TRABAJO FIN DE GRADO

Grado en Ingeniería Informática

UNIVERSIDAD DE ALMERIA

ESCUELA SUPERIOR DE INGENIERÍA

Implementación de aplicativo para la visualización de API Coinbase usando Flutter

Curso 2020/2021

Alumno/a:

Daniel Barroso Urrutia

Director/es:

Antonio Becerra Terón

Agradecimientos

Quiero agradecer en primer lugar a mis padres que me han apoyado siempre en mis estudios y me han dejado elegir siempre lo que yo he visto conveniente sin ponerse en contra de mi criterio, también quiero agradecer a mis compañeros de carrera que me han hecho el camino mucho más llevadero y que me han apoyado a llevar a cabo mi carrera universitario, haciendo mención a Juan José Escarabajal Hinojo y Manuel Marín Peral. Y el gran apoyo de Cristina, mi psicóloga, que me ha apoyado y me ha ayudado a tener la determinación para terminar la carrera. Además, agradezco a los profesores que sin obligación han decidido invertir su tiempo en hacer que consiga más motivación en los distintos campos de la Ingeniería Informática y en especial a mi coordinador de TFG, Antonio Becerra Terón, que desde el primer momento aceptó ayudarme a elaborar mi Trabajo de Fin de Grado.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción.

Este Trabajo de Fin de Grado (TFG) consiste en el desarrollo de una aplicación móvil dedicada a la visualización de datos referentes al mundo de las *criptomonedas*[1] haciendo uso de *Flutter*[2][3][4] y accediendo a *APIs*[5] para conseguir realizar un problema propuesto.

1.1. Motivación.

En estos últimos tiempos hemos observado que está muy de moda el tema de *criptomonedas*. Sin embargo, las monedas basadas en la criptografía llevan ya años en funcionamiento, en concreto la que surgió de la nada en Octubre de 2008 fue *Bitcoin*[6], la ya aclamada y reconocida moneda creada por un tal Satoshi Nakamoto aunque se rumorea que podría llegar a ser un conjunto de personas las que la bautizaron. *Bitcoin* se basa en las 3 tecnologías clave:

- **Blockchain o Cadena de Bloques**[7] (ver figura 1.1), lo que de forma simplificada sería una "lista" de todas la operaciones que se realizan para la moneda pertinente y esta se guarda en cada uno de los *nodos* que pertenecen a esta red, verificando así cada operación. Cada uno de los *nodos* tendría una copia exacta de esta cadena de bloques, evitando así que cualquier intruso pueda modificarla asegurando así la integridad de la misma.

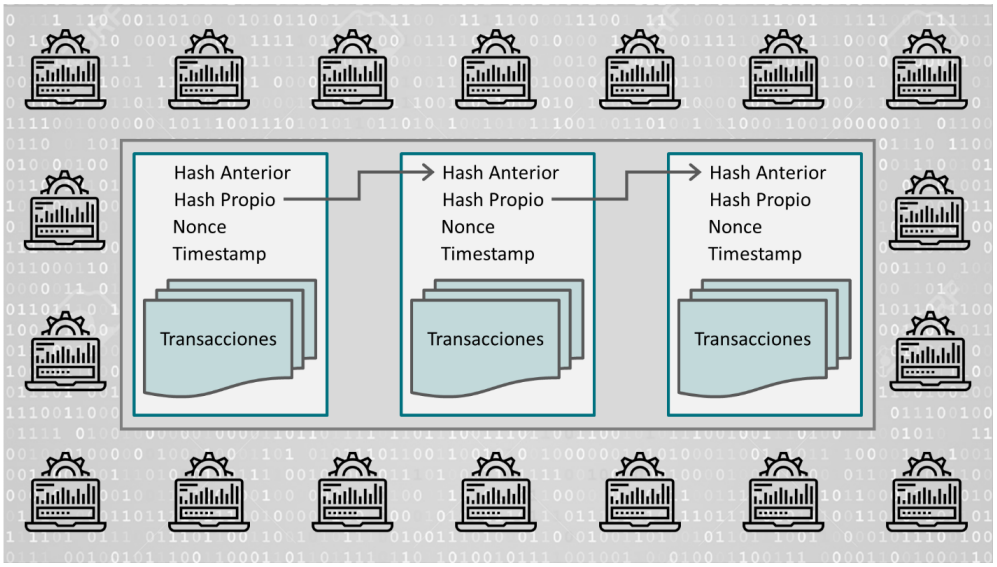

Figura 1.1: Representación gráfica de Blockchain.

- **P2P (Peer to Peer)**[8], esta tecnología ofrece la posibilidad de hacer las transacciones entre la persona que va a recibir el dinero y el que lo envía, de este modo se permite operar sin depender de una autoridad externa o banco.
- **Hashcash o Proof of Work**[9], se utiliza para garantizar la estabilidad de la moneda y se trata de establecer un trabajo criptográfico complejo que debe resolver haciendo uso de

un procesador, estos son los conocidos como *mineros*. Como resultado debe devolver una cadena alfanumérica que servirá de testigo de que el trabajo ha sido realizado de forma exitosa y se le recompensará con una cantidad de *Bitcoins*.

Bitcoin ofrece una gran cantidad de ventajas como son las siguientes:

- **Descentralizado**, como ya hemos comentado debido a la independencia de verificación de transacciones.
- **"Anonimato"**, aunque la dirección de tu billetera aparece en la cadena de bloques y cualquiera puede ver las transacciones que se realizan, la exposición es muchísimo menor que en las entidades habituales que almacenan toda información sobre nosotros.
- **Transparencia**, cualquier usuario puede observar todas las transacciones realizadas en el tiempo.
- **Rapidez**, debido a que no se depende de ninguna entidad o servicio bancario las transacciones se realizan en cuestión de minutos al contrario de las tradicionales que pueden tardar días.
- **Irreversible**, una vez se realiza una transacción se queda almacenada en la cadena y no se puede remover bajo ningún concepto aunque se desee, esto aunque puede ser un arma de doble filo evita que se realicen operaciones corruptas.

Aunque estas características son específicas de *Bitcoin* nos sirven para entender el funcionamiento de la gran mayoría de *criptomonedas* o al menos nos ayuda a comprender de manera general como se comportan este tipo de monedas.

Además, otra motivación para la realización de este proyecto es la de utilizar *Flutter*, ya que este framework está en continuo desarrollo y ofrece una cantidad enorme de opciones teniendo una muy buena proyección y me permite a mi aprender un lenguaje fresco y avanzado con el que desarrollarme como profesional.

1.2. Problema a resolver

Aunque hay grandes *exchanges* que ofrecen vistas muy elaboradas y profesionales de los distintos mercados, estas se ofrecen a nivel web no realmente adaptadas a un nivel portable. Sí es cierto que estas ofrecen sus variantes móviles, pero estas no permiten la visualización de los datos al mismo nivel que lo hacen las implementaciones web. De este modo se plantea realizar este proyecto con el fin de adaptar las conveniencias de sus versiones web a un entorno móvil, teniendo así los dos "mundos" en una sola plataforma.

El problema en cuestión será el de conseguir acoplar de una forma natural los gráficos de velas de la plataforma web y las distintas representaciones de una manera que no resulte abrumadora pero a su vez poder realizar las mismas funciones que en escritorio.

