PROYECTO FIN DE CICLO FORMATIVO

IES GINÉS PÉREZ CHIRINOS
DEPARTAMENTO DE INFORMÁTICA

TÉCNICO SUPERIOR EN
DESARROLLO DE APLICACIONES MULTIPLATAFORMA

CryptoTrack

AUTOR(ES)
D. ANTONIO JUAN GONZÁLEZ-CONDE ABRIL

TUTOR
D. DANIEL JESUS MARTINEZ

CARAVACA DE LA CRUZ, 10 DE MARZO DE 2025

ABSTRACT

Abstract (Español)

Este proyecto es una plataforma que permite a los usuarios gestionar criptomonedas de manera simulada, es decir, sin usar dinero real. Los usuarios pueden crear cuentas con su correo electrónico y una contraseña. Al iniciar sesión, acceden a un panel que muestra el saldo total de sus criptomonedas. Pueden agregar distintas criptomonedas a su cuenta, ver cuánto valen y realizar transacciones simuladas, como enviar y recibir criptomonedas. La plataforma también guarda un historial de todas estas actividades. Aunque las transacciones no son reales, los precios de las criptomonedas se actualizan utilizando datos de una API gratuita como CoinRanking, ofreciendo una experiencia de simulación realista.

Abstract (English)

This project is a platform that allows users to manage cryptocurrencies in a simulated environment, meaning no real money is used. Users can create accounts with their email and a password. After logging in, they can see a dashboard that shows the total balance of their cryptocurrencies. They can add different cryptocurrencies to their account, see their value, and perform simulated transactions, such as sending and receiving cryptocurrencies. The platform also keeps a record of all these activities. Even though the transactions are not real, the cryptocurrency prices are updated using data from a free API like CoinRanking, providing a realistic simulation experience.

ÍNDICE GENERAL

1. INTRODUCCIÓN..... 1

2. ANÁLISIS DEL PROYECTO. 3

2.1. OBJETIVOS. 3
2.2. ANÁLISIS DE REQUISITOS. 4
2.3. MARCO DE TRABAJO Y HERRAMIENTAS. 5
2.4. PLANIFICACIÓN. 7

3. DISEÑO DEL PROYECTO. 11

3.1. CASOS DE USO. 11
3.2. ESTRUCTURA DEL PROYECTO. 13
3.3. INTERFACES GRÁFICAS. 17

4. DESARROLLO DEL PRODUCTO SOFTWARE. 21

4.1. PROGRAMACIÓN. 21
4.2. PRUEBAS. 36

5. MANUAL DE PUESTA EN MARCHA. 41

6. CONCLUSIONES Y VÍAS FUTURAS. 45

7. REFERENCIAS Y BIBLIOGRAFÍA. 47

1. INTRODUCCIÓN.

En un mundo cada vez más digitalizado, las criptomonedas se han convertido en una parte esencial de la economía moderna. Sin embargo, muchas personas aún encuentran desafiante entender y manejar estas monedas digitales. Nuestra plataforma ofrece una solución sencilla y segura para aquellos interesados en explorar el mundo de las criptomonedas sin riesgos financieros. A través de un entorno de simulación, los usuarios pueden aprender a gestionar su Cartera de criptomonedas, realizar transacciones simuladas y seguir los precios. Ya sea que seas un principiante que busca familiarizarse con el uso de criptomonedas o un entusiasta que quiere practicar sin comprometer fondos reales, nuestra plataforma es la herramienta ideal para adquirir experiencia en este dinámico mercado. Únete a nosotros y comienza tu viaje hacia la comprensión y dominio del mundo cripto.

2. ANÁLISIS DEL PROYECTO.

2.1. OBJETIVOS.

O1 - Proveer una Plataforma de Gestión de Criptomonedas Simulada.

O2 - La aplicación tendrá un registro y acceso de Usuarios.

O3 - Permitir la Gestión de Carteras de Criptomonedas.

O4 - Simular Transacciones de Criptomonedas.

O5 - Registrar y Visualizar Historial de Transacciones.

O6 - Proveer Datos de Precios de Criptomonedas.

O7 - Asegurar la protección y seguridad de los datos sensibles.

O8 - Permitir a los usuarios gestionar y modificar la configuración de su perfil.

Alcance del Proyecto

- **Usuarios**: Personas interesadas en aprender sobre criptomonedas, incluyendo principiantes y gente que busca practicar sin comprometer dinero real.
- **Funcionalidades**:
    - Registro de usuarios y autenticación.
    - Gestión de Carteras de criptomonedas, incluyendo agregar y visualizar monedas.
    - Simulación de transacciones (envío y solicitud de criptomonedas).
    - Visualización de una pantalla principal con el saldo total y métricas de la Cartera. (Valor Total de la cartera/Distribución de la cartera/Rentabilidad de la cartera)
    - Historial detallado de transacciones simuladas.
    - Actualización de precios de criptomonedas.

- **Limitaciones**:
    - Las transacciones son simuladas y no involucran dinero real.
    - Los datos de precios de criptomonedas se obtienen de fuentes gratuitas y pueden no reflejar el mercado en tiempo real con precisión absoluta.

La plataforma está diseñada con fines educativos y de práctica, no para operaciones financieras reales.

2.2. ANÁLISIS DE REQUISITOS.

REQUISITO FUNCIONAL	DESCRIPCIÓN	OBJETIVO ASOCIADO
RF1	Permitir a los usuarios crear una cuenta utilizando un correo electrónico válido y una contraseña segura.	O2
RF2	Permitir a los usuarios iniciar sesión con sus credenciales registradas	O2
RF3	Mostrar el saldo total de criptomonedas del usuario en la pantalla principal	O3
RF4	Proporcionar gráficos y estadísticas básicas sobre el rendimiento de la cartera del usuario.	O3
RF5	Permitir a los usuarios agregar criptomonedas a su Cartera desde una lista predefinida.	O3
RF6	Mostrar el balance y el valor actual de cada criptomoneda en la cartera del usuario.	O3
RF7	Permitir a los usuarios simular el envío de criptomonedas a una dirección ficticia.	O4
RF8	Mantener un registro detallado de todas las transacciones simuladas realizadas por el usuario.	O5
RF9	Proporcionar opciones para filtrar y ordenar las	O5

	transacciones por diferentes criterios (fecha, tipo, dirección).	
RF10	Integrar con una API como CoinRanking para obtener y mostrar los precios actuales de las criptomonedas.	O6
RF11	Implementar cifrado de contraseñas y datos sensibles para proteger la información del usuario.	O7
RF12	Asegurar que solo los usuarios autorizados puedan acceder a sus datos personales y transacciones simuladas.	O7
RF13	Proporcionar una opción para que los usuarios puedan eliminar su cuenta si lo desean.	O8
RF14	Ofrecer una opción para activar o desactivar la adición automática de nuevas monedas a la cartera.	O8

2.3. MARCO DE TRABAJO Y HERRAMIENTAS.

Lenguajes de Programación y Frameworks:

- **HTML:** Utilizado para estructurar el contenido de la aplicación web, definiendo la disposición y jerarquía de los elementos en cada pantalla, asegurando una presentación coherente y semánticamente correcta del contenido.

- **CSS:** Empleado para estilizar los elementos HTML y definir la apariencia visual de la interfaz de usuario, incluyendo colores, tipografías, márgenes, y disposición general.
- **Bootstrap:** Framework CSS utilizado para crear un diseño responsive que se adapte a diferentes dispositivos y tamaños de pantalla. Facilita la implementación de componentes como formularios, botones y barras de navegación, garantizando una interfaz de usuario consistente y moderna.
- **JavaScript:** Este lenguaje de programación se utiliza para añadir interactividad a la aplicación. Permite implementar eventos dinámicos, como clics y validaciones de formularios, así como actualizar contenido sin recargar la página. Además, en el contexto de React, se emplean métodos modernos como fetch o bibliotecas como Axios para manejar la comunicación asíncrona con el servidor.

Backend:

- **Node.js con ExpressJS:** Node.js, junto con el framework Express.js, permite manejar peticiones HTTP, gestionar rutas y procesar lógica del lado del servidor de manera eficiente. Se utiliza para desarrollar APIs RESTful que conectan el frontend con la base de datos y otros servicios.

Gestor de Bases de Datos:

- **MongoAtlas:** Base de datos NoSQL utilizada para almacenar eficientemente información del usuario, criptomonedas y transacciones.
- **Mongoose:** Biblioteca de JavaScript para MongoDB que proporciona una abstracción de alto nivel sobre la base de datos, facilitando el manejo de esquemas y validaciones de datos.

Autenticación y Seguridad:

- **JSON Web Tokens (JWT):** Se implementa para la autenticación local, almacenándose en cookies con las opciones httpOnly y secure.
- **bcryptjs:** Se utiliza para hashear las contraseñas antes de guardarlas en la base de datos, asegurando su seguridad.
- **express-validator:** Empleado para validar datos de entrada del usuario en el backend.

APIs Externas:

- **CoinRanking API:** API utilizada para obtener precios de criptomonedas en tiempo real, mejorando la experiencia del usuario al mostrar información actualizada.

Entornos de Desarrollo:

- **WebStorm:** Elegido como entorno de desarrollo integrado (IDE) para facilitar la edición, depuración y gestión del código.

Sistema Operativo:

- Compatible con cualquier sistema operativo, ya que se trata de una aplicación web.

ACTUALIZACIONES PARA LA VERSIÓN MÓVIL

Tecnologías Adicionales

- **Flutter con WebView**: Se utiliza Flutter para desarrollar la versión móvil de la aplicación. Mediante **WebView**, se carga la versión web de la aplicación dentro de la aplicación móvil, proporcionando una experiencia consistente entre plataformas.

- **WebView Integration**: Se habilita JavaScript en WebView para soportar la interactividad completa de la aplicación.

