencilla y fluida, lo que impacta negativamente en la satisfacción del usuario. Esto incluye problemas con la legibilidad de textos debido a la elección inadecuada de tipografías y tamaños, así como la falta de iconos e imágenes de alta calidad que faciliten la comprensión rápida de las funciones.

4 Objetivos

4.1 Objetivo general.

Renovar completamente la interfaz de usuario de la aplicación Wiprecargas para mejorar su atractivo visual y modernizar su apariencia, haciendo que la aplicación sea más fácil de usar y más intuitiva para los usuarios.

4.2 Objetivos específicos.

- Actualizar la paleta de colores de la aplicación para que sea más atractiva y moderna.
- Elegir y aplicar nuevas tipografías que mejoren la legibilidad y el diseño general.
- Implementar nuevas iconografías y elementos visuales que sean más modernos y llamativos.
- Rediseñar la disposición de los elementos en la interfaz para mejorar la usabilidad y la navegación.

- Utilizar iconos e imágenes de alta calidad para aumentar el atractivo visual.
- Evaluar la satisfacción de los usuarios después de implementar los cambios.

5 Justificación

La modernización de la interfaz de usuario de Wiprecargas es crucial para asegurar que la aplicación siga siendo competitiva en el mercado. La actualización de la interfaz de usuario no es solo una cuestión estética, sino una necesidad estratégica que puede tener un impacto significativo en varios aspectos del negocio y la experiencia del usuario.

Primero, una interfaz moderna y atractiva mejora la percepción general de la aplicación. Los usuarios tienden a asociar un diseño visualmente atractivo con profesionalismo y calidad. Al actualizar la apariencia de Wiprecargas, se puede mejorar la percepción del servicio, haciendo que los usuarios actuales y potenciales vean la aplicación como una herramienta confiable y moderna.

En segundo lugar, la usabilidad es un componente esencial de la satisfacción del usuario. Una interfaz intuitiva y fácil de navegar puede hacer una gran diferencia en cómo los usuarios interactúan con la aplicación. Los cambios en la disposición de los elementos, la actualización de iconografías y la implementación de imágenes de alta calidad pueden reducir la curva de aprendizaje y hacer que el uso de la aplicación sea más eficiente. Esto es particularmente importante en una aplicación de venta de servicios, donde la rapidez y facilidad de acceso a las funciones principales pueden influir directamente en la satisfacción del usuario y en su disposición a utilizar la aplicación repetidamente.

Además, en el entorno altamente competitivo de las aplicaciones móviles, mantenerse al día con las tendencias de diseño es crucial para retener y atraer usuarios. Las aplicaciones con interfaces obsoletas no solo pierden atractivo visual, sino que también pueden parecer menos funcionales en comparación con las ofertas de la competencia. Al modernizar la interfaz de usuario de Wiprecargas, se puede asegurar que la aplicación se mantenga relevante y atractiva frente a las alternativas disponibles en el mercado.

Otro aspecto importante es la retención de usuarios. Una interfaz de usuario que no cumple con las expectativas modernas puede llevar a una disminución en la retención de usuarios. Los usuarios que se sienten frustrados con la navegación o encuentran la aplicación visualmente poco atractiva tienen más probabilidades de abandonar la plataforma. Mejorar la interfaz de usuario puede aumentar la retención de usuarios al proporcionar una experiencia más satisfactoria y reducir la fricción en la interacción diaria con la aplicación.

6 Marco teórico

6.1 Interfaces y Experiencia de Usuario (UX/UI)

La interfaz de usuario es uno de los aspectos más cruciales en el desarrollo de aplicaciones móviles. Su diseño y funcionalidad pueden influir significativamente en la experiencia del usuario y, en última instancia, en el éxito de la aplicación en el mercado.

6.1.1 Definición de UI (Interfaz de Usuario)

La Interfaz de Usuario (UI) se refiere a la parte de una aplicación o sistema con la que los usuarios interactúan directamente. Comprende todos los elementos visuales y táctiles que permiten la comunicación entre el usuario y la aplicación, tales como botones, menús, íconos, pantallas, y controles de entrada. La UI se centra en el diseño gráfico y en la disposición visual de estos elementos para garantizar que sean intuitivos y fáciles de usar. Una buena interfaz de usuario debe ser:

- Intuitiva: Los usuarios deben ser capaces de comprender cómo usar la aplicación sin necesidad de instrucciones detalladas.
- Consistente: El diseño y el comportamiento de los elementos deben ser coherentes en toda la aplicación para evitar confusión.
- Responsiva: Debe responder rápidamente a las acciones del usuario y proporcionar retroalimentación clara.

- Accesible: Debe ser usable por personas con diversas capacidades y limitaciones.

6.1.2 Definición de UX (Experiencia de Usuario)

La Experiencia de Usuario (UX) se refiere a cómo se siente una persona al interactuar con un sistema, producto o servicio. Incluye todos los aspectos de la interacción del usuario, desde la facilidad de uso hasta la eficiencia, la efectividad y la satisfacción general. La UX va más allá del diseño visual para abarcar la estructura, el flujo y la funcionalidad de la aplicación.

Los elementos clave de una buena UX son:

- Usabilidad: La aplicación debe ser fácil de aprender y usar.
- Accesibilidad: Todos los usuarios, incluidos aquellos con discapacidades, deben poder usar la aplicación.
- Desempeño: La aplicación debe ser rápida y confiable.
- Utilidad: La aplicación debe cumplir con las necesidades y expectativas de los usuarios.
- Satisfacción: La experiencia general debe ser agradable y gratificante para los usuarios.

Para mejorar la UX, los diseñadores deben realizar investigaciones con usuarios, pruebas de usabilidad y evaluar la retroalimentación para comprender las necesidades y

comportamientos de los usuarios. Herramientas como wireframes y prototipos ayudan a visualizar y refinar la experiencia del usuario antes de la implementación final.

6.1.3 Fundamentos Teóricos de Diseño de UI/UX

El diseño centrado en el usuario es un enfoque de diseño que pone a los usuarios en el centro del proceso de desarrollo, asegurando que sus necesidades, deseos y limitaciones sean considerados en cada etapa. Este enfoque busca crear interfaces que sean intuitivas y fáciles de usar, mejorando la satisfacción y eficiencia del usuario.

Los principios de usabilidad, tales como la facilidad de aprendizaje, eficiencia, memorabilidad, prevención de errores y satisfacción del usuario, son esenciales para una UI bien diseñada. La facilidad de aprendizaje permite que los usuarios nuevos puedan comenzar a utilizar la aplicación rápidamente sin necesidad de un entrenamiento extenso. La eficiencia se refiere a la rapidez con que los usuarios pueden realizar sus tareas después de haber aprendido a usar la aplicación. La memorabilidad implica que los usuarios pueden volver a utilizar la aplicación después de un periodo de no uso sin tener que reaprender todo de nuevo. La prevención de errores se centra en diseñar interfaces que minimicen las posibilidades de que los usuarios cometan errores, y la satisfacción del usuario se refiere a la experiencia global de uso.