Debido a esta situación se plantea también la opción de implementar una aplicación que pueda visualizar y hacer operaciones para cada mercado haciendo uso de *exchanges* que son las que ofrecen a los usuarios los distintos *productos o mercados* en los que poder operar.

Para este cometido vamos a usar la *API* de *Coinbase Pro*[10], se trata de un *Exchange* de *criptomonedas* que permite a los clientes la posibilidad de tener una cartera de las distintas

monedas que con estas se pueden establecer ordenes de compra y venta para los *productos/-mercados* de *Coinbase Pro*, esta plataforma ofrece una gran cantidad de productos con lo cual permite la posibilidad de analizar un gran espectro del mercado y entonces poder ver donde invertir nuestro capital.

1.3. Objetivos generales y específicos.

1.3.1. Objetivos generales.

El objetivo principal de este proyecto es implementar una *app móvil* que nos permita visualizar el progreso en cualquier plataforma *exchange*, y poder realizar operaciones en tiempo real teniendo una perspectiva de cada uno de los mercados de una forma más amplia.

Para hacer fácil la visualización del progreso en la plataforma lo ideal es crear una *aplicación móvil* debido a que de forma clara y concisa podemos observar las operaciones que se van realizando, además al ser móvil proporciona una serie de ventajas claves:

- Rapidez
- Versatilidad
- Accesibilidad

Otro de los objetivos, que serían a nivel personal y didáctico, sería el de aprender y afianzar el conocimiento aprendido en la carrera, realizando peticiones *HTTP Request*[11] a una *API* privada en la cual se necesita además autenticación y también ser capaz de adaptar la información de esta estandarizando y abstrayendo los datos para implementarlo en el aplicativo con una serie de analíticas y gráficos. Además de aprender a usar como framework para nuestro frontend *Flutter*, como he comentado en las motivaciones, a nivel profesional y laboral para mi persona tiene una muy buena proyección debido a que es un lenguaje reciente que está siendo muy utilizado ya que está respaldado de una gran empresa como es *Google*.

1.3.2. Objetivos específicos.

A un nivel más concreto se plantean una serie de objetivos para cumplir los generales, que serían los siguientes:

- Comparativas de distintos productos/mercados pudiendo seleccionar entre distintas variables y gráficos de muestreo.
- Visualización de los productos/mercados mediante gráficos de velas en tiempo real, además de poder ver el libro de órdenes que se estén realizando en ese instante con representaciones de barras para conocer el estado del mercado (a través de *WebSockets*[12]).
- Operaciones de compra y venta sobre los productos seleccionando la cantidad y al precio pertinente.
- Tener un resumen del balance de nuestra cartera.

Con esta serie de funcionalidades se considera que la aplicación cumple los requisitos para poder ser un alternativa para facilitar el uso del *exchange*.

1.4. Planificación del proyecto.

1.4.1. Estimación del proyecto

Primeramente, se realiza una estimación de tanto la duración del proyecto cómo los costes del mismo, para ello primero se realizó un *Análisis de Puntos de Función* considerando el proyecto de complejidad media distribuidos en los siguientes puntos:

- Entradas (3)
- Salidas (2)
- Consultas (7)
- Ficheros de interfaz (1)

En total tendríamos **53 puntos de función sin ajustar (PFSA)**, aplicando el factor de complejidad ($\sum Fi$) siguiendo la fórmula tenemos:

$$PF = PFSA * [0,65 + 0,01 * \sum Fi]$$

$$PF = 53 * [0,65 + 0,01 * 38]$$

$$PF = 13,09$$

También se realizó una estimación usando la herramienta de estimación *COCOMO II*, en concreto para este proyecto se usó el *Modelo de COCOMO básico* y usando el *modo orgánico*, se dió por supuesto que el desarrollo serían alrededor de 3 KLDC (Miles de líneas de código), los cálculos serían los siguientes (siendo a,b,c y d valores establecidos por *COCOMO*):

Esfuerzo $= a * KLDC^b$	Duración $= c * E^d$	Nº de personas $= E/D$
$E = 2,4 * 3^{1,05}$	$D = 2,5 * 7,6^{0,38}$	$N = 7,6/5,4$
$E = 7,6$ personas por mes	$D = 5,4$ meses	$N = 1,4$ personas

Teniendo estos datos podemos obtener una estimación de los costes económicos, suponiendo que el costo por persona/mes es de 1000€, teniendo un esfuerzo de 7,6 personas/mes en total el costo económico de 7600€. Además de tener una duración de 5,4 meses podemos realizar un *Diagrama de Gantt* donde se representa cómo se irán desarrollando las fases (ver figura 1.2).

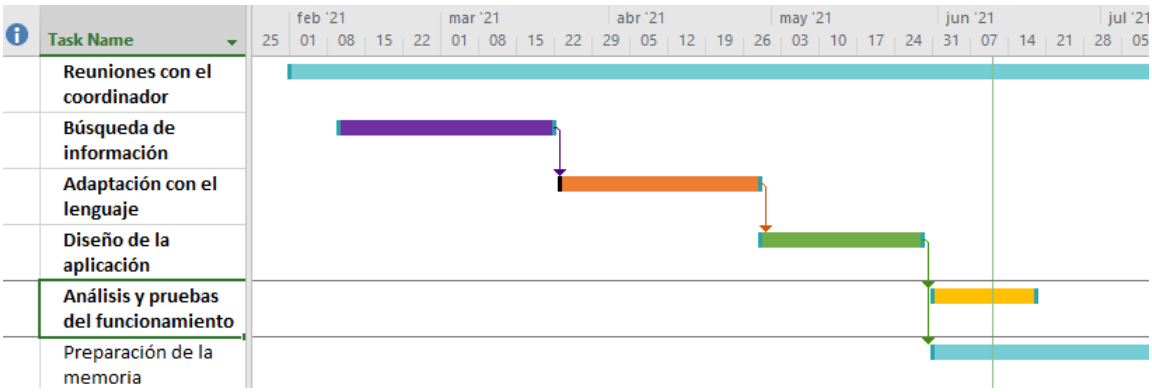

Figura 1.2: Diagrama de Gantt del TFG.

1.4.2. Realización real del proyecto

Este proyecto se ha realizado con la planificación establecida previamente, teniendo en cuenta ciertos márgenes de variación. La duración total del desarrollo han sido de unas 210 horas (sin contar la memoria), empezando el mes de febrero y finalizando alrededor del día 5 de julio. El proyecto se desarrolló a lo largo de aproximadamente 5 meses lo cual se asemeja bastante a la estimación realizada previamente, divididas en distintas fases:

- Reuniones con el coordinador (10 horas)
- Búsqueda de información (30 horas)
- Adaptación con el lenguaje (10 horas)
- Diseño de la aplicación (120 horas)
- Análisis y pruebas del correcto funcionamiento (40 horas)
- Preparación de la memoria (80 horas)

Finalmente, sumando la totalidad de horas incluyendo la preparación de la memoria, hacen un total de 290 horas teniendo en cuenta que en un principio el proyecto estaba estimado realizarse en unas 300, tenemos que en gran manera se ha cumplido con lo establecido teniendo unas 10 horas menos de los estimado.