Permisos Android: Se han configurado los permisos necesarios en AndroidManifest.xml para permitir el acceso a internet y manejar tráfico HTTP

2.4. PLANIFICACIÓN.

Mes 1: Fase de Planificación y Desarrollo Inicial

Semana 1-2: Planificación y Diseño

- **Objetivo**: Establecer una base sólida para el desarrollo de la aplicación.
- **Tareas**:
  1. **Requisitos Detallados**: Finalizar la especificación de requisitos funcionales y no funcionales.
  2. **Arquitectura del Sistema**: Diseñar la arquitectura general del sistema, incluyendo la elección de tecnologías (Frontend, Backend, Base de Datos).
  3. **Diseño:** Crear maquetas y prototipos de la interfaz de usuario.

Semana 3-4: Configuración y Desarrollo Inicial del Backend

- **Objetivo**: Establecer el entorno de desarrollo y comenzar con la construcción del backend.
- **Tareas**:
  4. **Configuración del Entorno**: Configurar el entorno de desarrollo (Instalación de dependencias, repositorio Git).
  5. **Estructura del Proyecto Backend**: Configurar la estructura del proyecto utilizando Node.js/Express.
  6. **Implementación de Autenticación**: Desarrollar el sistema de registro e inicio de sesión de usuarios (JWT).
  7. **Configuración de la Base de Datos**: Diseñar e implementar el esquema de la base de datos (MongoDB).
  8. **Desarrollo de APIs Iniciales**: Crear las APIs básicas para la gestión de usuarios y criptomonedas.

Mes 2: Fase de Desarrollo Completo

Semana 5-6: Desarrollo del Backend y Funcionalidades Básicas

- **Objetivo**: Completar las funcionalidades principales del backend.
- **Tareas**:
    9. **Gestión de Criptomonedas**: Desarrollar funcionalidades para agregar, y visualizar criptomonedas en la cartera.
    10. **Simulación de Transacciones**: Implementar la lógica de simulación de envío y recepción de criptomonedas.
    11. **Historial de Transacciones**: Desarrollar la funcionalidad para registrar y visualizar el historial de transacciones.

Semana 7-8: Desarrollo del Frontend y Aplicación Móvil

- **Objetivo**: Construir la interfaz de usuario basada en los diseños anteriores y desarrollar una versión móvil funcional de la aplicación web.
- **Tareas**:
    12. **Configuración del Frontend**: Configurar el entorno de desarrollo del frontend (HTML, CSS, JavaScript).
    13. **Implementación de la Pantalla Principal**: Desarrollar la pantalla principal para mostrar el saldo total, estadísticas y la cartera de criptomonedas.
    14. **Interfaz de Gestión de la Cartera**: Crear la interfaz para agregar, visualizar.
    15. **Pantallas de Transacciones**: Implementar la interfaz para simular envíos, recepciones y visualizar el historial de transacciones.
    16. **Desarrollo de la Aplicación Móvil:**
        i. **Integración con Flutter y WebView:** Configurar un proyecto de Flutter para cargar la aplicación web mediante WebView, asegurando que la experiencia de usuario sea similar en dispositivos móviles.
        ii. **Ajustes de Diseño Responsive:** Optimizar el diseño utilizando CSS y Bootstrap para garantizar que todos los elementos sean accesibles y legibles en pantallas pequeñas.
        iii. **Pruebas en Dispositivos Reales:** Verificar el funcionamiento de la aplicación en dispositivos móviles para garantizar que los botones, enlaces y formularios respondan correctamente.**Conexión con el Backend en Render:** Configurar las solicitudes API para que apunten al backend desplegado en Render, garantizando el acceso a las funcionalidades de la aplicación.

Mes 3: Fase de Integración, Pruebas

Semana 9-10: Integración y Pruebas

- **Objetivo**: Asegurar que todos los componentes del sistema funcionen bien juntos.
- **Tareas**:
    17. **Integración de Frontend y Backend**: Conectar el frontend con el backend a través de las APIs desarrolladas.
    18. **Pruebas de Integración**: Realizar pruebas de integración para verificar que las funcionalidades principales funcionan como se espera.
    19. **Pruebas de Interfaz de Usuario**: Realizar pruebas de interfaz para asegurarse de que la UI/UX es intuitiva y funcional.

Semana 11: Optimización y Seguridad

- **Objetivo**: Optimizar el rendimiento de la aplicación y asegurar la protección de datos.
- **Tareas**:
    20. **Optimización del Código**: Revisar y optimizar el código del frontend y backend para mejorar el rendimiento.
    21. **Medidas de Seguridad**: Implementar cifrado de datos sensibles, asegurar las APIs, y realizar pruebas de penetración básicas.

Semana 12: Documentación Final

- **Objetivo**: Finalizar la documentación.
- **Tareas**:
    22. **Documentación Final**: Completar la documentación del proyecto, incluyendo el manual de usuario y la documentación técnica.
    23. **Presentación del Proyecto**: Preparar y realizar una presentación del proyecto.

- **Mes 1**: Planificación, diseño, configuración, y desarrollo inicial del backend.
- **Mes 2**: Desarrollo completo del backend y frontend, integrando funcionalidades clave.
- **Mes 3**: Integración, pruebas exhaustivas, optimización, y documentación final.

3. DISEÑO DEL PROYECTO.

3.1. CASOS DE USO.

A continuación, se describen las acciones e interacciones entre los actores y el sistema CryptoTrack:

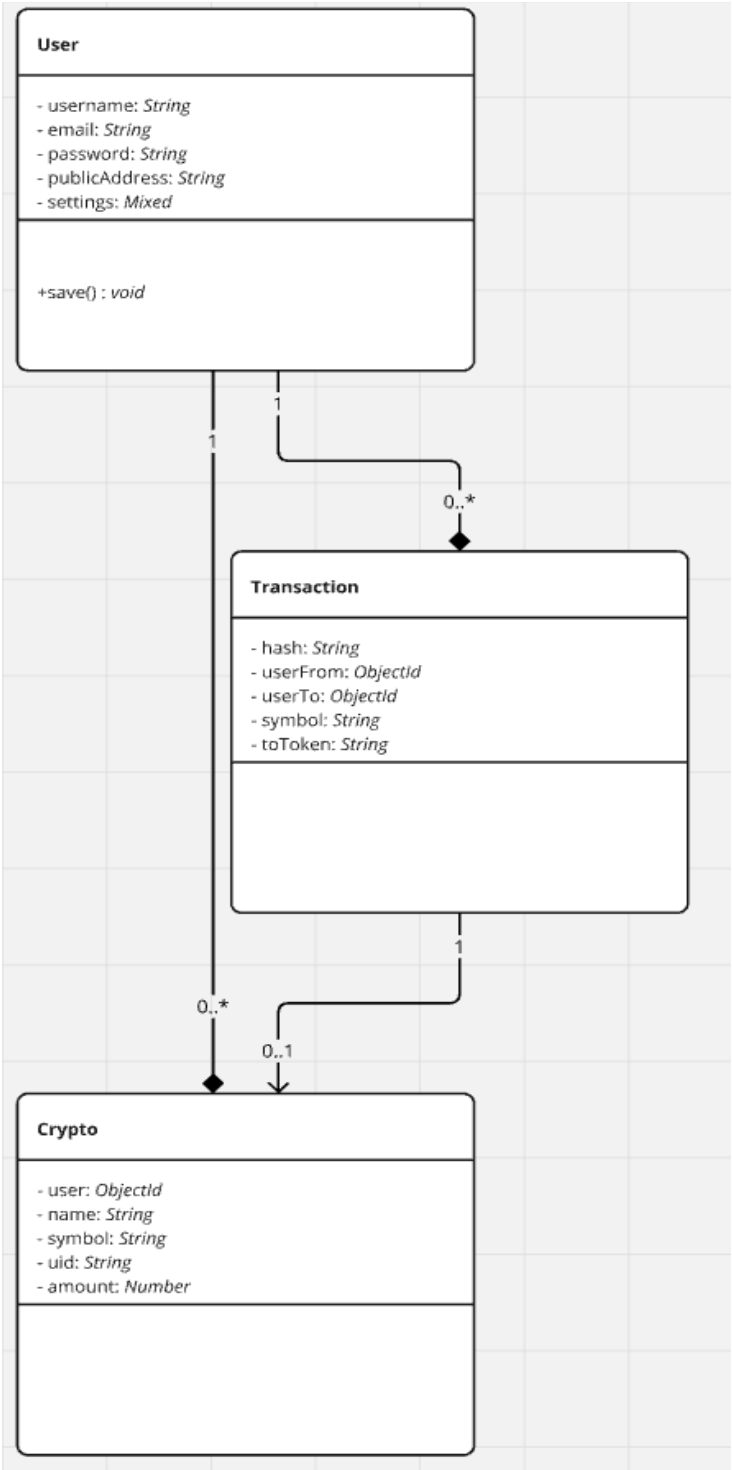
1. Registro de Usuario
   Requisito funcional: RF1
   Actor: Usuario no registrado
   Descripción: El sistema debe permitir a un nuevo usuario crear una cuenta.
   Comportamiento:
   - El sistema solicita al usuario ingresar un correo electrónico y una contraseña.
   - El sistema verifica que el correo electrónico no esté ya registrado.
   - El sistema crea una nueva cuenta de usuario si los datos son válidos.
2. Inicio de Sesión
   Requisito funcional: RF2
   Actor: Usuario registrado
   Descripción: El sistema debe permitir a un usuario registrado acceder a su cuenta.
   Comportamiento:
   - El sistema solicita al usuario sus credenciales (correo electrónico y contraseña).
   - El sistema verifica las credenciales y permite el acceso si son correctas.
3. Visualización de la Cartera
   Requisito funcional: RF3
   Actor: Usuario autenticado
   Descripción: El sistema debe mostrar al usuario el estado actual de su cartera de criptomonedas.
   Comportamiento:
   - El sistema muestra el saldo total en dólares.
   - El sistema presenta un desglose de las criptomonedas en la cartera.
   - El sistema muestra gráficos con la distribución y rentabilidad de la cartera.
4. Agregar Criptomoneda a la Cartera
   Requisito funcional: RF4
   Actor: Usuario autenticado
   Descripción: El sistema debe permitir al usuario agregar una nueva criptomoneda a su cartera.
   Comportamiento:
   - El sistema presenta una lista de criptomonedas disponibles.
   - El sistema permite al usuario seleccionar una criptomoneda y especificar la cantidad.
   - El sistema actualiza la cartera del usuario con la nueva criptomoneda.
5. Simular Envío de Criptomonedas
   Requisito funcional: RF5
   Actor: Usuario autenticado
   Descripción: El sistema debe permitir al usuario simular el envío de criptomonedas.
   Comportamiento:

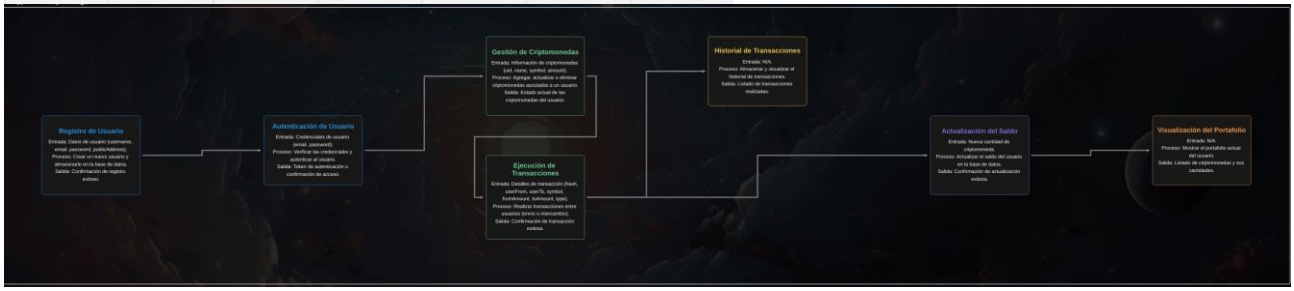
- El sistema solicita al usuario seleccionar la criptomoneda, la cantidad y la dirección del destinatario.
- El sistema verifica que el usuario tenga saldo suficiente.
- El sistema simula la transacción y actualiza los saldos correspondientes.

6. Simular Intercambio de Criptomonedas
   Requisito funcional: RF6
   Actor: Usuario autenticado
   Descripción: El sistema debe permitir al usuario simular el intercambio de una criptomoneda por otra.
   Comportamiento:
   - El sistema solicita al usuario seleccionar la criptomoneda de origen, la cantidad, y la criptomoneda de destino.
   - El sistema calcula y muestra la tasa de cambio.
   - El sistema simula el intercambio y actualiza los saldos correspondientes.
7. Visualizar Historial de Transacciones
   Requisito funcional: RF7
   Actor: Usuario autenticado
   Descripción: El sistema debe mostrar al usuario un historial de sus transacciones simuladas.
   Comportamiento:
   - El sistema presenta una lista de todas las transacciones realizadas por el usuario.
   - El sistema permite al usuario filtrar las transacciones por tipo o fecha.
8. Modificar Configuración de Usuario
   Requisito funcional: RF8
   Actor: Usuario autenticado
   Descripción: El sistema debe permitir al usuario ajustar la configuración de su cuenta.
   Comportamiento:
   - El sistema muestra las opciones de configuración disponibles.
   - El sistema permite al usuario activar o desactivar la opción de añadir automáticamente criptomonedas recibidas a la cartera.
   - El sistema guarda los cambios en la configuración del usuario.
9. Cerrar Sesión
   Requisito funcional: RF9
   Actor: Usuario autenticado
   Descripción: El sistema debe permitir al usuario cerrar su sesión actual.
   Comportamiento:
   - El sistema cierra la sesión del usuario.
   - El sistema redirige al usuario a la pantalla de inicio de sesión.
10. Eliminar Cuenta
    Requisito funcional: RF10
    Actor: Usuario autenticado
    Descripción: El sistema debe permitir al usuario eliminar permanentemente su cuenta.
    Comportamiento:
    - El sistema solicita confirmación al usuario.
    - El sistema elimina la cuenta y todos los datos asociados si se confirma la acción.

11. Actualización de Precios de Criptomonedas
    Requisito funcional: RF11
    Actor: Sistema
    Descripción: El sistema debe actualizar los precios de las criptomonedas periódicamente.
    Comportamiento:
    - El sistema se conecta a una API externa para obtener los precios actualizados.
    - El sistema actualiza los precios de las criptomonedas en la base de datos.
    - El sistema refleja los nuevos precios en la cartera de los usuarios.
12. Visualización de Gráficos de Rendimiento
    Requisito funcional: RF12
    Actor: Usuario autenticado
    Descripción: El sistema debe mostrar gráficos de rendimiento de las criptomonedas.
    Comportamiento:
    - El sistema genera gráficos basados en los datos históricos de precios.
    - El sistema muestra gráficos de rendimiento para cada criptomoneda en la cartera del usuario.
    - El sistema permite al usuario seleccionar diferentes intervalos de tiempo para los gráficos.
13. Notificaciones de Cambios Significativos
    Requisito funcional: RF13
    Actor: Sistema
    Descripción: El sistema debe notificar a los usuarios sobre cambios significativos en los precios.
    Comportamiento:
    - El sistema monitorea los cambios de precios de las criptomonedas.
    - El sistema identifica cambios que superen un umbral predefinido.
    - El sistema envía notificaciones a los usuarios afectados por estos cambios.
14. Simulación de Portafolio a Largo Plazo
    Requisito funcional: RF14
    Actor: Usuario autenticado
    Descripción: El sistema debe permitir al usuario simular el rendimiento de su portafolio a largo plazo.
    Comportamiento:
    - El sistema permite al usuario definir un horizonte temporal para la simulación.
    - El sistema utiliza datos históricos y proyecciones para simular el rendimiento futuro.
    - El sistema muestra los resultados de la simulación, incluyendo ganancias o pérdidas potenciales.

3.2. ESTRUCTURA DEL PROYECTO.





(https://mymap.ai/share/cryptocurrency-management-process-vQLXP0fkpXeNs)

En la aplicación de gestión de criptomonedas simulada, hemos diseñado varias pantallas clave que permiten a los usuarios interactuar con las diferentes funcionalidades del sistema. A continuación, se detallan las pantallas disponibles, su propósito y la navegación entre ellas.

Pantalla Principal.

- **Descripción**: Esta es la pantalla de inicio después de que el usuario haya iniciado sesión. Desde aquí, el usuario puede acceder a todas las funciones principales de la aplicación, como la visualización de la cartera, simulación de transacciones, y consulta del historial de transacciones.
- **Control de acceso**: Disponible solo para usuarios que han iniciado sesión.
- **Navegación**:
    - Desde esta pantalla, el usuario puede navegar hacia las pantallas de **Agregar Criptomonedas**, **Simular Transacciones**, **Historial de Transacciones**, **Ajustes de Usuario**.

Razonamiento del Diseño

En el diseño inicial de la pantalla principal, el objetivo era priorizar la usabilidad y el acceso rápido a las funciones principales.
Dado que el botón para agregar criptomonedas y el acceso a la cartera son de mayor uso, estos fueron situados en la barra de navegación. El uso de modales permitió mantener la interfaz limpia y organizada, evitando la sobrecarga visual y manteniendo la simetría en el diseño.

Además, la implementación de botones específicos para el perfil, configuración y cierre de sesión permite simplificar la experiencia del usuario, ofreciendo una interfaz adaptativa que se ajusta según el estado de la sesión.
Este enfoque asegura que la pantalla principal no solo sea funcional, sino también intuitiva y estéticamente agradable, proporcionando una experiencia de usuario fluida y eficiente.

Pantalla de Registro de Usuario

- **Descripción**: Permite a los nuevos usuarios crear una cuenta ingresando un correo electrónico y una contraseña segura.
- **Control de acceso**: Accesible desde la pantalla de inicio de sesión. No disponible para usuarios que ya han iniciado sesión.
- **Navegación**:

- Desde esta pantalla, el usuario puede volver a la **Pantalla de Inicio de Sesión**.

Pantalla de Inicio de Sesión

- **Descripción**: Permite a los usuarios registrados ingresar sus credenciales para acceder al sistema.
- **Control de acceso**: Accesible para todos los usuarios que aún no han iniciado sesión.
- **Navegación**:
  - Desde esta pantalla, el usuario puede acceder a la **Pantalla de Registro** o navegar a la **Pantalla Principal** después de iniciar sesión.

Evolución del Diseño

La decisión de utilizar un a única pantalla para tanto el inicio de sesión como el registro se tomó para simplificar la navegación y mejorar la experiencia del usuario. Inicialmente se consideraron dos pantallas separadas, pero se optó por una solución más eficiente que reduce la complejidad y proporciona una experiencia más fluida al usuario.

Pantalla de Ajustes de Usuario

- **Descripción**: Muestra el “publicAddress” del usuario y permite realizar cambios en la configuración de la cuenta como si al recibir una criptomoneda se añadirá automáticamente. También estarán los botones para cerrar sesión y para eliminar el usuario
- Control de acceso: Disponible solo para usuarios que han iniciado sesión.
- **Navegación**:
  - Desde esta pantalla, el usuario puede volver a la **Pantalla Principal** o **Cerrar Sesión**.