La teoría del color es otro componente fundamental en el diseño de UI. Los colores pueden afectar significativamente la percepción y usabilidad de una aplicación. Una elección adecuada de colores no solo mejora la estética de la aplicación, sino que

también puede guiar a los usuarios a través de la interfaz de manera más eficiente, proporcionando contraste adecuado para facilitar la lectura y la navegación.

La tipografía es crucial para la legibilidad y la apariencia de una aplicación. La selección de tipografías legibles y atractivas es esencial para una buena UI. Las fuentes deben ser coherentes y adecuadas para diferentes tamaños de pantalla, proporcionando una experiencia de lectura cómoda y agradable.

La iconografía y los elementos visuales son componentes vitales para la comprensión y navegación dentro de una aplicación. Los iconos y elementos gráficos deben ser claros y representativos, facilitando la comprensión y navegación. Los elementos visuales de alta calidad no solo mejoran la estética de la aplicación, sino que también pueden proporcionar indicaciones visuales importantes para guiar a los usuarios.

6.2 Android Studio.

6.2.1 ¿Qué es Android Studio?

Android Studio es el entorno de desarrollo integrado (IDE) oficial para la creación de aplicaciones Android. Desarrollado por Google, Android Studio proporciona un conjunto completo de herramientas para el desarrollo, depuración y despliegue de aplicaciones en el sistema operativo Android.

Android Studio se basa en el software IntelliJ IDEA de JetBrains y está diseñado específicamente para la programación en Android, lo que lo convierte en una plataforma robusta y eficiente para desarrolladores de todos los niveles.

6.2.2 Características Principales de Android Studio

Editor de Código Inteligente: Android Studio ofrece un editor de código avanzado con soporte para la finalización de código, análisis de código estático y navegación avanzada. Proporciona sugerencias de código y resaltado de sintaxis para mejorar la productividad y minimizar errores.

Diseñador de Interfaz de Usuario (UI): Incluye un diseñador visual que permite arrastrar y soltar componentes de la interfaz de usuario para construir layouts. El diseñador visual facilita la creación de interfaces de usuario complejas sin necesidad de escribir todo el código XML manualmente.

Sistema de Compilación Basado en Gradle: Android Studio utiliza Gradle como sistema de construcción, lo que permite una configuración flexible y escalable. Gradle facilita la gestión de dependencias y la construcción de aplicaciones con diferentes variantes y configuraciones.

6.3 Lenguajes de programación

6.3.1 ¿Qué es un Lenguaje de programación?

Un lenguaje de programación es un conjunto de instrucciones y reglas que permiten a los desarrolladores comunicar y crear programas de software. Estos lenguajes proporcionan

la sintaxis y la estructura necesarias para escribir algoritmos y manipular datos, permitiendo a las computadoras ejecutar tareas específicas. Los lenguajes de programación pueden variar en su nivel de abstracción, desde lenguajes de bajo nivel que están más cerca del código máquina, hasta lenguajes de alto nivel que son más fáciles de leer y escribir para los humanos.

Los lenguajes de programación se dividen en varias categorías según su propósito y nivel de abstracción:

- *Lenguajes de Bajo Nivel:* Estos lenguajes, como el lenguaje ensamblador, están muy cerca del hardware y permiten un control detallado de los recursos del sistema. Son eficientes en términos de rendimiento, pero difíciles de aprender y usar debido a su complejidad.
- *Lenguajes de Alto Nivel:* Estos lenguajes, como Python, Java y C++, abstraen muchos detalles del hardware, lo que facilita su uso y comprensión. Son más adecuados para el desarrollo de aplicaciones complejas y permiten a los desarrolladores centrarse más en la lógica del programa que en los detalles del hardware.
- *Lenguajes de Programación Orientados a Objetos (OOP):* Estos lenguajes, como Java y C#, están diseñados para facilitar el desarrollo de programas modulares y reutilizables mediante el uso de objetos y clases.

- *Lenguajes Funcionales:* Lenguajes como Haskell y Lisp se centran en el uso de funciones matemáticas y evitan el estado y las variables mutables, lo que puede conducir a un código más predecible y fácil de depurar.

6.3.2 Lenguajes de programación para Android

El desarrollo de aplicaciones Android se puede realizar utilizando varios lenguajes de programación, cada uno con sus propias ventajas y características. A continuación, se presentan algunos de los lenguajes más populares utilizados para desarrollar aplicaciones Android:

1. **Kotlin:** Kotlin es un lenguaje de programación moderno que se ha convertido en el lenguaje preferido para el desarrollo de aplicaciones Android desde que Google lo adoptó oficialmente en 2017. Kotlin es totalmente interoperable con Java, lo que permite a los desarrolladores utilizar bibliotecas y frameworks existentes de Java mientras aprovechan las características avanzadas de Kotlin. Kotlin ofrece una sintaxis más concisa y expresiva que Java, reduciendo el código boilerplate y mejorando la legibilidad del código. Además, incluye características modernas como la nulabilidad controlada por el compilador, funciones de extensión y soporte para la programación funcional.
2. **C++:** C++ se utiliza principalmente en el desarrollo de componentes de rendimiento crítico para aplicaciones Android, como motores de juegos y bibliotecas de procesamiento multimedia. Las aplicaciones desarrolladas en C++

pueden aprovechar el Android Native Development Kit (NDK) para ejecutar código nativo en la plataforma Android.

Aunque C++ ofrece un control detallado sobre los recursos del sistema y puede mejorar significativamente el rendimiento de las aplicaciones, su complejidad y dificultad de uso pueden hacer que sea menos accesible para los desarrolladores principiantes.

3. **Dart:** Dart, en combinación con el framework Flutter, ha ganado popularidad como una opción para el desarrollo de aplicaciones multiplataforma, incluyendo Android. Dart es un lenguaje de programación desarrollado por Google que es fácil de aprender y usar. Flutter permite a los desarrolladores crear aplicaciones nativas de alto rendimiento con una sola base de código para múltiples plataformas.
4. **Python:** Aunque Python no es un lenguaje de programación nativo para Android, se puede utilizar para desarrollar aplicaciones Android a través de frameworks como Kivy y BeeWare. Estos frameworks permiten a los desarrolladores escribir aplicaciones en Python que se pueden ejecutar en dispositivos Android.
5. **Java:** Java ha sido el lenguaje principal para el desarrollo de aplicaciones Android desde el lanzamiento de la plataforma. Es un lenguaje de programación orientado a objetos que es ampliamente utilizado y soportado por una gran comunidad de desarrolladores. Las aplicaciones escritas en Java para Android se ejecutan en la máquina virtual de Android (Dalvik/ART), lo que permite que el código sea portátil

y eficiente. Java proporciona una gran cantidad de bibliotecas y frameworks que facilitan el desarrollo de aplicaciones complejas. Además, su sintaxis clara y estructura modular hacen que el código sea más fácil de mantener y escalar.