Estableciendo unos rangos para la realización de las mismas los cuales se tuvieron en cuenta para organizar correctamente cómo se irían desarrollando cada uno de las fases. En las primeras fases se hizo una labor de entender de forma correcta el funcionamiento tanto del lenguaje de programación utilizado tanto analizar las opciones que nos dan los *Endpoints* [5] de *Coinbase Pro* y que se podría llegar a hacer con esos datos, además de ver cómo adaptar el diseño de nuestra aplicación a distintos gráficos. Las siguientes fases se centraron en implementar el aplicativo según lo analizado en las fases anteriores, realizando a su vez distintos análisis y tests del mismo.

Inicialmente en la estimación se realizaron los cálculos haciendo la suposición de unas 3000 líneas de código, finalmente fueron en total unas 4800 líneas de código, lo cual realizando de nuevo las estimaciones con estos datos, los cálculos serían de la siguiente manera:

Esfuerzo $= a * KLDC^b$	Duración $= c * E^d$	Nº de personas $= E/D$
$E = 2,4 * 4,8^{1,05}$	$D = 2,5 * 12,4^{0,38}$	$N = 12,4/6,5$
$E = 12,4$ personas por mes	$D = 6,5$ meses	$N = 1,9$ personas

Lo cual supondría un coste de 12400€ con la suposición de 1000€ por persona/mes.

Capítulo 2

Estado del arte.

2.1. Plataformas de intercambio de monedas

Una vez se planteó el proyecto, durante la búsqueda de información se observaron los distintos *exchanges* que ya existen en el mercado como podrían ser las aplicaciones móviles de *Binance*[13], *Coinbase*[14] o *Coinbase Pro*, las dos últimas diseñadas por la propia empresa que nos ofrece la *API* con los *endpoints* a partir de los cuales hacemos todas las operaciones. Los problemas que se plantean son que estas ofrecen gráficos y opciones limitadas, aunque la que más nos funciones nos permite sería la de *Binance* teniendo representaciones con *gráficos de velas japonesas* que aporta la manera de visualizar los movimientos de mercado de una forma más clara que un simple gráfico lineal. La tesitura sería que la *API* de *Coinbase Pro* ofrece el acceso a la información mucho más sencilla que la de *Binance*.

También existen aplicaciones *exchanges* que utilizan mercados tradicionales distintos a las *criptomonedas* como puede ser *XTB*[15] o algunos que incluyen ambos tipos de mercado pero de forma más limitada como son *eToro*[16] o *Vivid*[17].

2.1.1. Binance.

Binance se trata de una plataforma de intercambio/exchange de *criptomonedas*, se trata de una de las plataformas con más volumen de usuarios además de proporcionar para operar más de 100 activos digitales.

Figura 2.1: Logotipo de Binance.

Su plataforma web ofrece gran cantidad de opciones y personalización, además de tener distintos tipos de mercado como pueden ser los mercados a *futuro*. Se visualiza los mercado mediante un gráfico de velas en el cual elegir entre distintas *granularidades* ya sean 15m, 1h, 4h, etc... También incorpora una integración con la plataforma de visualización de mercados más conocida llamada *TradingView*[18], en la cual se permite añadir anotaciones, dibujos, arcos y todo tipo de ayudas para mejorar la comprensión.

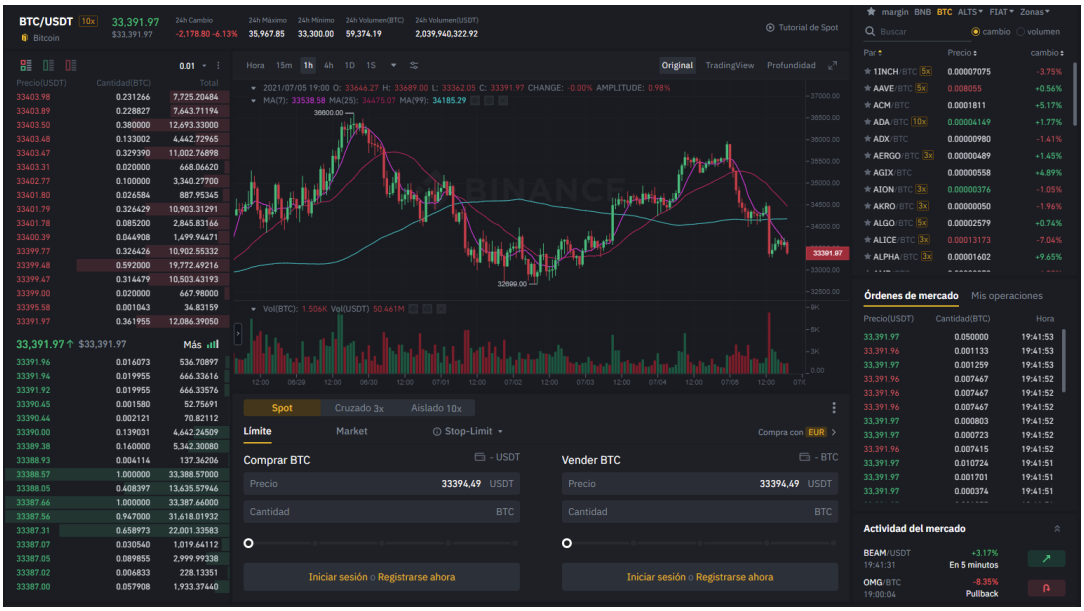

Figura 2.2: Visualización de plataforma web de Binance.

La plataforma móvil es más limitada, tiene una versión con gráficos lineales y no se actualiza de forma instantánea como lo hace la otra, también es cierto que estaría pensada para personas de iniciación, para las más expertas ofrece una versión con algunas opciones más.

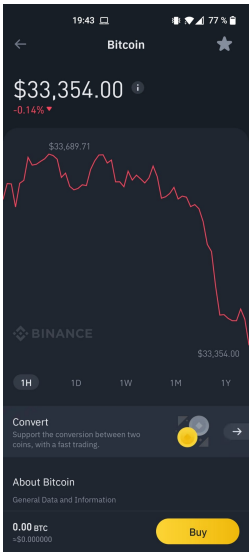

Figura 2.3: Visualización de plataforma móvil de Binance.

2.1.2. Coinbase.

Coinbase se trata de una plataforma de intercambio/exchange de *criptomonedas*, esta plataforma ofrece soporte en una gran cantidad de países, en concreto 32 países.

Figura 2.4: Logotipo de Coinbase.

Esta permite a los usuarios operar de una forma muchísimo más simple y amigable para inexpertos en la materia, además con frecuencia realiza promociones regalando distintas monedas para hacer accesible a cualquiera que lo desee. Ofrece muy pocas opciones de intercambio entre monedas y con límites y comisiones que no hacen viable hacer una gran cantidad de operaciones.