Modal de Agregar Criptomonedas

- **Descripción**: Permite al usuario seleccionar y agregar nuevas criptomonedas a su Cartera desde una lista predefinida.
- Control de acceso: Accesible desde la **Pantalla Principal**.
- **Navegación**:
  - Después de agregar una criptomoneda, el usuario es redirigido a la **Pantalla Principal**.

Modales de Envío o Cambio de Criptomonedas

- **Descripción**: Permite al usuario simular el envío, la recepción o permuta de criptomonedas. El usuario introduce la cantidad, selecciona la criptomoneda y proporciona una dirección ficticia de destinatario o el token *(Un **token** es un activo digital creado sobre una blockchain que puede representar una moneda, derecho, servicio o propiedad.)* a cambiar.
- Control de acceso: Accesible desde la **Pantalla Principal**.
- **Navegación**:
    - Después de completar la simulación de una transacción, el usuario puede regresar a la **Pantalla Principal**

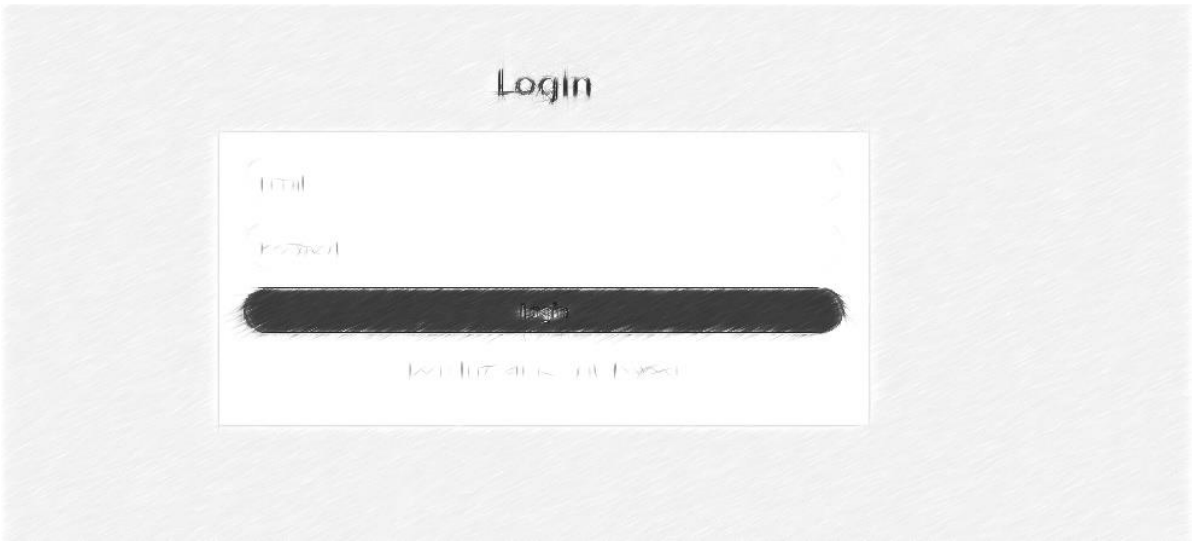
Modal de Historial de Transacciones

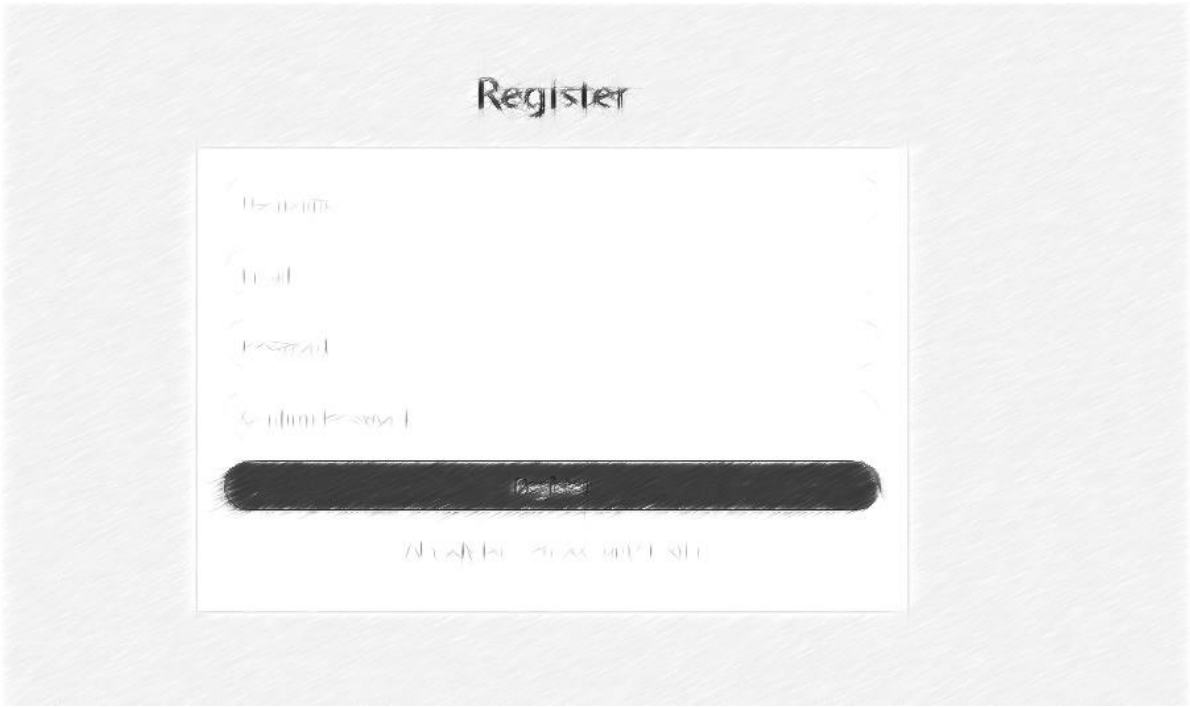
- **Descripción**: Muestra un registro detallado de todas las transacciones simuladas por el usuario, incluyendo la fecha, tipo de transacción, cantidad y criptomoneda involucrada.
- Control de acceso: Accesible desde la **Pantalla Principal**.
- **Navegación**:
    - Desde esta pantalla, el usuario puede regresar a la **Pantalla Principal**.

Navegación General

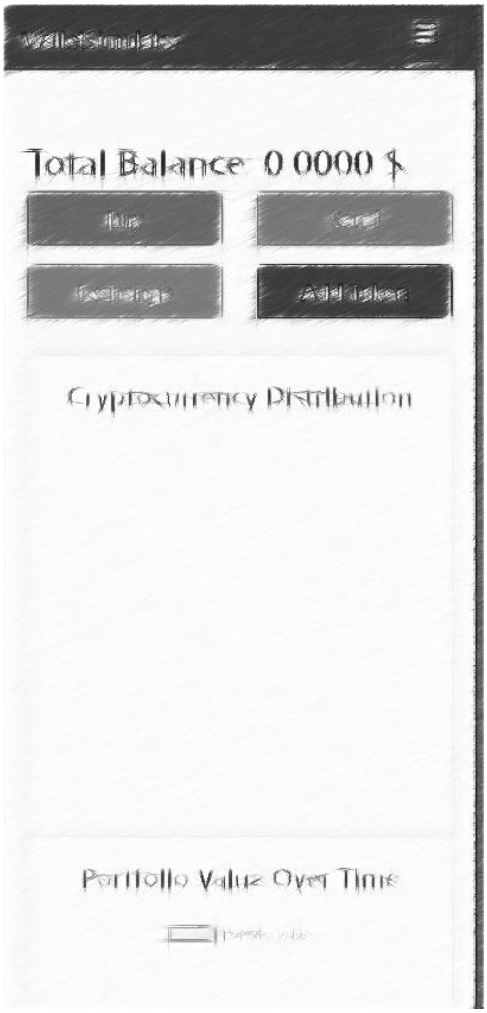
La estructura de navegación ha sido diseñada para ser intuitiva y permitir un acceso rápido a las funcionalidades clave de la aplicación. Todas las pantallas principales son accesibles desde la pantalla principal, y las pantallas de registro e inicio de sesión están diseñadas para ser intuitivas para el usuario.

3.3. INTERFACES GRÁFICAS.


Login


Register


Total Balance: 0.0000 $
Cryptocurrency Distribution
Portfolio Value Over Time


Total Balance: 0.0000 $
Cryptocurrency Distribution
Portfolio Value Over Time

4. DESARROLLO DEL PRODUCTO SOFTWARE.

4.1. PROGRAMACIÓN.

En este apartado, se describirán las tecnologías y lenguajes de programación empleados en cada una de las funcionalidades clave de la aplicación de gestión de criptomonedas. La intención es proporcionar una visión detallada del desarrollo del software, facilitando tanto el entendimiento del proyecto como sirviendo de guía para futuros desarrollos o mantenimientos. Este sistema ha sido diseñado como una aplicación web dinámica y segura que aprovecha una serie de tecnologías modernas para ofrecer una experiencia de usuario fluida y efectiva.

PANTALLA PRINCIPAL

Descripción General

La pantalla principal de la aplicación de gestión de criptomonedas es el punto de partida desde el cual los usuarios pueden acceder a todas las funcionalidades esenciales. Esta pantalla ha sido diseñada para ofrecer una visión clara y accesible del estado de la cartera del usuario, permitiendo una navegación intuitiva a las diferentes secciones de la aplicación.