6.4 Java

6.4.1 ¿Qué es Java?

Java es un lenguaje de programación de propósito general que se destaca por su orientación a objetos y su alto nivel de abstracción. Fue desarrollado por Sun Microsystems, una empresa que posteriormente fue adquirida por Oracle Corporation, y su primera versión fue lanzada al público en 1995. Java se ha ganado una reputación por su capacidad para ser portado a diferentes plataformas, lo que significa que el mismo código puede ejecutarse en diferentes dispositivos sin necesidad de modificaciones significativas. Además, es reconocido por su robustez, lo que se traduce en una mayor estabilidad y menor probabilidad de fallos. La seguridad también es uno de los pilares fundamentales de Java, lo que lo convierte en una opción preferida para aplicaciones que requieren un alto grado de protección. Gracias a estas características, Java ha sido ampliamente adoptado en una variedad de áreas, incluyendo sistemas empresariales, aplicaciones móviles y desarrollo web.

6.4.2 Características de Java

Java posee varias características que lo han hecho popular y confiable en el desarrollo de software:

- *Orientación a Objetos*: Java es un lenguaje de programación orientado a objetos, lo que significa que organiza el software en objetos y clases, promoviendo la reutilización del código y facilitando el mantenimiento y la expansión de los sistemas.
- *Portabilidad*: El lema "write once, run anywhere" (escribe una vez, ejecuta en cualquier lugar) destaca la capacidad de Java para ejecutarse en cualquier dispositivo con una Máquina Virtual de Java (JVM). Esto se debe a que el código Java se compila en bytecode, que puede ser interpretado por cualquier JVM.
- *Seguridad*: Java proporciona características de seguridad robustas mediante su modelo de caja de arena (sandbox), que restringe el acceso de las aplicaciones a los recursos del sistema.
- *Gestión Automática de Memoria*: Java maneja automáticamente la memoria a través del recolector de basura (garbage collector), que se encarga de liberar memoria ocupada por objetos que ya no son referenciados.
- *Multihilo (Multithreading)*: Java soporta la programación concurrente a través de su modelo de multihilo, permitiendo la ejecución simultánea de múltiples partes de un programa.

- *Bibliotecas Ricas*: Java cuenta con una amplia biblioteca estándar (Java Standard Edition) que proporciona muchas funciones predefinidas para tareas comunes, lo que acelera el desarrollo.

6.4.3 Java en el Desarrollo de Aplicaciones Android

Java ha sido el lenguaje de programación principal para el desarrollo de aplicaciones Android desde el inicio de la plataforma. Aunque recientemente Kotlin ha ganado popularidad, Java sigue siendo ampliamente utilizado debido a su estabilidad y la vasta cantidad de recursos y bibliotecas disponibles.

- *Compatibilidad con Android*: La mayoría de las APIs de Android están diseñadas con Java en mente, lo que facilita el desarrollo de aplicaciones Android utilizando este lenguaje.
- *Extensa Documentación y Comunidad*: Java tiene una amplia documentación oficial y una gran comunidad de desarrolladores, lo que facilita encontrar soluciones a problemas y obtener apoyo.
- *Integración con Android Studio*: Android Studio proporciona un soporte robusto para el desarrollo en Java, incluyendo herramientas de depuración y pruebas específicas para aplicaciones Android.
- *Bibliotecas y Frameworks*: Existen muchas bibliotecas y frameworks desarrollados en Java que pueden ser utilizados en proyectos Android para añadir funcionalidades, mejorar el rendimiento y simplificar el desarrollo.

6.5 Modelo Vista Controlador (MVC)

6.5.1 Definición

El Modelo Vista Controlador (MVC) es un patrón de diseño arquitectónico que se utiliza para separar la lógica de negocio, la interfaz de usuario y el control de entrada en aplicaciones de software. Este patrón promueve la separación de preocupaciones, lo que facilita la gestión, el mantenimiento y la escalabilidad de las aplicaciones. MVC divide una aplicación en tres componentes principales: Modelo, Vista y Controlador. Cada uno de estos componentes tiene responsabilidades específicas y se comunica con los otros componentes de manera definida.

6.5.2 Componentes

6.5.2.1 Modelo

El Modelo es responsable de la lógica de negocio y la gestión de datos de la aplicación. Representa los datos que se utilizan en la aplicación y las reglas de negocio que gobiernan el acceso y la manipulación de estos datos. El Modelo se encarga de:

- Consultar y actualizar datos de la base de datos o de otras fuentes de datos.
- Aplicar las reglas de negocio y las validaciones.
- Notificar a la Vista sobre los cambios en los datos.

  En el contexto de una aplicación Android, el Modelo puede incluir clases de datos, servicios, y la lógica de acceso a la base de datos, utilizando herramientas como SQLite o Room.

6.5.2.2 Vista

La Vista es responsable de la representación visual de los datos. Muestra la información al usuario y gestiona la interacción del usuario con la interfaz. La Vista se encarga de:

- Renderizar los datos proporcionados por el Modelo.
- Capturar y manejar la entrada del usuario.
- Actualizar la interfaz de usuario en respuesta a los cambios en el Modelo.

En aplicaciones Android, las Vistas se implementan utilizando elementos de la interfaz de usuario como actividades, fragmentos y layouts XML.

6.5.2.3 Controlador

El Controlador actúa como un intermediario entre el Modelo y la Vista. Procesa la entrada del usuario, llama a los métodos del Modelo para actualizar los datos y selecciona la Vista adecuada para mostrar la salida. El Controlador se encarga de:

- Recibir la entrada del usuario desde la Vista.
- Interpretar la entrada y realizar acciones correspondientes en el Modelo.
- Actualizar la Vista en respuesta a los cambios en el Modelo.

En Android, los Controladores pueden ser implementados como clases separadas o integrarse dentro de actividades y fragmentos que coordinan las interacciones entre el Modelo y la Vista.

6.5.3 Implementación de MVC en Aplicaciones Android

La implementación de MVC en aplicaciones Android puede variar según las necesidades específicas del proyecto. Sin embargo, el patrón general implica definir claramente las responsabilidades de cada componente y establecer la comunicación entre ellos. En una aplicación Android típica, el Modelo incluye clases de datos y servicios que manejan la lógica de negocio y el acceso a la base de datos. Las Vistas son representadas por actividades, fragmentos y layouts que muestran la información al usuario. Los Controladores, que pueden ser actividades o fragmentos, gestionan la entrada del usuario y coordinan las interacciones entre el Modelo y la Vista.