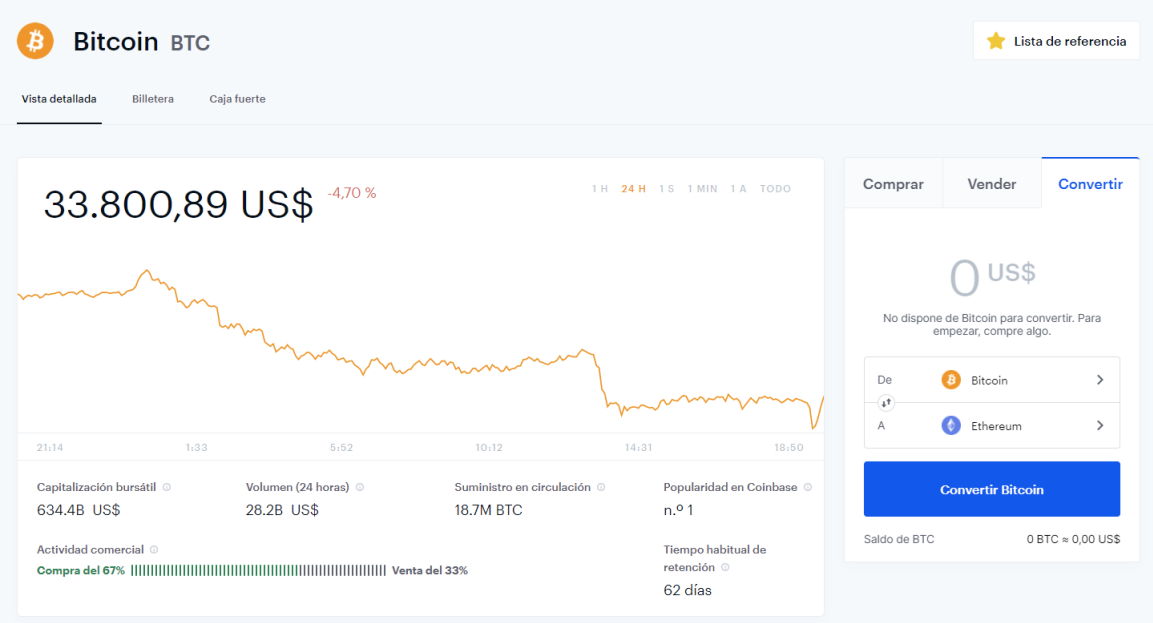

Figura 2.5: Visualización de plataforma web de Coinbase.

2.1.3. Coinbase Pro.

Coinbase Pro se trata de una plataforma basada en *Coinbase* pero esta ya permite trabajar con los mercados de una forma más avanzada que la versión básica, además de ofrecer intercambios entre ambos.

Figura 2.6: Logotipo de Coinbase Pro.

Aparte de tener las ventajas de *Coinbase* se añaden las de tener una estructura de comisiones mucho más liviana y también tener una *API* más avanzada pudiendo realizar automatizaciones mediante BOTs.

Figura 2.7: Visualización de plataforma web de Coinbase Pro.

2.1.4. XTB.

XTB se trata de uno de los mayores Brokers cotizados del mundo, proporcionando a inversores minoristas acceso instantáneo a cientos de mercados globales de *CFDs, Fx, Acciones y ETFs*. Se centran en mercados tradicionales teniendo oficinas en más de 13 países, incluyendo España, Reino Unido, Polonia, Alemania, Francia o Chile.

Figura 2.8: Logotipo de XTB.

XTB tiene una plataforma web, una móvil y una plataforma de escritorio y además para los iniciados ofrece una cuenta demo en la cual probar y aprender a manejarnos en los distintos mercados

Figura 2.9: Visualización de XTB.

2.1.5. Vivid.

Esta plataforma se centra en los dos mundos tanto en mercados tradicionales como en *criptomonedas*, pudiendo tener todo incluido en un sistema haciendo la experiencia más cómoda. Sin embargo, no ofrece una gran cantidad de monedas digitales.

Figura 2.10: Logotipo de Vivid.

Permite inversiones ilimitadas sin comisiones y ofrece opciones de cashback en las que usando la tarjeta que proporciona obtienes un porcentaje del pago realizado, además incluye un sistema de transferencias de dinero a tus contactos o por IBAN o configurar pagos periódicos.

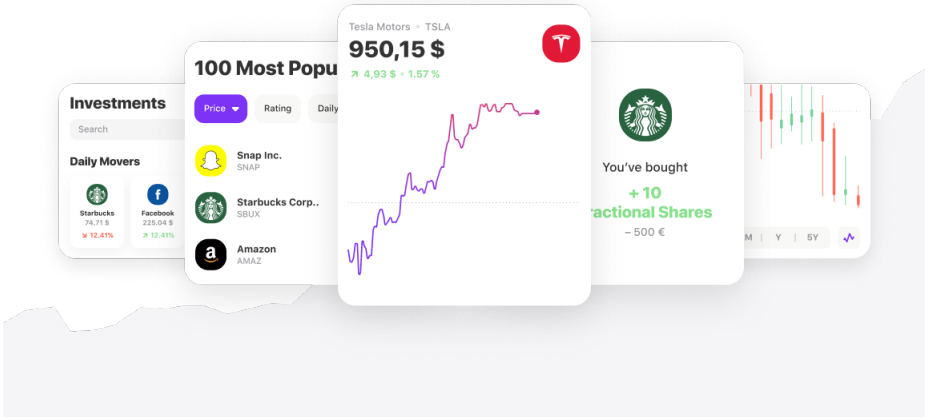

Figura 2.11: Visualización de Vivid.

2.1.6. Comparativa

A continuación, se muestra una tabla en la que se pueden observar las virtudes de cada una de las plataformas

Exchange / Funcionalidad	Binance	Coinbase	Coinbase Pro	XTB	Vivid
Gran oferta de criptomonedas	✓	✓	✓	✗	✗
API Abierta	✓	✓	✓	✓	✗
Datos avanzados	✓	✗	✓	✗	✗
Comisiones bajas	✓	✗	✓	✓	✓
Utilización sencilla de los datos	✗	✓	✓	✓	✓

Tabla 2.1: Comparativa de las plataformas de intercambio.

2.2. Frameworks multiplataforma

Muchas empresas están decidiendo usar *frameworks multiplataforma* debido a que tiene la enorme ventaja de que reduce el tiempo de desarrollo enormemente.

2.2.1. Flutter

Flutter se trata de un *SDK* de código fuente abierto para desarrollo de aplicaciones móviles multiplataforma creado por *Google*, presentado en el *Dart Developer Summit* de 2015, pensado para desarrollar interfaces de usuario para aplicaciones Android, iOS y Web. *Flutter* se basa en una serie de componentes:

- **Dart Platform**, es el lenguaje que se utiliza para escribir las apps en *Flutter*, se trata de un lenguaje de programación de código abierto desarrollado también por *Google*, revelado en 2011. *Dart* está pensado para ofrecer mejores resultados a algunos problemas de *JavaScript*, pretendiendo ser una plataforma sencilla para proyectos más grandes y ofrecer una mejor seguridad. Viene con un conjunto bastante extenso de herramientas integrado, como su propio gestor de paquetes, varios compiladores, un analizador y formateador, además de una maquina virtual *Dart* y la capacidad de compilar en tiempo de ejecución *Just-in-Time* para hacer cambios rápidamente. La aplicación se compila en el lenguaje nativo del sistema operativo utilizado, proporcinando así una mejora de rendimiento, y en el caso de ser Web se compila en *JavaScript*.

- **Flutter Engine**, está escrito en C++ y proporciona una capa de renderización de gráficos a nivel bajo haciendo uso de *Google Skia*[19], además usa los *SDKs* específicos de cada plataforma iOS o Android, aporta a *Flutter* las librerías principales, como pueden ser de animaciones, gráficos, archivos, accesibilidad y la ejecución y compilación de *Dart*.