Estructura del Archivo HTML

El archivo HTML que define la pantalla principal consta de las siguientes partes:

- **Encabezado**:
  - **Título**: En la parte superior de la pantalla, se presenta un título que refleja el nombre de la aplicación.
  - **Botones de Navegación**: Se encuentran dos botones en una navbar:
    - **Botón de Transacciones**: Abre la pantalla donde el usuario puede buscar y ver transacciones.
    - **Botón de Ajustes**: Abre la pantalla de ajustes de la cuenta del usuario:
- **Cuerpo**:
  - **Visión General de la cartera**: En el centro de la pantalla, se muestra un resumen del saldo total del usuario en diversas criptomonedas, junto con los valores actuales. También 2 gráficos con información importante de la cartera
  - **Simulación de Transacciones**: Si el usuario tiene criptomonedas en su Cartera, mediante los botones podrá iniciar distintas simulaciones de transacciones.

MODAL DE AJUSTES DE USUARIO

Descripción General

La pantalla de perfil de usuario en la aplicación de gestión de criptomonedas está diseñada para permitir a los usuarios visualizar su información personal básica y gestionar su sesión. Este espacio proporciona acceso directo a las funciones de cierre de sesión y es un punto crucial para la seguridad y personalización de la experiencia del usuario.

Estructura del Archivo HTML

El archivo HTML que define la pantalla de perfil de usuario incluye las siguientes secciones:

- **Encabezado**:
  - **Título**: Un título claro que indica que el usuario se encuentra en su perfil.
- **Cuerpo**:
  - **Información del Usuario**:
    - **Dirección Pública**: Muestra la dirección de la cartera del usuario que actualmente está en sesión.
  - **Opciones de Configuración**:
    - **Añadir Criptomonedas Automáticamente**: Permite al usuario activar o desactivar la opción de añadir criptomonedas automáticamente a su Cartera.
  - **Botones de Acción**
    - **Botón de Cerrar Sesión:** Permite al usuario finalizar su sesión de manera segura.
    - **Botón de Borrar Cuenta:** Permite al usuario eliminar su cuenta de manera segura.

PANTALLA DE INICIO DE SESIÓN REGISTRO

Descripción General

La pantalla de inicio de sesión y registro es una única interfaz que permite tanto a los usuarios existentes iniciar sesión como a los nuevos usuarios crear una cuenta. La pantalla se adapta dinámicamente en función de la acción que el usuario desea realizar, mostrando los campos y botones correspondientes para cada operación.

Estructura del Archivo HTML

El archivo HTML que define la pantalla de inicio de sesión y registro consta de las siguientes secciones:

- **Encabezado**:
  - **Título**: Muestra un título descriptivo que cambia según el estado de la pantalla, como "Login" o "Register".
- **Cuerpo**:
  - **Formulario de Inicio de Sesión**:
    - **Campos**:
      - **Campo de Nombre de Correo Electrónico**: Para que el usuario introduzca su correo electrónico.
      - **Campo de Contraseña**: Campo para introducir la contraseña.
    - **Botón de Iniciar Sesión**: Envía los datos introducidos para la verificación.
    - **Botón de Registro**: Cambia la interfaz para mostrar el formulario de registro si el usuario no tiene una cuenta.
  - **Formulario de Registro** (cuando se selecciona la opción de registro):
    - **Campos**:
      - **Campo de Nombre de Usuario**: Para que el usuario introduzca un nombre de usuario único.
      - **Campo de Correo Electrónico**: Para la dirección de correo electrónico del usuario.
      - **Campo de Contraseña**: Campo para introducir la contraseña.
      - **Campo de Confirmación de Contraseña**: Para que el usuario confirme la contraseña introducida.
    - **Botón de Registrarse**: Envía los datos del formulario para crear una nueva cuenta.
    - **Botón de Inicio de Sesión**: Cambia la interfaz para mostrar el formulario de inicio de sesión si el usuario ya tiene una cuenta.

Gestión de Criptomonedas

La gestión de criptomonedas es una funcionalidad central de la aplicación, permitiendo a los usuarios visualizar y gestionar sus activos digitales de manera eficiente. A continuación, se detalla la implementación de las distintas pantallas y funcionalidades relacionadas con la gestión de criptomonedas.

PANTALLA PRINCIPAL

La pantalla principal proporciona una vista general del saldo total del usuario en las diversas criptomonedas que posee. Está diseñado para ofrecer una representación clara y concisa del valor total de la cartera del usuario.

Estructura HTML:

- **Encabezado**:
  - Un título que indica la Pantalla Principal
- **Cuerpo**:
  - Un resumen del valor total de la cartera en dólares.
  - Botones para realizar acciones rápidas como comprar, enviar, intercambiar y agregar tokens.
  - Gráficos o indicadores visuales que resumen el rendimiento de la cartera.

PANTALLA PARA AGREGAR CRIPTOMONEDAS

La función de **Agregar Criptomonedas** permite a los usuarios incorporar nuevas criptomonedas a su cartera desde una lista predefinida.

Estructura HTML:

- **Encabezado**:
  - Un título que indica la sección de "Agregar Criptomonedas".
- **Cuerpo**:
  - Un menú desplegable o lista de selección que permite al usuario elegir entre varias criptomonedas disponibles.
  - Un botón para confirmar la adición de la criptomoneda a la cartera.

PANTALLA PARA ENVIAR CRIPTOMONEDAS

La función de **Enviar Criptomonedas** simula el proceso de enviar criptomonedas desde la cartera del usuario a otra dirección (usuario).

Estructura HTML:

- **Encabezado**:
  - Un título que indica la sección de "Enviar Criptomonedas".
  - Un botón para navegar hacia atrás, ubicado en la parte superior izquierda.
- **Cuerpo**:
  - Un campo de selección para que el usuario elija la criptomoneda que desea enviar.

- Un campo de entrada para la cantidad de criptomonedas a enviar.
- Un campo de entrada para la dirección del destinatario.
- Un botón para confirmar la transacción.

PANTALLA DE HISTORIAL DE TRANSACCIONES

La sección de **Historial de Transacciones** proporciona un registro detallado de todas las transacciones simuladas realizadas por todos los usuarios.

Estructura HTML:

- **Encabezado**:
  - Un título que indica la sección de "Historial de Transacciones".
- **Cuerpo**:
  - Una tabla o lista que muestra cada transacción realizada, con las siguientes columnas:
    - **Fecha**: Fecha y hora en que se realizó la transacción.
    - **Tipo de Transacción**: Indica si la transacción fue un envío o un “exchange”.
    - **Cantidad**: La cantidad de criptomonedas involucradas en la transacción.
    - **Criptomoneda**: La criptomoneda transferida.
    - **Dirección Destinataria/Remitente**: Dirección involucrada en la transacción (simulada).
    - **Hash**: Identificador para cada transacción, único y requerido.

PRECIOS DE CRIPTOMONEDAS

La funcionalidad de actualización de precios es crucial para proporcionar a los usuarios una visión actualizada y precisa del valor de sus criptomonedas. Esta característica se implementa mediante la integración con una API externa, CoinRanking, que proporciona datos de mercado.

Detalles Técnicos:

- **Integración API:**
  - La API de CoinRanking se integra utilizando claves de API específicas que permiten la autenticación y el acceso a los datos de precios. Estos precios son actualizados y enviados al frontend donde se muestran de forma dinámica.
- **Seguridad y Gestión de Claves:**
  - Las claves de la API se gestionan de forma segura, almacenándolas en variables de entorno en el backend para evitar su exposición en el cliente.

DESARROLLO DE LA APLICACIÓN MOVIL CON WEBVIEW Y FLUTTER

Introducción

Como parte de la expansión del proyecto y el desarrollo de la aplicación móvil, se implementaron dos enfoques complementarios para ofrecer una experiencia optimizada y accesible en dispositivos Android:

1. Progressive Web App (PWA): Se desarrolló una PWA para aprovechar ventajas como la capacidad de funcionar offline, la facilidad de actualización y el menor uso de almacenamiento. Esta permite a los usuarios acceder a Wallet Simulator directamente desde el navegador de su dispositivo Android, con la opción de añadir un acceso directo a la pantalla de inicio.
2. Aplicación Flutter con WebView: Se construyó una aplicación móvil utilizando Flutter, un framework multiplataforma, en conjunto con el paquete WebView. Esto permitió integrar la aplicación web existente dentro de un entorno móvil nativo.

Ambos enfoques se centraron en reutilizar la funcionalidad del frontend web, garantizando una experiencia consistente entre las versiones web y móvil. Esta estrategia dual ofrece a los usuarios flexibilidad en la forma de acceder y utilizar la aplicación en sus dispositivos móviles, complementando la implementación web y proporcionando opciones adaptadas a diferentes preferencias de uso.

Proceso de Desarrollo

1. **Objetivo del Proyecto Móvil** El propósito de la aplicación móvil fue ofrecer a los usuarios una forma eficiente y directa de interactuar con Wallet Simulator desde sus dispositivos Android. Al reutilizar el frontend de la aplicación web a través de WebView, se logró mantener una experiencia visual y funcional consistente.
2. **Arquitectura del Proyecto**
    - La aplicación móvil está basada en una estructura de Flutter básica con una vista única (WebView) que carga la interfaz de usuario desde la URL del proyecto web desplegado en Render.
    - Se optó por este enfoque para reducir tiempos de desarrollo y mantener sincronizados los cambios entre la versión web y la móvil.
3. **Integración de WebView**
    - El paquete **webview_flutter** fue utilizado para integrar la WebView dentro del proyecto Flutter.
    - La WebView se configuró para cargar la URL principal de la aplicación web (https://walletsimulator.onrender.com) y permitir la ejecución de JavaScript, necesaria para el funcionamiento completo del backend.

4. **Generación de la APK**
    - Una vez desarrollada la aplicación móvil, se compiló y empaquetó en un archivo APK, apto para dispositivos Android. Esto permitió realizar pruebas en hardware real y asegurar el correcto funcionamiento de la aplicación.
5. **Pruebas y Validación**
    - Se llevaron a cabo pruebas funcionales en dispositivos Android para garantizar la interacción correcta con el backend a través de las APIs.
    - Se validaron aspectos clave, como el inicio de sesión, la visualización de datos de la cartera de criptomonedas y la realización de simulaciones de transacciones.