6.5.4 Ventajas y Desventajas del Patrón MVC

Ventajas:

- **Separación de Preocupaciones:** MVC permite separar la lógica de negocio de la interfaz de usuario, lo que facilita la gestión y el mantenimiento del código. Cada componente tiene responsabilidades claras y definidas.
- **Facilidad de Pruebas:** La separación de componentes facilita la realización de pruebas unitarias y de integración, ya que cada componente puede ser probado de manera independiente.
- **Reutilización de Código:** El Modelo y la Vista pueden ser reutilizados en diferentes contextos, lo que aumenta la eficiencia del desarrollo.

- **Colaboración Mejorada:** La separación de responsabilidades permite que diferentes equipos trabajen en paralelo en la lógica de negocio, la interfaz de usuario y el control de flujo.

6.6 Control de Versiones

6.6.1 ¿Qué es control de versiones?

El control de versiones es un sistema que permite a los desarrolladores gestionar los cambios en el código fuente a lo largo del tiempo. Este sistema es fundamental en el desarrollo de software, ya que permite rastrear y revertir cambios, colaborar con otros desarrolladores y mantener un historial completo de modificaciones. El control de versiones no solo se aplica al código fuente, sino también a documentos, configuraciones y cualquier tipo de información que pueda ser modificada y necesite ser rastreada.

6.7 Control de versiones con Git

6.7.1 ¿Qué es Git?

Git es un sistema de control de versiones distribuido diseñado para manejar todo tipo de proyectos con velocidad y eficiencia. Fue creado por Linus Torvalds en 2005 para el desarrollo del núcleo de Linux, pero desde entonces se ha convertido en una de las herramientas más populares para la gestión de código fuente en la industria del software.

6.7.2 Uso de Git en Proyectos Colaborativos

Git es especialmente potente en entornos colaborativos, donde varios desarrolladores trabajan juntos en el mismo proyecto. Facilita la coordinación y la integración de cambios, lo que es crucial en proyectos de gran escala.

6.7.3 Flujo de Trabajo con Git

El flujo de trabajo con Git puede variar dependiendo del proyecto y el equipo, pero generalmente sigue un conjunto de pasos bien definidos para asegurar una gestión eficiente del código.

6.8 Emulador Físico o Virtual

6.8.1 ¿Qué son los Emuladores Android?

Los emuladores Android son programas de software que replican el hardware y el sistema operativo de un dispositivo Android en una computadora. Estos emuladores permiten a los desarrolladores probar y depurar aplicaciones en un entorno virtual que simula el comportamiento de dispositivos reales. Los emuladores son una parte esencial del desarrollo de aplicaciones móviles, ya que proporcionan una plataforma flexible y accesible para el desarrollo y la prueba de aplicaciones sin necesidad de utilizar dispositivos físicos.

6.8.2 Ventajas y Desventajas de Usar Emuladores.

Ventajas:

- **Accesibilidad y Conveniencia:** Los emuladores permiten a los desarrolladores probar aplicaciones en múltiples configuraciones de dispositivos sin necesidad de adquirir dispositivos físicos, lo que ahorra costos y simplifica el proceso de prueba.
- **Pruebas en Diferentes Configuraciones:** Los emuladores pueden simular diferentes versiones de Android y configuraciones de hardware, facilitando la prueba de compatibilidad en un amplio rango de dispositivos.
- **Depuración y Monitoreo:** Los emuladores se integran con herramientas de desarrollo como Android Studio, ofreciendo potentes capacidades de depuración y monitoreo de rendimiento que ayudan a identificar y resolver problemas en el código.
- **Automatización de Pruebas:** Los emuladores se pueden utilizar en procesos de integración continua (CI) para automatizar pruebas de aplicaciones, mejorando la eficiencia del desarrollo y asegurando la calidad del software.

6.9 Emulador MEmu

6.9.1 ¿Qué es?

MEmu es un emulador de Android que permite ejecutar aplicaciones y juegos de Android en una computadora con sistema operativo Windows. A diferencia de otros emuladores, MEmu está optimizado específicamente para ofrecer una experiencia de juego fluida y

de alto rendimiento en PC. Soporta una amplia gama de configuraciones de hardware y versiones de Android, lo que lo hace versátil para diferentes necesidades de desarrollo y pruebas.

Características Principales de MEmu:

- **Compatibilidad Amplia:** MEmu puede emular varias versiones de Android, desde KitKat (4.4) hasta Pie (9.0), permitiendo a los desarrolladores probar sus aplicaciones en múltiples versiones del sistema operativo.
- **Optimización para Juegos:** MEmu está diseñado con un enfoque en el rendimiento de los juegos, ofreciendo una experiencia de usuario superior con gráficos de alta calidad y tiempos de respuesta rápidos.
- **Soporte para Periféricos:** Permite el uso de teclados, ratones y controladores de juegos, proporcionando una experiencia de usuario más rica y permitiendo a los desarrolladores probar interacciones más complejas.
- **Multitarea y Múltiples Instancias:** Los usuarios pueden ejecutar múltiples instancias de MEmu simultáneamente, lo que es útil para pruebas de aplicaciones que requieren interacción entre varias cuentas o dispositivos.
- **Acceso a Google Play Store:** MEmu incluye soporte para Google Play Store, facilitando la instalación y prueba de aplicaciones desde la tienda oficial de Android.

6.9.2 Ventajas

Rendimiento:

- Alto Rendimiento: MEmu está optimizado para utilizar eficientemente los recursos del sistema, lo que resulta en una experiencia de emulación fluida y de alta velocidad. Esto es particularmente importante para juegos y aplicaciones que requieren gráficos intensivos y procesamiento rápido.
- Gráficos Avanzados: Soporta gráficos OpenGL y DirectX, permitiendo una representación visual de alta calidad y mejorando la experiencia de usuario en aplicaciones gráficamente exigentes.

Facilidad de Uso:

- Interfaz Intuitiva: MEmu ofrece una interfaz de usuario amigable y fácil de navegar, lo que facilita su configuración y uso incluso para aquellos que no son expertos en tecnología.
- Instalación Sencilla: La instalación de MEmu es directa y no requiere configuraciones complejas, lo que permite a los desarrolladores y usuarios ponerse en marcha rápidamente.

Flexibilidad:

- Configuraciones Personalizables: Los usuarios pueden ajustar las configuraciones de CPU, memoria y resoluciones de pantalla para simular diferentes dispositivos y condiciones de uso, lo que es útil para pruebas detalladas y específicas.

- Soporte para Aplicaciones de 32 y 64 bits: MEmu puede ejecutar aplicaciones tanto de 32 como de 64 bits, lo que amplía su compatibilidad con una mayor variedad de aplicaciones y juegos.