- **Foundation Library**, proporciona las clases y funciones básicas utilizadas para construir aplicaciones *Flutter*, como las *APIs* para comunicar con el motor.

- **Widgets**, está son la base de desarrollo de las interfaces en *Flutter*, se construyen basado en el framework *React*[20] siendo la idea de construir las interfaces a través de estos widgets, cada widget define cómo debería mostrarse la vista dado una configuración y estado, cada vez que el estado de un *widget* cambia se reconstruye este, en el que el framework observa las diferencias del estado anterior para minimizar los cambios a realizar y así pasar de un estado a otro.

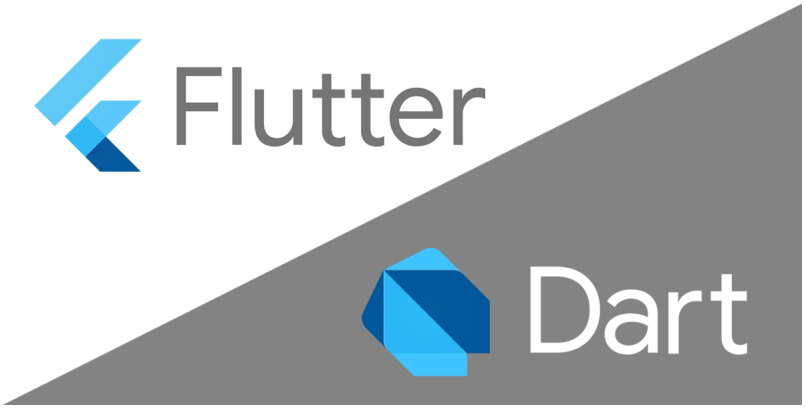

Figura 2.12: Logotipo de Flutter y Dart.

2.2.2. Ionic

Ionic[21] es uno de los frameworks de desarrollo de aplicaciones más populares del mercado, se trata de una red frontend que ayuda a construir aplicaciones móviles nativas con *HTML*[22], *CSS3*[23] y *Javascript*[24]. La ventaja significativa de Ionic es que se pueden utilizar cientos de elementos de interfaz de usuario predeterminados como formularios, filtros, hojas de acción, vistas de lista, barras de pestañas y menú de navegación en su diseño. Además, permite la adaptación con *Angular*[25] y la implementación con librerías como *React y Vue*[26]. La desventaja es que no es un framework gratuito, requiere de pago para obtener la versión completa.

Figura 2.13: Logotipo de Ionic.

2.2.3. React Native

React Native se trata de un framework *JavaScript* para la creación de aplicaciones reales nativas para iOS y Android, basado en la librería de *JavaScript React* para la creación de componentes visuales, usa la construcción de bloques de UI (componentes visuales con los que interacciona el usuario) que las aplicaciones nativas reales de Android e iOS, pero gestiona la interacción entre los mismos utilizando las capacidades de *JavaScript y React*.

- **Compatibilidad Cross-Platform**
- **Funcionalidad nativa**, la unión de *React Native* junto con *JavaScript* permite la ejecución de aplicaciones más complejas de manera suave, mejorando incluso el rendimiento de las apps nativas y sin el uso de un *WebView*[27].
- **Actualizaciones instantáneas (para desarrollo y/o test)**, permite realizar cambios en el dispositivo del usuario directamente sin depender de la tienda del sistema.
- **Sencilla curva de aprendizaje**

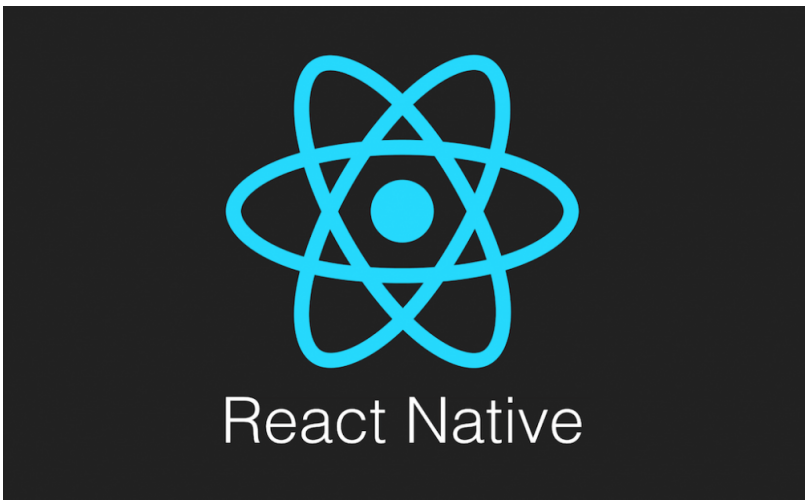

Figura 2.14: Logotipo de React Native.

2.2.4. Xamarin

Xamarin[28] es un software para el desarrollo de aplicaciones móviles, puesto que permite a los desarrolladores compartir hasta un 90 % del código entre plataformas distintas, utilizando un único lenguaje de programación, C#, que cada vez está alcanzando más usuarios debido al rendimiento como por la posibilidad del acceso al API nativo usando el método *AOT Ahead-of-time*[29].

- **Amplia variedad de código de terceros**, esta herramienta ofrece interoperabilidad con lenguajes Objective-C, C y C++.
- **Importantes mejoras con el lenguaje C#**, ofrece construcciones funcionales como lambdas o características de programación en paralelo.
- **Compatibilidad con XML**
- **Sistema de administración de proyectos**, al usar *Visual Studio*, un entorno de desarrollo integrado (IDE) moderno, *Xamarin* aporta nuevas características como un potente sistema de administración de proyectos y soluciones, así como la finalización automática de código o un mayor control del mismo.

Figura 2.15: Logotipo de Xamarin.

2.2.5. Comparativa

A continuación, se muestra una tabla en la que se pueden observar las virtudes de cada una de los frameworks multiplataforma.

Framework / Funcionalidad	Flutter	Ionic	React Native	Xamarin
Compilación nativa	✓	✓	✓	✓
Lenguaje similar a Java	✓	✗	✗	✓
Uso gratuito	✓	✗	✓	✓
Integración sencilla con entorno Google	✓	✗	✗	✗

Tabla 2.2: Comparativa de los frameworks multiplataforma.

Una vez repasadas las ventajas y desventajas de cada uno se decidió usar *Flutter* porque era el que más encajaba ya que ofrece un rendimiento muy alto debido a la compilación nativa usando el *SDK* del sistema operativo, el lenguaje con el que más desarrollo he realizado es Java con lo cual es una ventaja que sea similar a este y otra ventaja que puede ser a futuro es que

al ser de un framework de *Google* permite el uso de sus tecnologías más facil, como puede ser *Firebase*, en la cual se podría en un futuro realizar en la nube las operaciones que irían realizando los BOTS de compra/venta, tanto adaptar la autenticación en esta plataforma incluso validando con una cuenta *Gmail* o tener una base de datos en esta.

2.3. Aplicaciones basadas en *Flutter*

Muchas eligen *Flutter* debido a que a parte de estar respaldada por la gigante tecnológica *Google* tiene una enorme comunidad y se está poniendo muchos recursos en lograr que sea un *framework* exitoso. Además, los desarrolladores tienen gran cantidad de tutoriales y ejemplos que hacen el aprendizaje mucho más sencillo y motivante.