DESPLIEGUE DEL BACKEND

El backend se ha desplegado exitosamente en **Render.com**, lo que permite que sea accesible desde cualquier dispositivo conectado a internet, incluyendo la aplicación móvil.

1. URL del Backend: El WebView y las solicitudes API de la aplicación móvil están configurados para apuntar a la URL pública del servicio desplegado (https://walletsimulator.onrender.com).
2. Configuración CORS: El backend ha sido configurado para aceptar solicitudes desde cualquier origen, incluyendo la aplicación móvil.

DISEÑO DE LA API HTTP

La API RESTful diseñada para este proyecto proporciona la infraestructura necesaria para interactuar con la gestión de usuarios, criptomonedas y transacciones simuladas. Aunque sigue muchos principios RESTful, no es completamente RESTful debido a ciertas decisiones de diseño que priorizan la funcionalidad y la simplicidad sobre la adherencia estricta a los principios REST. A continuación, se explican las rutas y los métodos disponibles, junto con una reflexión sobre su alineación con los principios REST.

¿Por qué no es completamente RESTful?

1. **Uso de verbos en las rutas:**
    a. Algunas rutas incluyen verbos como /updateBalance, /send, o /exchange, lo cual no es estrictamente RESTful. En una API RESTful pura, las operaciones deberían representarse como acciones sobre recursos (por ejemplo, POST /transactions para crear una transacción).
2. **Rutas específicas para operaciones concretas:**
    a. Rutas como /verifyToken o /balanceHistory están diseñadas para tareas muy específicas que podrían haberse integrado en recursos más generales.
3. **Estado de sesión:**
    a. Aunque se utiliza autenticación basada en tokens (JWT), el manejo de sesiones mediante cookies introduce un estado en el cliente, lo cual contradice el principio REST de ser completamente stateless.
4. **Decisiones pragmáticas:**
    a. Estas decisiones se tomaron para mejorar la claridad, usabilidad y funcionalidad de la API. Como resultado, aunque no es 100% RESTful, sigue siendo efectiva y fácil de usar.

Códigos de Estado

- **200 OK:** Solicitud completada con éxito.
- **201 Created:** Solicitud exitosa y ha resultado en la creación de un nuevo recurso.
- **400 Bad Request:** Error del cliente, solicitud mal formada.
- **401 Unauthorized:** Se requiere autenticación válida para acceder al recurso.
- **404 Not Found:** Recurso solicitado no encontrado.
- **409 Conflict:** Conflicto con el estado actual del recurso.
- **500 Internal Server Error:** Error inesperado en el servidor que impidió completar la solicitud.

Operaciones de la API

Gestión de Usuarios

Registro

- **Método:** POST
- **Ruta:** api/users/register
- **Descripción:** Registra un nuevo usuario en el sistema.
- **Cuerpo de la petición:**

```
{
  "username": "string",
  "email": "string",
  "password": "string"
}
```

- **Respuestas:**
  - **201 Created:** Usuario registrado correctamente.
  - **400 Bad Request**: Correo electrónico inválido, dominio de correo no permitido, usuario ya existe o correo ya registrado.
  - **500 Internal Server Error:** Error interno del servidor.

Iniciar Sesión

- **Método:** POST
- **Ruta:** api/users/login
- **Descripción:** Inicia sesión un usuario existente.
- **Cuerpo de la petición:**

```
{
  "username": "string",
  "email": "string",
  "password": "string"
}
```

- **Respuestas:**
  - **200 OK:** Inicio de sesión exitoso, devuelve un token de autenticación.
  - **400 Bad Request**: Error al cerrar la sesión.
  - **500 Internal Server Error:** Error interno del servidor.

Obtener detalles del Usuario Actual

- **Método:** GET
- **Ruta:** api/users/me
- **Descripción:** Obtiene los detalles del usuario autenticado.
- **Autenticación**: Requiere token JWT en el encabezado de la solicitud.
- **Encabezado**: Authorization: Bearer [token]
- **Respuestas:**
  - **200 OK:** Devuelve los detalles del usuario.
  - **401 Unauthorized:** Usuario no autenticado.
  - **500 Internal Server Error:** Error interno del servidor.

Actualizar perfil del Usuario Actual

- **Método:** PUT
- **Ruta:** api/users/me
- **Descripción:** Actualiza el perfil del usuario autenticado.
- **Autenticación**: Requiere token JWT en el encabezado de la solicitud.
- **Encabezado**: Authorization: Bearer [token]
- **Respuestas:**
  - **200 OK:** Perfil actualizado correctamente.
  - **401 Unauthorized:** Usuario no autenticado.
  - **500 Internal Server Error:** Error interno del servidor.

Eliminar Usuario Actual

- **Método:** DELETE
- **Ruta:** api/users/me
- **Descripción:** Elimina el perfil del usuario autenticado.

- **Autenticación**: Requiere token JWT en el encabezado de la solicitud.
- **Encabezado**: Authorization: Bearer [token]
- **Respuestas:**
  - **200 OK:** Usuario eliminado correctamente.
  - **401 Unauthorized:** Usuario no autenticado.
  - **500 Internal Server Error:** Error interno del servidor.

Actualizar Balence

- **Método:** POST
- **Ruta:** /api/users/updateBalance
- **Descripción:** Actualiza el balance del usuario y registra el historial.
- **Cuerpo de la petición:**

```
{
  "balance": "number",
 }
```

- **Autenticación**: Requiere token JWT en el encabezado de la solicitud.
- **Encabezado**: Authorization: Bearer [token]
- **Cuerpo de la Solicitud:**

```
{
  "balance": "number"
}
```

- **Respuestas:**
  - **200 OK:** Balance actualizado correctamente.
  - **401 Unauthorized:** Usuario no autenticado.
  - **500 Internal Server Error:** Error al actualizar el balance.

Historial del Balance

- Método: GET
- Ruta: /api/users/balanceHistory
- Descripción: Obtiene el historial de balance del usuario.
- **Autenticación**: Requiere token JWT en el encabezado de la solicitud.
- **Encabezado**: Authorization: Bearer [token]
- Respuestas:
  - 200 OK: Devuelve el historial de balance.
  - 401 Unauthorized: Usuario no autenticado.
  - 500 Internal Server Error: Error al obtener el historial.

Gestión de Criptomonedas

Obtener lista de criptomonedas

- **Método**: GET
- **Ruta**: /api/crypto/cryptocurrencies
- **Descripción**: Devuelve una lista de todas las criptomonedas disponibles en el sistema.
- **Cuerpo de la Solicitud:**
- {
  "balance": "number"
  }
- **Respuestas**:
  - **200 OK**: Devuelve la lista de criptomonedas.
  - **500 Internal Server Error**: Error al recuperar la lista.

Obtener detalles de una Criptomoneda

- **Método:** GET
- **Ruta:** /api/crypto/cryptocurrencies/:uid
- **Descripción:** Devuelve los detalles de una criptomoneda específica.
- **Cuerpo de la Solicitud:**
- **Respuestas:**
  - **200 OK:** Devuelve los detalles de la criptomoneda.
  - **500 Internal Server Error**: Error al recuperar los detalles de la criptomoneda.

Agregar una Criptomoneda a la cartera

- **Método:** POST
- **Ruta:** /api/crypto
- **Descripción:** Agrega una criptomoneda a la cartera del usuario.
- **Cuerpo de la Solicitud:**
  {
  "uid": "string"
  "name": "string"
  "symbol": "string"
  "amount": "number"
  }
- **Respuestas:**
  - **201 Created:** Criptomoneda agregada correctamente.
  - **400 Bad Request:** Solicitud mal formada o criptomoneda no válida.
  - **500 Internal Server Error:** Error al agregar la criptomoneda.

Visualizar la cartera de Criptomonedas

- **Método:** GET
- **Ruta:** /api/crypto
- **Descripción:** Devuelve la cartera de criptomonedas del usuario.
- **Respuestas:**
  - **200 OK:** Devuelve la cartera de criptomonedas.
  - **500 Internal Server Error**: Error al recuperar la cartera.

Agregar una Criptomoneda a la cartera

- **Método:** POST
- **Ruta:** /api/crypto/cryptocurrencies
- **Descripción:** Agrega una nueva criptomoneda al sistema.
- **Autenticación**: Requiere token JWT en el encabezado de la solicitud.
- **Encabezado**: Authorization: Bearer [token]
- **Cuerpo de la Solicitud:**

```
{
  "uid": "string",
}
```

- **Respuestas:**
  - **201 Created:** Criptomoneda agregada correctamente.
  - **400 Bad Request:** Datos de criptomoneda inválidos.
  - **401 Unauthorized:** Usuario no autenticado.
  - **500 Internal Server Error:** Error al agregar la criptomoneda.

Simulación de Transacciones

Enviar Criptomonedas

- **Método:** POST
- **Ruta**: /api/transactions/send
- **Descripción**: Envía criptomonedas de un usuario a otro.
- **Cuerpo de la Solicitud:**

```
{
  "symbol": "string"
  "amount": "number"
  "receiverAddress": "string"
}
```

- **Respuestas**:
  - **200 OK:** Transacción exitosa.
  - **400 Bad Request:** Fondos insuficientes o criptomoneda no encontrada.
  - **404 Not Found:** Receptor no encontrado.
  - **500 Internal Server Error:** Error al realizar la transacción.

Cambiar Criptomonedas

- **Método:** POST
- **Ruta:** /api/transactions/exchange
- **Descripción:** Cambia una criptomoneda por otra.
- **Cuerpo de la Solicitud:**
  ```
  {
    "fromToken": "string"
    "toToken": "string"
    "amount": "number"
  }
  ```
- **Respuestas:**
  - **200 OK:** Cambio realizado correctamente.
  - **404 Not Found:** Criptomoneda no encontrada.
  - **500 Internal Server Error:** Error al realizar el cambio.