Multitarea:

- Múltiples Instancias: Permite la ejecución de múltiples instancias de Android en una sola máquina, lo que es beneficioso para pruebas de aplicaciones que requieren interacción entre varias cuentas o usuarios.
- Sincronización de Instancias: Los usuarios pueden sincronizar operaciones en varias instancias, facilitando pruebas repetitivas y mejorando la eficiencia.

Integración y Compatibilidad:

- Soporte para Google Play Store: Incluye acceso a Google Play Store, lo que facilita la instalación y prueba de aplicaciones desde la tienda oficial de Android.
- Compatibilidad con Juegos: Está especialmente optimizado para juegos de Android, ofreciendo soporte para controladores de juegos, mapeo de teclado y ratón, y otras características que mejoran la experiencia de juego.

6.10 Git.

El uso de Git como sistema de control de versiones es fundamental para gestionar el código fuente de manera eficiente. Git permite a los desarrolladores colaborar en proyectos de software, realizando un seguimiento detallado de los cambios realizados en el código fuente a lo largo del tiempo. Con Git, es posible crear ramas (branches) para

desarrollar nuevas funcionalidades o corregir errores, y luego fusionarlas (merge) en la rama principal (main) una vez que se han probado y validado. Esta capacidad de gestionar versiones y ramas facilita la colaboración entre equipos de desarrollo y asegura que el código sea coherente y estable.

6.11 Figma.

Figma es una herramienta de diseño de interfaces que permite la creación de prototipos y la colaboración en tiempo real. Es fundamental para la planificación visual y el diseño de la nueva UI. Figma permite a los diseñadores crear prototipos interactivos que simulan la experiencia de usuario final, facilitando la evaluación y ajuste del diseño antes de su implementación. Además, Figma ofrece una plataforma colaborativa en la que múltiples diseñadores y desarrolladores pueden trabajar simultáneamente, compartiendo ideas y feedback en tiempo real. Esto no solo acelera el proceso de diseño, sino que también asegura que todos los miembros del equipo estén alineados con la visión del proyecto.

6.12 MySQL.

MySQL es un sistema de gestión de bases de datos relacional de código abierto que se utiliza para almacenar y gestionar datos de aplicaciones móviles.

6.13 Requisitos Funcionales

- **Registro de Usuarios:** La aplicación debe permitir a los usuarios registrarse creando cuentas personales con información básica, como nombre, número de teléfono y dirección de correo electrónico.
- **Recargas Móviles:** La capacidad de recargar saldo de manera rápida y segura para teléfonos móviles, incluyendo diferentes operadores y aviones de prepago, y pago de servicios y pines.
- **Historial de Transacciones:** Un registro detallado de todas las transacciones realizadas, incluyendo la fecha, la cantidad recargada.
- **Notificaciones en Tiempo Real:** La capacidad de enviar notificaciones a los usuarios para confirmar el éxito de la recarga y proporcionar detalles sobre la transacción y mensajes de suma importancia.
- **Métodos de Pago Seguros:** Integración con métodos de pago seguros, como tarjetas de crédito, débito o servicios de pago en línea.
- **Promociones y Descuentos:** La posibilidad de aplicar promociones, descuentos o programas de lealtad para incentivar el uso continuo de la aplicación.
- **Interfaz Intuitiva y Amigable:** Una interfaz de usuario intuitiva y fácil de usar para facilitar la navegación y la realización de recargas sin complicaciones.

6.14 Requisitos no funcionales.

- **Seguridad de Datos:** Garantizar la seguridad de los datos del usuario y las transacciones mediante la implementación de medidas de cifrado y protección contra amenazas cibernéticas.
- **Tiempo de Respuesta Rápido:** Mantener un tiempo de respuesta rápido para las transacciones y la navegación dentro de la aplicación, brindando una experiencia fluida al usuario.
- **Disponibilidad Continua:** Garantizar la disponibilidad continua de la aplicación para que los usuarios puedan realizar recargas en cualquier momento sin interrupciones.
- **Escalabilidad:** Diseñar la aplicación para ser escalable y manejar un aumento en la cantidad de usuarios y transacciones a medida que la base de usuarios crece.
- **Respaldo de Datos:** Implementar un sistema robusto de respaldo de datos para evitar la pérdida de información crítica en caso de fallas técnicas.
- **Actualizaciones y Mantenimiento:** Garantizar la facilidad de realizar actualizaciones y mantenimiento de la aplicación para agregar nuevas funciones y corregir posibles problemas.
- **Eficiencia en el Consumo de Recursos:** Optimizar el consumo de recursos, como la batería del dispositivo y el uso de datos, para garantizar una eficiencia energética y una experiencia de usuario económica.

- **Usabilidad en Diversos Contextos:** Asegurar que la aplicación sea utilizable en diferentes contextos, como entornos con poca luz, y que sea accesible para usuarios con diversas capacidades.

7 Desarrollo.

7.1 Estudio de situacion actual.

El objetivo principal del proyecto es renovar completamente la interfaz de usuario de la aplicación Wiprecargas para mejorar su atractivo visual y modernizar su apariencia. La actualización se centrará en varios aspectos clave del diseño y la funcionalidad de la aplicación, con el fin de proporcionar una experiencia de usuario más intuitiva y agradable.

7.1.1 Colores y Estética.

El diseño actual de la aplicación Wiprecargas utiliza colores que no son tan llamativos, lo cual afecta negativamente la percepción visual de la aplicación. Los usuarios pueden encontrar la interfaz monótona y poco atractiva, lo que puede disminuir el interés en su uso continuo.

En esta imagen muestra el como es la primera vista al entrar a la app en el cual el usuario tendra que pponer su usuario y contraseña, pero sus colores de botones, imágenes y tipo de letra se veian mal esteticamente.

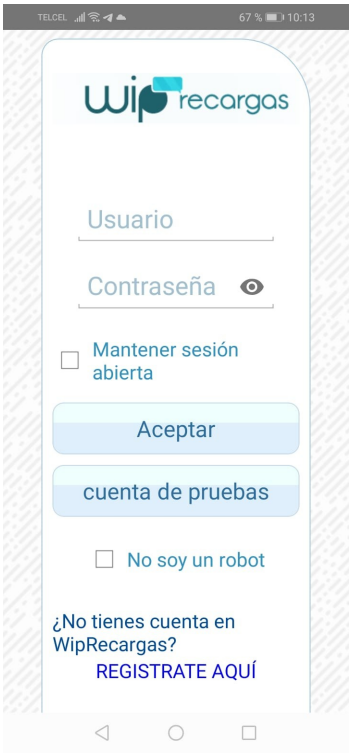

Ilustración 3 Página de Inicio de sesión

A continuación se muestra la pantalla para registrarse por primera vez, en este campo nos pide que el usuario se registre con su nombre, email, lada, teléfono y extención.