2.3.1. Google Ads

Google Ads es una herramienta esencial para cualquiera que necesite organizar sus campañas publicitarias, te ayuda a crear campañas publicitarias para tus empresas para aumentar la exposición en los mercados con mínimo esfuerzo. Se pueden observar las estadísticas de estas, como pueden ser visualizaciones, clicks, alertas en tiempo real y tener soporte por *Google*.

Figura 2.16: Logotipo de Google Ads.

2.3.2. Xianyu by Alibaba

Alibaba, la gigante multinacional China dedicada al *eCommerce* la cual es la plataforma más grande del mundo en ventas online. *Xianyu* esta siendo usada por mas de 50 millones de usuarios para comprar y vender millones de productos en una enorme cantidad de categorias, ayudando a establecer conexiones entre consumidores haciendo el proceso de compra/venta más natural.

Figura 2.17: Logotipo de Xianyu.

2.3.3. Hue Sync & Hue Bluetooth by Phillips Hue

Phillips Hue se trata de un sistema de iluminación para hogares inteligentes, se encargan de vender lámparas, bombillas y accesorios para consumidores que desean luces inteligentes en sus casas. Dos de sus apps estan desarrolladas en *Flutter*, *Hue Sync y Hue Bluetooth* en las cuales sus clientes puedes modificar el tono de las luces y sincronizar todos sus productos. *Phillips* ha usado *Flutter* desde 2018.

Figura 2.18: Logotipo de Phillips Hue.

2.3.4. Reflecty

Reflecty es una aplicación de diario que te ayuda a ir guardando tus pensamientos en un lugar de forma ordenada y poder ver cómo te has sentido durante la semana o que has ido mejorando. Originalmente se desarrolló usando *React Native* pero se trasladaron a *Flutter* cuando supieron que podían desarrollar aplicaciones para *Android e iOS* reduciendo así el tiempo de desarrollo hasta casi la mitad.

Figura 2.19: Vista de app Reflecty.

Capítulo 3

Desarrollo del TFG

En este Trabajo de Fin de Grado se va a desarrollar como ya comentado, una *app móvil* basada en *Flutter* haciendo uso de *API Coinbase*, estos datos en formato JSON se procesan y se adaptan a los distintos objetos en la aplicación según se van necesitando, para la actualización de la información se usan los llamados *Streams*, los cuales son unos "canales" por los que se transmiten los datos y estos se actualizaran repetidamente con un tiempo previamente establecido. En el caso de necesitar una actualización en tiempo real se hace uso de *WebSockets* permitiendo así actualizar los datos al instante en que se modifican en el *exchange*, simplemente estableces una "subscripción" a uno de los canales que deseeas escuchar e irán llegando *snapshots* con la información.

3.1. Metodología.

Para la gestión y desarrollo correcto del proyecto se han utilizado las metodologías *ágiles*, concretamente la metodología *Scrum*[30]. Esta metodología se trata de un framework o marco de trabajo que tiene como fin desarrollar en períodos cortos de tiempo basados en: la transparencia, la inspección y la adaptación. Esta metodología se basa en los siguientes puntos:

- La flexibilidad para las modificaciones debido a los cortos desarrollos.
- El factor humano.
- La colaboración e interacción con el cliente.
- La iteración asegura buenos resultados.

En concreto, para nuestro proyecto se plantea de la siguiente forma, primeramente se estudia la *API* de *Coinbase Pro* y se piensa que funcionalidades de podrían agregar en función de las opciones de la misma, luego se inicia el desarrollo de estas seguido de una fase de testeo, después hay una fase de "despliegue" y finalmente una revisión final del software completo.

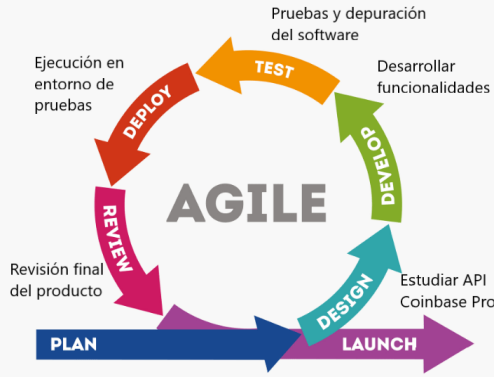

Figura 3.1: Representación de metodología agile de nuestro proyecto.

A la hora de desarrollar el proyecto se ha realizado de forma ordenada usando métodos como el Modelo-Vista-Controlador[31] (ver figura 3.2) el cual nos permite elaborar un software tanto adaptable como flexible y en cualquier momento escalable, esto pondrá en manifiesto la forma de desarrollar software de un modo profesional que permita además en el supuesto trabajo en equipo que alguien ajeno al proyecto pueda modificar y entender claramente el funcionamiento del mismo.

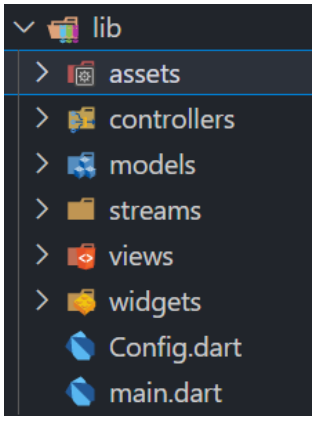

Figura 3.2: Visualización de la gestión de directorios.

3.2. Hardware.

Para la realización del trabajo se han utilizado pocos elementos de hardware ya que se trata de un proyecto software y con equipo realmente básico se podría llegar a realizar.

- Equipo de sobremesa:
  - Descripción: Equipo desde el cual se realizará el desarrollo del proyecto.
  - Especificaciones: Sistema Operativo *"Windows 10"*, 16GB de RAM, procesador AMD Ryzen 5 3600 3.6Ghz y para el almacenamiento 256GB SSD y 1TB HDD.
- OnePlus Nord (ver figura 3.3):
  - Descripción: Smartphone en el cual se ejecuta la aplicación cuando se necesita testear fuera del emulador.
  - Características: Sistema Operativo *"OxygenOS Android 11"*, 8GB de RAM, procesador Qualcomm® Snapdragon™ 765G 5G y 128GB de almacenamiento.

Figura 3.3: OnePlus Nord.

- Conjunto de 2 monitores:
  - Descripción: Se han utilizado 2 monitores para el desarrollo más adecuado y veloz.
    - Monitor 1 (AOC 24G2U) (ver figura 3.4):
      - Especificaciones: Tamaño 23,8", Resolución 1920x1080, frecuencia de actualización 144Hz, tiempo de respuesta 1ms, panel IPS.
    - Monitor 2 (Acer M222HQML) (ver figura 3.5):
      - Especificaciones: Tamaño 21,5", Resolución 1920x1080, frecuencia de actualización 60Hz, tiempo de respuesta 5ms, panel IPS.

Figura 3.4: AOC 24G2U.

Figura 3.5: Acer M222HQML.

3.3. Software.

En objetivo de realizar el proyecto se ha instalado en el equipo con sistema operativo *Windows 10* editores de texto tales como *Visual Studio Code* el cual se trata de un editor de texto muy liviano y simple pero que tiene la capacidad de añadir extensiones las cuales permiten la personalización casi infinita del mismo. Por ejemplo, para poder trabajar con *Flutter* se ha instalado las propias extensiones oficiales de *Dart Code* como son las de *Dart*[32] y *Flutter*, estas ofrecen soporte al lenguaje y facilitan el desarrollo agregando *IntelliSense* y autocompletado además de ayudar con la ejecución y depuración del código.