Confirmar intercambio y registrar transacción

- **Método:** POST
- **Ruta:** /api/transactions/confirm
- **Descripción:** Confirma el intercambio de criptomonedas y registra la transacción en la base de datos.
- **Cuerpo de la Solicitud:**
  ```
  {
    "fromToken": "string"
    "toToken": "string"
    "amount": "number"
    "exchangedAmount": "number"
  }
  ```
- **Respuestas:**
  - **200 OK:** Intercambio confirmado y transacción registrada.
  - **404 Not Found:** Criptomoneda no encontrada en la cartera del usuario.
  - **500 Internal Server Error:** Error al procesar la transacción.

Obtener transacciones con filtros opcionales

- **Método:** GET
- **Ruta:** /api/transactions/user-transactions

- **Descripción:** Devuelve las transacciones del usuario autenticado con filtros opcionales.
- **Respuestas:**
  - **200 OK:** Devuelve las transacciones filtradas.
  - **500 Internal Server Error:** Error al recuperar las transacciones.

Autenticación y manejo de sesiones

Verificar Token

- **Método:** GET
- **Ruta:** /api/auth/verifyToken
- **Descripción:** Verifica si el token de autenticación es válido.
- **Autenticación**: Requiere token JWT en el encabezado de la solicitud.
- **Encabezado**: Authorization: Bearer [token]
- **Respuestas:**
  - **200 OK:** El token es válido.
  - **401 Unauthorized:** No se proporcionó token o el token no es válido.

Cerrar sesión

- **Método:** POST
- **Ruta:** /api/auth/logout
- **Descripción:** Cierra la sesión del usuario autenticado.
- **Respuestas:**
  - **200 OK:** Sesión cerrada correctamente.
  - **500 Internal Server Error:** Error al cerrar la sesión.

Obtener todas las transacciones

- **Método:** GET
- **Ruta:** /api/transactions
- **Descripción:** Obtiene todas las transacciones del usuario.
- **Autenticación**: Requiere token JWT en el encabezado de la solicitud.
- **Encabezado**: Authorization: Bearer [token]
- **Respuestas:**
  - **200 OK:** Devuelve la lista de transacciones.
  - **401 Unauthorized:** Usuario no autenticado.
  - **500 Internal Server Error:** Error al obtener las transacciones.

Configuración del Usuario

Guardar configuración

- **Método:** PUT
- **Ruta:** /api/settings/saveSettings
- **Descripción:** Actualiza la configuración del usuario.
- **Autenticación**: Requiere token JWT en el encabezado de la solicitud.
- **Encabezado**: Authorization: Bearer [token]
- **Respuestas:**
  - **200 OK:** Configuración actualizada correctamente.
  - **401 Unauthorized:** Usuario no autenticado.
  - **500 Internal Server Error:** Error al actualizar la configuración.

Obtener configuración

- **Método:** GET
- **Ruta:** /api/settings/getSettings
- **Descripción:** Obtiene la configuración actual del usuario.
- **Autenticación**: Requiere token JWT en el encabezado de la solicitud.
- **Encabezado**: Authorization: Bearer [token]
- **Respuestas:**
  - **200 OK:** Devuelve la configuración del usuario.
  - **401 Unauthorized:** Usuario no autenticado.
  - **500 Internal Server Error:** Error al obtener la configuración.

Verificar Token

- **Método:** GET
- **Ruta:** /api/auth/verifyToken
- **Descripción:** Verifica la validez del token de autenticación.
- **Encabezados:**
  - Authorization: Bearer [token]
- **Respuestas:**
  - **200 OK:** Token válido.
  - **401 Unauthorized:** Token inválido o expirado.
  - **500 Internal Server Error:** Error al verificar el token.

Middleware de autenticación

Middleware que verifica la validez del token JWT en las cookies de la solicitud. Si el token es válido, añade la información del usuario a la solicitud y permite el acceso a las rutas protegidas.

4.2. PRUEBAS.

La sección de pruebas está diseñada para garantizar que cada funcionalidad de la aplicación funcione correctamente y cumpla con los requisitos especificados. A continuación, se presentan los casos de prueba para diferentes áreas de la aplicación.

Caso de Prueba: Gestión de Criptomonedas

Título del Caso de Prueba: Verificar la visualización de la cartera y el manejo de criptomonedas.

Requisito funcional: RF3, RF5, RF6

Escenario 1: Visualización de la cartera

- **Paso:** Iniciar la aplicación y navegar a la pantalla principal.
  - **Resultado Esperado:** La pantalla principal se carga correctamente.
- **Paso:** Identificar y seleccionar el botón para ver la cartera de criptomonedas.
  - **Resultado Esperado:** La pantalla de Cartera muestra el balance y el valor actual de cada criptomoneda.

Escenario 2: Agregar Criptomonedas a la cartera

- **Paso:** Navegar a la pantalla de gestión de criptomonedas.
  - **Resultado Esperado:** La pantalla de gestión de criptomonedas se muestra correctamente.
- **Paso:** Seleccionar una criptomoneda de la lista predefinida para agregar.
  - **Resultado Esperado:** La criptomoneda seleccionada se agrega a la cartera del usuario.
- **Paso:** Verificar que el balance y valor de la criptomoneda agregada se reflejan en la pantalla de Cartera.
  - **Resultado Esperado:** El balance y valor de la criptomoneda se actualizan correctamente en la cartera.

Escenario 3: Agregar Criptomonedas

- **Paso:** Navegar a la pantalla de gestión de criptomonedas.
  - **Resultado Esperado:** La pantalla de gestión se carga correctamente.
- **Paso:** Seleccionar una criptomoneda de la lista predefinida para agregar y especificar la cantidad.
  - **Resultado Esperado:** La criptomoneda se agrega a la cartera y el balance se actualiza correctamente.
- **Paso:** Verificar que la criptomoneda recién agregada aparece en la pantalla de Cartera con el balance correcto.
  - **Resultado Esperado:** La criptomoneda y su balance se reflejan correctamente en la pantalla de Cartera.

Caso de Prueba: Simulación de Transacciones

Título del Caso de Prueba: Verificar el envío de criptomonedas y el historial de transacciones.

Requisito funcional: RF7, RF8, RF9

Escenario 1: Enviar Criptomonedas

- **Paso:** Navegar a la pantalla de simulación de envío de criptomonedas.
  - **Resultado Esperado:** La pantalla de simulación de envío se muestra correctamente.
- **Paso:** Introducir los detalles de la transacción (usuario receptor, cantidad, criptomoneda).
  - **Resultado Esperado:** La transacción se simula correctamente y el saldo se actualiza.
- **Paso:** Verificar que la transacción aparece en el historial de transacciones.
  - **Resultado Esperado:** La transacción se muestra en el historial con los detalles correctos.

Escenario 3: Ver Historial de Transacciones

- **Paso:** Navegar a la pantalla de historial de transacciones.
  - **Resultado Esperado:** El historial de transacciones se muestra correctamente.
- **Paso:** Verificar que todas las transacciones recientes están listadas con detalles precisos.
  - **Resultado Esperado:** El historial muestra todas las transacciones realizadas con fecha, tipo, cantidad y criptomoneda.

Caso de Prueba: Pantallas de la Aplicación

Título del Caso de Prueba: Verificar la funcionalidad y acceso a las diferentes pantallas de la aplicación.

Requisito funcional: RF1, RF2, RF4, RF13, RF14

Escenario 1: Acceso a la Pantalla Principal

- **Paso:** Iniciar la aplicación y navegar a la pantalla principal.
  - **Resultado Esperado:** La pantalla principal se carga correctamente con accesos a todas las funcionalidades.

Escenario 2: Pantalla de Ajustes de Usuario

- **Paso:** Acceder a la pantalla de perfil de usuario desde la pantalla principal.
  - **Resultado Esperado:** La pantalla de perfil muestra el “publicAddress” del usuario, una opción para eliminar la cuenta y otra para cerrar sesión. También un ajuste para añadir o no automáticamente una criptomoneda en la cartera al recibirla
- **Paso:** Cerrar sesión desde la pantalla de perfil.
  - **Resultado Esperado:** La sesión se cierra y el usuario es redirigido a la pantalla de Login.
- **Paso:** Borrar el usuario
  - **Resultado Esperado:** Se borra el ususario y el usuario es redirigido a la pantalla de Login.
- **Paso:** Activar la opción de los ajustes
  - **Resultado Esperado:** Cuando el usuario reciba una criptomoneda si está activa esta opción se añadirá automáticamente a la cartera

Escenario 3: Pantalla de Inicio de Sesión y Registro

- **Paso:** Navegar a la pantalla de inicio de sesión y probar el inicio de sesión con credenciales válidas.
  - **Resultado Esperado:** El usuario inicia sesión correctamente y es redirigido a la pantalla principal.
- **Paso:** Navegar a la pantalla de registro y crear una nueva cuenta.
  - **Resultado Esperado:** La cuenta se crea correctamente y el usuario puede iniciar sesión con la nueva cuenta.

Caso de Prueba: Seguridad y Autenticación

Título del Caso de Prueba: Verificar la seguridad y autenticación de usuarios.

Requisito funcional: RF11, RF12

Escenario 1: Acceso no Autorizado

- **Paso:** Intentar acceder a una pantalla que requiere autenticación sin iniciar sesión.
  - **Resultado Esperado:** La aplicación redirige al usuario a la pantalla de inicio de sesión y muestra un mensaje de error.

Escenario 2: Intentos de Inicio de Sesión Fallidos

- **Paso:** Introducir credenciales incorrectas en la pantalla de inicio de sesión.
  - **Resultado Esperado:** La aplicación muestra un mensaje de error indicando que las credenciales son incorrectas.