Ilustración 4 Pantalla de registro para nuevos usuarios.

La siguiente pantalla nos muestra los servicios que podemos hacer en la pantalla como lo es el poner recargas móviles en el cual se pone el número y se confirma.

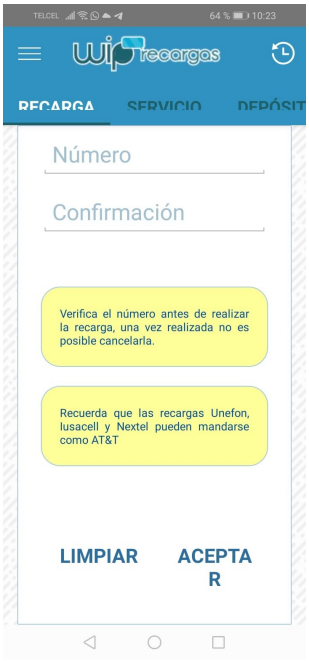

Ilustración 5 Pantalla para poner recargas móviles.

En esta imagen muestra que se puede pagar por deposotos bancarios de esta manera el cliente podra contar con los beneficios de la app como recargargas y compra de pines.

Ilustración 6 Pantalla para depósitos bancarios.

Al final de la pantalla muestra los terminos y condiciones.

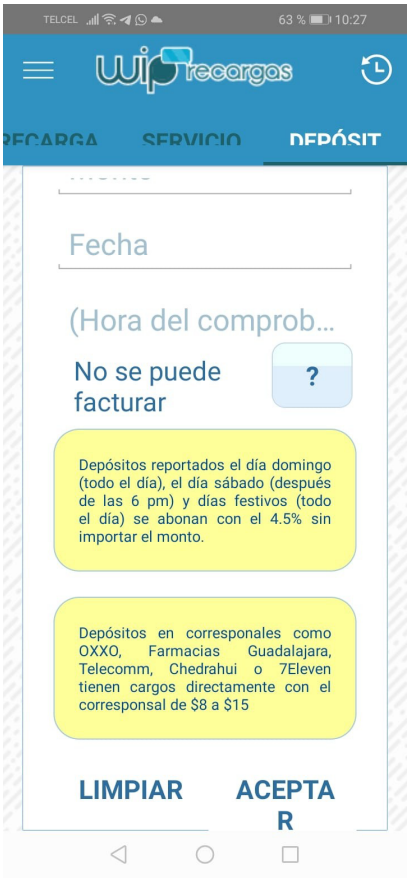

Ilustración 7Pantalla de términos y condiciones.

7.1.2 Interacción con el cliente.

La interacción con el cliente en la versión actual de la aplicación es un poco difícil. Los usuarios pueden encontrar complicaciones al navegar por la aplicación o al realizar ciertas acciones, lo que afecta la usabilidad y la satisfacción del usuario.

7.2 Prototipo.

7.2.1 Descripción del prototipo.

En Figma, se ha desarrollado un prototipo detallado que muestra los cambios planificados para la aplicación Wiprecargas. Este prototipo incluye varias mejoras en el diseño y la funcionalidad. Se han seleccionado nuevas imágenes y una paleta de colores más vibrante y atractiva. Estos cambios están diseñados para hacer que la aplicación sea visualmente más atractiva y moderna. Se ha elegido una nueva tipografía que mejora la legibilidad y aporta un aspecto más contemporáneo a la aplicación. La tipografía juega un papel crucial en la estética y la usabilidad de la interfaz de usuario. Se han diseñado nuevas pantallas para incorporar funcionalidades adicionales y mejorar la estructura de la aplicación. Esto incluye la adición de pantallas específicas para la recuperación de contraseñas, la gestión de cuentas y otras funcionalidades clave.

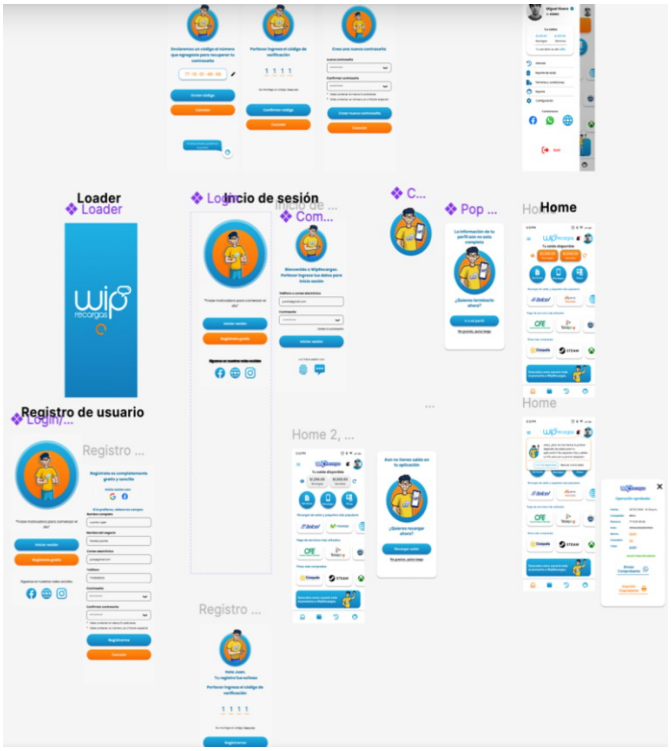

Ilustración 8 Esta pantalla muestra desde el inicio de sesión.

Se muestra todos los cambios y creación de nuevas pantallas que se deben realizar dentro de la app para tener una mejor interfaz más llamativa.

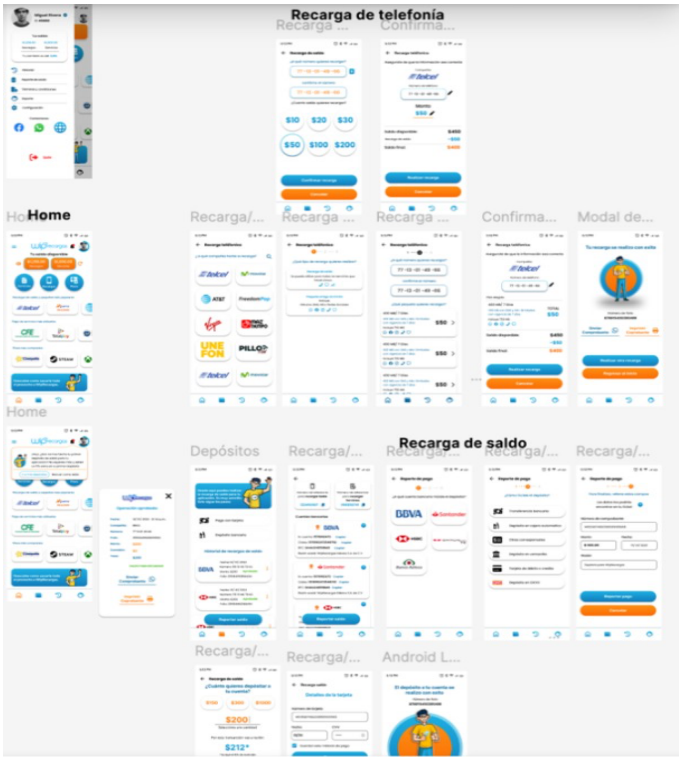

Ilustración 9 Pantalla que muestra el prototipo de las pantallas de recargas móviles.