Además, se tuvo en cuenta desde el principio el control del desarrollo y para ello se utilizaron Sistemas de Control de Versiones como sería *Git*, para ello se debe instalar *Git* en nuestro sistema, una vez añadido se creó un repositorio en *GitHub*[33] y se vinculó con *Visual Studio Code*[34] el cual tiene incorporado de serie un apartado de control de versiones donde se trabaja de un modo muy simple, rápido e intuitivo el poder controlar el desarrollo de cada una de las funciones del aplicativo. Incluso se tuvo en cuenta durante toda realización del proyecto, realizar el software y las funcionalidades divididas en distintas *ramas* con finalidad de poder retroceder cuando no se pueda avanzar en el desarrollo y así evitar acumular errores donde nuestro software ya está funcionando correctamente y que tenga más firmeza.

Para la ejecución de la aplicación se hizo uso de un emulador de Android, para incorporarlo debemos instalar *Android Studio* el cual se trata de un IDE para el desarrollo de aplicaciones android y proporciona una función de emulación llamada *AVD* (ver figura 3.6 y 3.7) donde nos permite crear distintos emuladores para distintos dispositivos móviles ya sean smartphones, smartwatches, tablets, TVs o incluso para *Android Auto*.

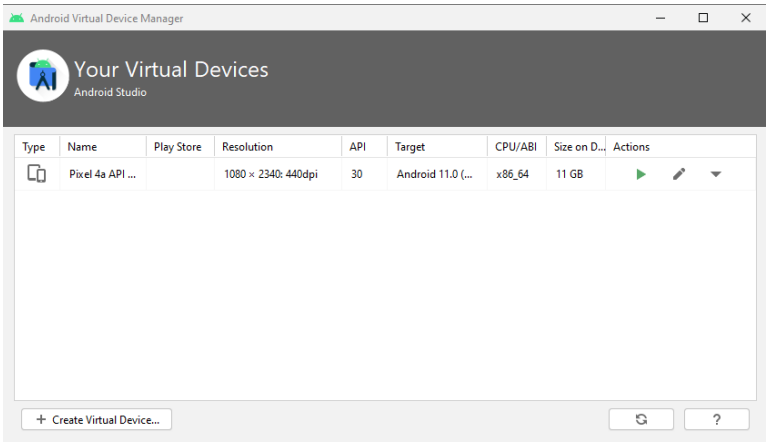

Figura 3.6: AVD.

Figura 3.7: Vista de emulador.

También para las pruebas de llamadas *Request* se ha utilizado una extensión para *Visual Studio Code* llamada *Thunder Client* (ver figura 3.8) agilizando así el proceso viendo de manera inmediata la *Response* de la petición

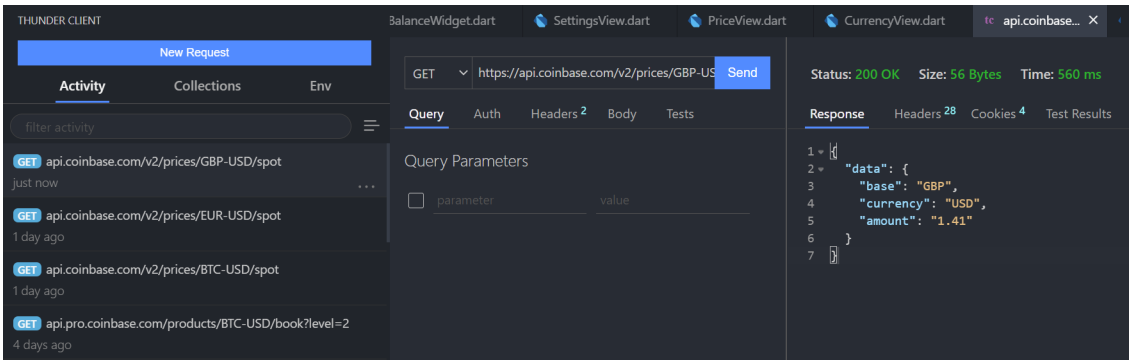

Figura 3.8: Thunder Client.

Para la realización de la aplicación se han usado una serie de librerías (ver figura 3.9), en las que caben destacar *fl_chart[35], k_chart[36] y charts_flutter[37]* que se tratan de librerías que ofrecen distintos gráficos.

```
dependencies:
  flutter:
    sdk: flutter
  charts_flutter: ^0.9.0
  flutter_dotenv: ^3.1.0
  rxdart: ^0.26.0
  smooth_page_indicator: ^0.1.5
  fl_chart: ^0.8.7
  http: ^0.12.1
  crypto: ^3.0.0
  stomp:
  web_socket_channel: ^2.0.0
  k_chart: ^0.1.3
  multi_select_flutter: ^4.0.0
  palette_generator: ^0.3.0
```

Figura 3.9: Librerías utilizadas en el proyecto.

- **fl_chart**, librería que permite añadir a nuestro proyecto gráficos con un estilo minimalista, simple y moderno.

Figura 3.10: Gráficos de fl_charts.

- **k_chart**, librería específica para los gráficos de velas la cual ofrece una representación detallada de las velas tanto mínimos, máximos, aperturas, cierres, vólumenes, además de representación lineal y medias aritméticas.

Figura 3.11: Gráficos de k_chart.

- **charts_flutter**, librería que nos ofrece la propia *Flutter* que tiene una gran cantidad de tipos de gráficos y configuraciones aunque es cierto que no son diseños tan llamativos, se ha utilizado para representaciones simples.

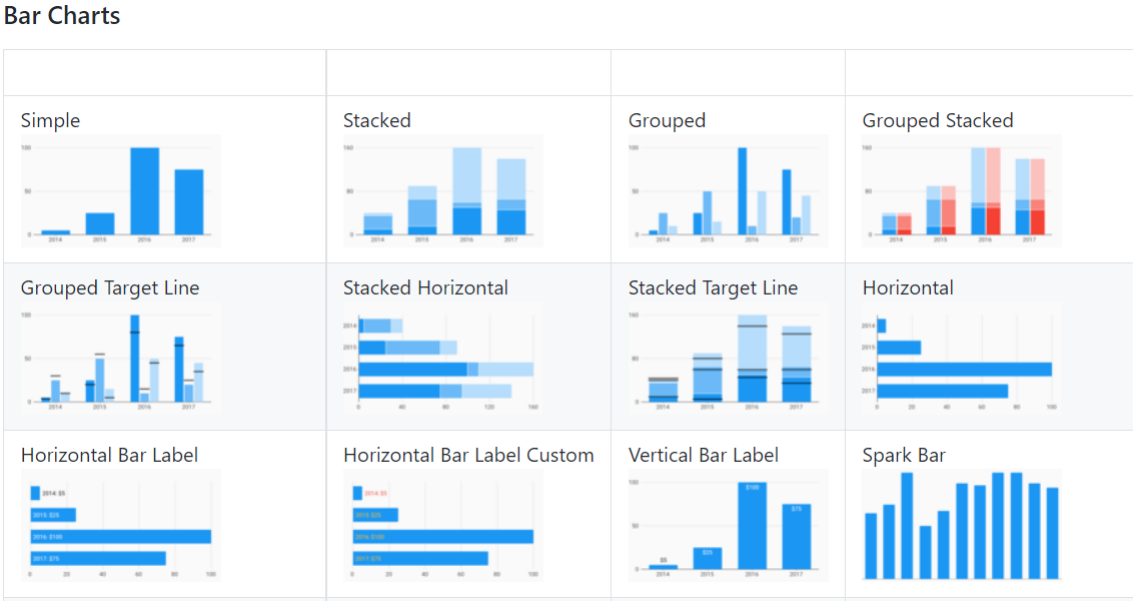

Figura 3.12: Gráficos de charts_flutter.