Escenario 3: Expiración de Sesión

- **Paso:** Iniciar sesión y dejar la aplicación inactiva durante un período prolongado.
  - **Resultado Esperado:** La sesión expira y el usuario es redirigido a la pantalla de inicio de sesión cuando intenta acceder a funciones que requieren autenticación.

Caso de Prueba: Interacción con la API REST

Título del Caso de Prueba: Verificar la interacción con la API REST para asegurar la correcta comunicación entre el cliente y el servidor.

Requisito funcional: RF10

Escenario 1: Solicitudes Exitosas a la API

- **Paso:** Realizar una solicitud GET para obtener datos del perfil de usuario.
  - **Resultado Esperado:** La solicitud devuelve datos correctos y un código de estado 200 OK.

Escenario 2: Manejo de Errores en la API

- **Paso:** Realizar una solicitud a un endpoint de la API con parámetros incorrectos.
  - **Resultado Esperado:** La solicitud devuelve un mensaje de error adecuado y un código de estado 400 Bad Request.

5. MANUAL DE PUESTA EN MARCHA.

Este manual proporciona instrucciones detalladas para la instalación, configuración y puesta en marcha de CryptoTrack, tanto para la aplicación web como para la versión móvil.

Aplicación Web

Requisitos previos

- Node.js (versión 16.0.0 o superior)
- npm (incluido con Node.js)
- Cuenta de MongoDB Atlas (para la base de datos)

Pasos de instalación

1. Clonar el repositorio:

git clone https://github.com/Phosky71/WalletSimulator.git
cd cryptotrack

2. Instalar dependencias:

npm install

3. Configurar variables de entorno:
   Crea un archivo .env en la raíz del proyecto con el siguiente contenido:

PORT=3000
MONGODB_URI=tu_uri_de_mongodb_atlas
JWT_SECRET=tu_clave_secreta_para_jwt
COINRANKING_API_KEY=tu_clave_api_de_coinranking

4. Iniciar la aplicación:

npm start

5. Acceder a la aplicación:
   Abre un navegador y visita http://localhost:3000

Configuración de la base de datos

1. Crea una cuenta en MongoDB Atlas
2. Crea un nuevo cluster y obtén la URI de conexión

3. Reemplaza tu_uri_de_mongodb_atlas en el archivo .env con la URI obtenida

Configuración de la API de CoinRanking

1. Regístrate en CoinRanking para obtener una API key
2. Reemplaza tu_clave_api_de_coinranking en el archivo .env con la clave obtenida

Aplicación Móvil

Requisitos previos

- Flutter SDK (versión 2.5.0 o superior)
- Android Studio
- Dispositivo móvil o emulador

Pasos de instalación

1. Clonar el repositorio de la aplicación móvil:

```
git clone https://github.com/tu-usuario/cryptotrack-mobile.git
cd cryptotrack-mobile
```

2. Instalar dependencias:

```
flutter pub get
```

3. Configurar la URL del backend:
   Abre el archivo lib/config/app_config.dart y actualiza la URL del backend:

```
static const String apiUrl = 'https://tu-backend-url.com';
```

4. Ejecutar la aplicación:

```
flutter run
```

Generación de APK para Android

1. Ejecutar el siguiente comando:

```
flutter build apk
```

2. El APK generado se encontrará en build/app/outputs/flutter-apk/app-release.apk

Verificación de la instalación

1. Aplicación Web: Navega a http://localhost:3000 y verifica que puedes registrarte, iniciar sesión y acceder a todas las funcionalidades.

2. Aplicación Móvil: Instala el APK en un dispositivo Android y verifica que puedes iniciar sesión y usar todas las funciones de la aplicación.

Solución de problemas comunes

- Si la aplicación web no se conecta a la base de datos, verifica la URI de MongoDB en el archivo .env.
- Si los precios de las criptomonedas no se actualizan, asegúrate de que la API key de CoinRanking sea válida.
- Para problemas con la aplicación móvil, verifica que la URL del backend sea correcta y accesible desde el dispositivo.

6. CONCLUSIONES Y VÍAS FUTURAS.

6.1 Conclusiones

El desarrollo de Wallet Simulator ha culminado en la creación de una plataforma educativa innovadora que permite a los usuarios explorar el fascinante mundo de las criptomonedas de manera segura y controlada. A lo largo de este proyecto, hemos alcanzado varios hitos significativos:

1. **Diseño intuitivo y accesible:** Hemos logrado crear una interfaz de usuario que equilibra la complejidad inherente del mercado de criptomonedas con una experiencia de usuario fluida y comprensible. Esto facilita el aprendizaje y la participación de usuarios con diversos niveles de experiencia.
2. **Integración de datos en tiempo real:** La implementación exitosa de la API de CoinRanking nos permite ofrecer a los usuarios información actualizada sobre precios y tendencias del mercado, replicando la dinámica volátil del mundo cripto.
3. **Entorno de simulación robusto:** Hemos desarrollado un sistema que permite a los usuarios realizar transacciones simuladas, gestionar carteras virtuales y experimentar con estrategias de inversión sin riesgo financiero real.
4. **Seguridad y privacidad:** La plataforma incorpora medidas de seguridad avanzadas para proteger los datos de los usuarios y garantizar la integridad de las simulaciones.
5. **Plataforma multiplataforma:** Hemos logrado desarrollar una aplicación que funciona en diferentes sistemas operativos, permitiendo a los usuarios acceder desde diversos dispositivos.
6. **Soporte multiusuario:** La plataforma ha sido diseñada para manejar múltiples usuarios simultáneamente, cada uno con su propia cuenta y datos personalizados.

Sin embargo, el desarrollo no estuvo exento de **desafíos:**

- La integración inicial de la API de CoinRanking requirió un esfuerzo considerable para adaptar la estructura de datos a nuestras necesidades específicas.
- Garantizar la escalabilidad del sistema para soportar un gran número de usuarios concurrentes presentó retos técnicos que requirieron optimizaciones en la arquitectura del backend.
- Implementación de una API completamente RESTful: Uno de los desafíos más significativos fue desarrollar una API que adhiriera estrictamente a los principios REST. A pesar de nuestros esfuerzos, no logramos implementar una API completamente RESTful, lo que presentó limitaciones en términos de escalabilidad y consistencia.

6.2 Vías Futuras

Wallet Simulator sienta las bases para un ecosistema educativo en constante evolución. Las siguientes son algunas de las direcciones prometedoras para el futuro desarrollo de la plataforma:

1. **Expansión de funcionalidades:**

   a. Implementar un sistema de trading automático basado en reglas personalizables por el usuario.
   b. Desarrollar un módulo de predicción de precios utilizando técnicas de machine learning.
   c. Crear un mercado virtual de NFTs para simular este aspecto emergente del ecosistema cripto.
2. **Mejoras para una experiencia educativa:**
3.
   a. Introducir un sistema de gamificación con misiones, logros y competiciones entre usuarios.
   b. Desarrollar un "modo historia" que guíe a los usuarios a través de diferentes escenarios del mercado cripto.
   c. Implementar un asistente virtual impulsado por IA para proporcionar consejos personalizados.
4. **Ampliación del alcance:**
   a. Crear versiones localizadas de la plataforma para alcanzar audiencias globales.
   b. Desarrollar una API pública que permita a desarrolladores externos crear aplicaciones y herramientas complementarias.
5. **Optimización técnica:**
   a. Migrar a una arquitectura de microservicios para mejorar la escalabilidad y el mantenimiento.
   b. Explorar la integración de tecnologías blockchain reales para ciertas funcionalidades, manteniendo la naturaleza simulada de las transacciones.
6. **Expansión multiplataforma:**
   a. Desarrollar aplicaciones nativas para iOS y Android para ofrecer una experiencia móvil optimizada.
   b. Crear una versión de escritorio con capacidades avanzadas de análisis y visualización de datos.

El potencial de Wallet Simulator para revolucionar la educación en finanzas digitales es vasto. Con estas mejoras y expansiones, aspiramos a crear un ecosistema educativo completo que no solo simule el mercado de criptomonedas, sino que también prepare a los usuarios para navegar con confianza en el futuro financiero digital. La visión a largo plazo es evolucionar hacia una plataforma basada en Web 3.0, integrando tecnologías blockchain reales y descentralizadas. Esto permitiría a los usuarios interactuar con un ecosistema verdaderamente descentralizado, ofreciendo una experiencia de aprendizaje más auténtica y preparándolos para el futuro de las finanzas digitales y la economía descentralizada.

7. REFERENCIAS Y BIBLIOGRAFÍA.

HTML Standard. (2000). https://html.spec.whatwg.org2

MDN Web Docs. (2025). CSS: Cascading Style Sheets. https://developer.mozilla.org/en-US/docs/Web/CSS3

Bootstrap. (n.d.). CSS · Bootstrap. https://getbootstrap.com/docs/3.3/css/4

MDN Web Docs. (2025). JavaScript Guide. https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide5

W3Schools. (2025). Node.js Tutorial. https://www.w3schools.com/nodejs/6

MDN Web Docs. (2024). Express/Node introduction. https://developer.mozilla.org/en-US/docs/Learn_web_development/Extensions/Server-side/Express_Nodejs/Introduction7

MongoDB. (n.d.). MongoDB Documentation. https://www.mongodb.com/docs/8

Flutter. (n.d.). Flutter - Dart API docs. https://api.flutter.dev9

Public APIs. (2018). CoinRanking API. https://publicapis.io/coin-ranking-api10

ChatGPT. (n.d.).

Anthropic Claude. (n.d.).

Google Bard. (n.d.).

Microsoft Copilot. (n.d.).

GitHub Copilot. (n.d.).