Todo estos cambios incluyen colores, imágenes, tipo de letra, creación de nuevas pantallas.

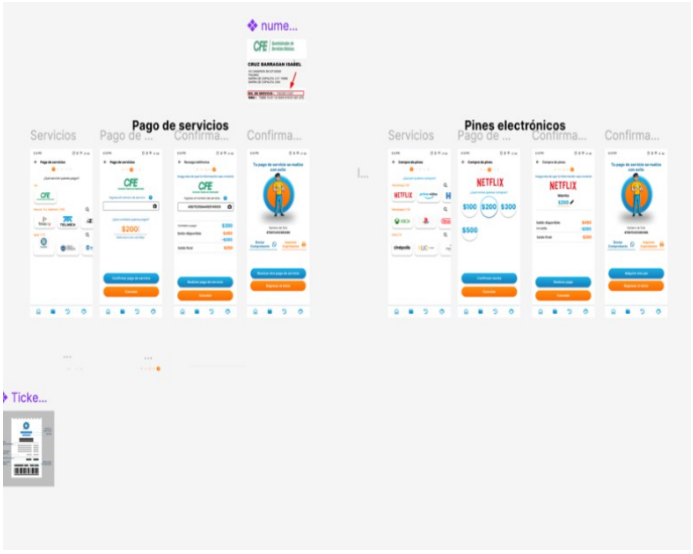

Ilustración 10 En esta sección muestra la venta de pines electrónicos y pago de servicios.

7.3 Instalación de Git.

7.3.1 Descargar Git.

Primero ir a la página oficial de Git: [git-scm.com](git-scm.com).

Después seleccionar la versión adecuada para tu sistema operativo (Windows, macOS, Linux) en mi caso fue Windows para descargar el instalador.

7.3.2 Instalación de Git.

- Ejecutar el instalador descargado.
- Seguir las instrucciones del asistente de instalación, aceptando las configuraciones predeterminadas o personalizando según tus preferencias.
- Completar la instalación y verificar que Git esté instalado correctamente abriendo una terminal y escribiendo **git --version**.

7.4 Guardar los cambios locales.

Para guardar los cambios realizados se utilizó la terminal y se utilizó el siguiente comando:

```
git add .
git commit -m "Cambio de colores de pantallas y botones"
```

Ilustración 11 Ilustración donde se muestra el cómo guardar los cambios realizados.

Por último para subir los cambios al repositorio remoto se hizo con el siguiente comando:

```
git push origin dev_newDesing
```

Ilustración 12 Se muestra cómo se guardan los cambios al repositorio.

7.5 Modelo Vista Controlador

1. **DTO:** Es una clase que que se ocupara ya que contiene datos del objeto por ejemplo en la App de Wip recargas en el apartado de usuarios contiene campos como el nombre, apellido, correo en otros campos de suma importancia entonces este modelo nos va a permitir insertar la información requerida.

2. **DAO:** Este modelo se usa ya que es la que nos permite interactuar con la BD, con ella podemos realizar operaciones como insertar usuario, modificar, obtener usuario o eliminar.

3. **VISTA:** Todas las actividades se utilizan como vistas por ejemplo MainActivity y LoginActivity las cuales pueden ser las vistas que muestren la interfaz de usuario.

8 Resultados.

En el proyecto de renovación de la interfaz de usuario de la aplicación Wiprecargas, se realizaron diversas mejoras y actualizaciones para modernizar su apariencia y mejorar la experiencia del usuario. Se seleccionó una nueva paleta de colores más vibrante y atractiva para hacer que la aplicación sea visualmente más atractiva y moderna. Estos colores se aplicaron a todos los elementos de la interfaz, mejorando la cohesión y el atractivo visual. Además, se incorporaron nuevas imágenes de alta calidad que complementan el diseño actualizado de la aplicación. Estas imágenes se utilizaron en diversas pantallas para mejorar la experiencia visual y proporcionar un contexto más rico. Los botones en la aplicación se rediseñaron para ser más intuitivos y visualmente atractivos. Se ajustaron los tamaños, colores y estilos de los botones para mejorar la

usabilidad y facilitar la interacción del usuario. También se implementó una nueva tipografía que mejora la legibilidad y aporta un aspecto más contemporáneo a la aplicación. Esto ayuda a que el texto sea más fácil de leer y a que la interfaz general se vea más moderna.

Se añadieron acciones y funcionalidades que faltaban en la versión anterior de la aplicación. Por ejemplo, se agregó una función de recuperación de contraseña, permitiendo a los usuarios recuperar el acceso a sus cuentas de manera sencilla y segura. Además, se desarrollaron nuevas pantallas para incluir las funcionalidades adicionales y mejorar la estructura de la aplicación.

Esta es la pantalla de inicio la cual muestra cambios muy significativos con nuevas imágenes, colores y tamaño de botones haciendo que se vea más llamativa.

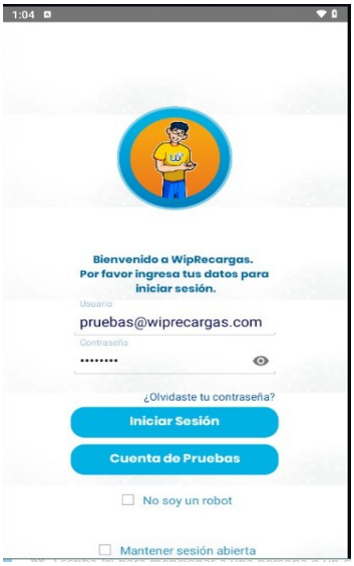

Ilustración 13 Pantalla nueva de inicio de sesión.

En esta pantalla muestra alertas ceste caso es cuando el usuario esta teniendo problemas de internet.

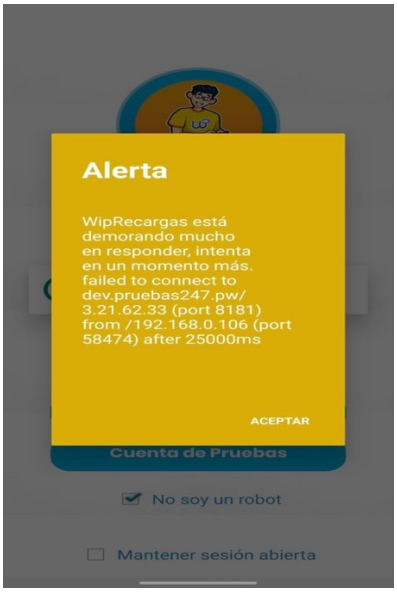

Ilustración 14 En esta pantalla muestra las alertas que muestra la app.

Se agregarón nuevas pantallas un ejemplo de ello es la pantalla de recuperación de contraseña.

Ilustración 15 Pantalla para recuperar contraseña.

Despues de agregar los campos que nos pide la pantalla anterior debemos de agregar el código de verificación que nos solicita para conformar que el usuario es quien solicita la recuperación de contraseña.

Ilustración 16 En esta pantalla nueva nos pide código de verificación para la recuperación de contraseña.

Esta es la nueva pantalla de registro de usuario la cual muestra cambios y es más llamativa para el usuario.

Ilustración 17 En esta nueva pantalla muestra los campos para registro de usuarios.

9 Conclusiones.

La renovación de la aplicación Wiprecargas no solo modernizó su apariencia visual, sino que también mejoró significativamente su usabilidad y funcionalidad, ofreciendo una experiencia de usuario más intuitiva y agradable. El rediseño visual incluyó una paleta de colores vibrante y atractiva, una tipografía moderna y una iconografía coherente, elementos que juntos crearon una interfaz más atractiva y profesional. Además, la integración de nuevas imágenes y la creación de nuevas pantallas añadieron profundidad y contexto visual a la aplicación, haciendo que la navegación sea más intuitiva y visualmente agradable.

En términos de usabilidad, la incorporación de nuevas funcionalidades, como la opción de recuperación de contraseñas, respondió directamente a las necesidades de los usuarios, aumentando la eficiencia y la accesibilidad de la aplicación. La reorganización de los botones y la optimización de los flujos de usuario facilitaron la interacción con la aplicación, reduciendo la complejidad y el tiempo necesario para realizar tareas comunes. Estas mejoras no solo hicieron que la aplicación fuera más fácil de usar, sino que también aumentaron la satisfacción del usuario, fomentando un uso más frecuente y prolongado. Desde el punto de vista técnico, la implementación de nuevas tecnologías y prácticas de desarrollo, como el uso de Git para el control de versiones y la utilización de Android Studio para el desarrollo y pruebas, garantizó que el proyecto se ejecutara de manera eficiente y estructurada. El uso de herramientas de prototipado como Figma permitió una planificación y una retroalimentación más efectivas durante el proceso de diseño, asegurando que todos los cambios estuvieran alineados con los objetivos del proyecto. Además, este proyecto me permitió aprender y dominar nuevas herramientas y tecnologías que serán de gran valor para futuros proyectos. La experiencia con Git y Android Studio, junto con el uso de emuladores como MEmu, no solo mejoró mis habilidades técnicas, sino que también amplió mi capacidad para gestionar y ejecutar proyectos complejos de manera más efectiva.

10Competencias desarrolladas.

Diseño de Interfaz de Usuario (UI) Se aplicaron conocimientos avanzados de diseño de interfaz de usuario para crear una apariencia más moderna y atractiva. Esto incluyó la selección de colores, tipografía, iconografía y elementos visuales que mejoran la experiencia del usuario.

Desarrollo de Software Se utilizaron habilidades en desarrollo de software, específicamente en el entorno de Android Studio, para implementar los cambios necesarios en la aplicación. Esto implicó la creación y modificación de actividades, fragmentos y recursos de la aplicación.

Control de Versiones con Git El uso de Git fue fundamental para el control de versiones del proyecto. Las competencias desarrolladas en este ámbito incluyeron la configuración de repositorios, la gestión de ramas, la resolución de conflictos y la colaboración en equipo a través de Git.

Gestión de Bases de Datos Se aplicaron conocimientos de gestión de bases de datos MySQL para asegurar que los cambios en la interfaz de usuario y las nuevas funcionalidades se integraran adecuadamente con la base de datos existente.

Prototipado con Figma La competencia en el uso de Figma permitió la creación de prototipos detallados que guiaron el proceso de desarrollo. Estos prototipos ayudaron a

visualizar los cambios propuestos y a obtener retroalimentación temprana sobre el diseño.

Emulación y Pruebas El uso de emuladores, específicamente MEmu, permitió realizar pruebas exhaustivas del proyecto en un entorno simulado. Esta competencia fue crucial para identificar y solucionar problemas antes del lanzamiento.

11Referencias.

(S/f). Git-scm.com. Recuperado el 26 de julio de 2024, de https://git-scm.com/downloads.

Avenida 5 de mayo #804, colonia centro, Apizaco Tlaxcala. C.P 90000 - Google Search. (n.d.). Google.com. Retrieved July 31, 2024, from https://www.google.com/search?client=safari&rls=en&q=Avenida+5+de+mayo+%23804%2C+colonia+centro%2C+Apizaco+Tlaxcala.++C.P+90000&ie=U TF-8&oe=UTF-8

Android Developers (2023). "Run apps on the Android Emulator". Google. developer.android.com.

Steele, J., To, N., & Conder, S. (2010). *The Android Developer's Cookbook: Building Applications with the Android SDK*. Addison-Wesley.

Mednieks, Z., Dornin, L., Meike, G., & Nakamura, M. (2012). *Programming Android.* O'Reilly Media.

Steele, J., To, N., & Conder, S. (2010). *The Android Developer's Cookbook: Building Applications with the Android SDK*. Addison-Wesley.

Scott Chacon and Ben Straub (2014). *Pro Git*. Apress.

Vincent Driessen (2010). "A Successful Git Branching Model". nvie.com.

Jon Loeliger and Matthew McCullough (2012). *Version Control with Git*. O'Reilly Media.

Chacon, S., & Straub, B. (2014). *Pro Git*. Apress.

Loeliger, J., & McCullough, M. (2012). *Version Control with Git*. O'Reilly Media.

Bryan O'Sullivan (2009). *Mercurial: The Definitive Guide*. O'Reilly Media.

Pilato, C. M., Collins-Sussman, B., & Fitzpatrick, B. W. (2008). *Version Control with Subversion*. O'Reilly Media.

Pressman, R. S. (2014). *Software Engineering: A Practitioner's Approach*. New York, NY: McGraw-Hill.

Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P., & Stal, M. (1996). *Pattern-Oriented Software Architecture Volume 1: A System of Patterns*. Chichester, UK: John Wiley & Sons.

Burnette, E. (2015). *Hello, Android: Introducing Google's Mobile Development Platform*. Dallas, TX: Pragmatic Bookshelf.

Murphy, M. L. (2018). *The Busy Coder's Guide to Android Development*. CommonsWare.