Además, se han utilizado ciertas librerías para la comunicación con las distintas *APIs y WebSockets* como son *http o web_socket_channel*, las demás librerías serían de interés secundario ya sean para la selección de paletas de colores, añadir variables de entorno o para crear hashes.

3.4. Descripción y modelado

El desarrollo de la aplicación se distribuye como se ha comentado, siguiendo la metodología *Modelo-Vista-Controlador*, tendremos dos controladores base los cuales se utilizan a lo largo de toda la aplicación para realizar las distintas funciones:

- **CoinbaseController**, con este controlador se realizan todas las llamadas a la *API* teniendo distintas funciones en función de lo que se necesita, se realiza también el procesado de estas para poder ser llamadas desde cualquier lugar de la aplicación.
- **RequestController**, este controlador es donde se hacen las llamadas como tal, introduciendo las opciones de la llamada (GET, POST, body, etc...), generalmente se usa desde *Coinbase Controller* .

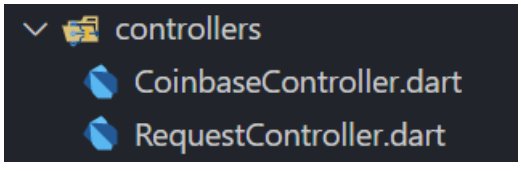

Figura 3.13: Diagrama de casos de uso de la aplicación.

También tendremos una serie de modelos con los cuales organizar la información:

- **Balance**, en este modelo se guarda un listado de *Wallet* teniendo todas las carteras y además el balance total.
- **Order**, serían cada una de las ordenes en la plataforma.
- **Pair**, se utiliza como modelo "secundario" para tener pares de objetos para cualquier utilidad.
- **Product**, contiene la id del producto y un listado con las *KLineEntity o velas* para cada uno.
- **Wallet**, serían cada una de las carteras almacenando la moneda, la cantidad, el precio y el color predominante del icono.

Para cada una de las vistas tendremos los siguientes:

- **CompareView**
- **CurrencyView**
- **DonutView**
- **HomeView**
- **MainLayout**
- **MainPage**
- **OperationsView**
- **PricesView**
- **PriceView**
- **ProductView**
- **SettingsView**
- **WelcomeView**

Para widgets que se usan repetidamente se tiene un directorio aparte:

- **BalanceWidget**
- **BarChartWidget**
- **LineChartWidget**
- **OperationsWidget**
- **PagesWidget**
- **ProductsWidget**
- **SimpleTimeSeriesChart**
- **TitleWidget**
- **VerticalBarLabelChart**
- **WalletWidget**

Para la obtención de datos se utilizan como ya comentado los *Streams* para cada "dato" que se quiere conseguir se tiene un "controlador" con un *stream* y una función *fetchData* que se llamará cada vez que se quiera actualizar este *stream*(ver figura 3.15), estos stream se llamarán usando los *StreamBuilder* (ver figura 3.14)

```
@override
Widget build(BuildContext context) {
  return StreamBuilder<Balance>(
      stream: balanceStream.stream,
      builder: (context, AsyncSnapshot<Balance> snapshot) {
        if (snapshot.hasData) {
          List<PieChartSectionData> sections = [];
          Map<String, double> percentages = {};
```

Figura 3.14: Ejemplo de StreamBuilder.

```dart
class BalanceStream {
  final BehaviorSubject<Balance> _balanceCount = BehaviorSubject<Balance>();
  Stream<Balance> get stream => _balanceCount.stream;

  fetchData() async {
    Balance balance = await CoinbaseController.getBalances();

    _balanceCount.add(balance);
  }
}

final balanceStream = BalanceStream();
```

Figura 3.15: Ejemplo de controllador con Stream.

A continuación, se muestra cómo se distribuye la aplicación enseñando un diagrama de casos de uso (ver figura 3.16) y un diagrama de clases (ver figura 3.17):

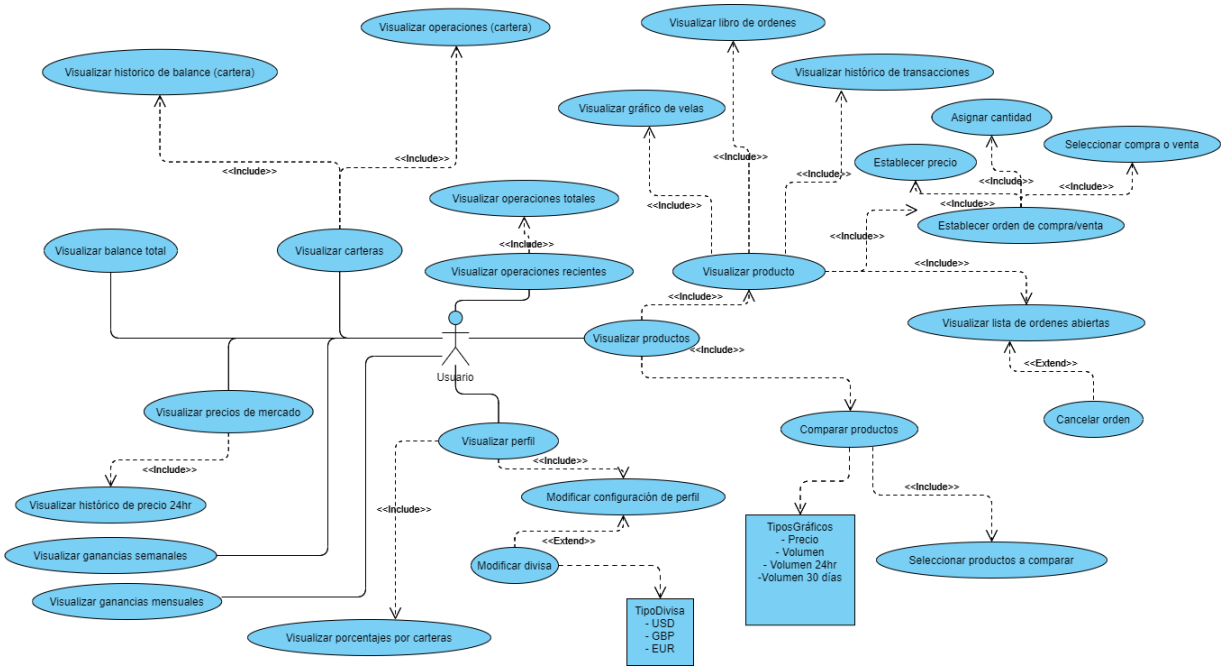

Figura 3.16: Diagrama de casos de uso de la aplicación.

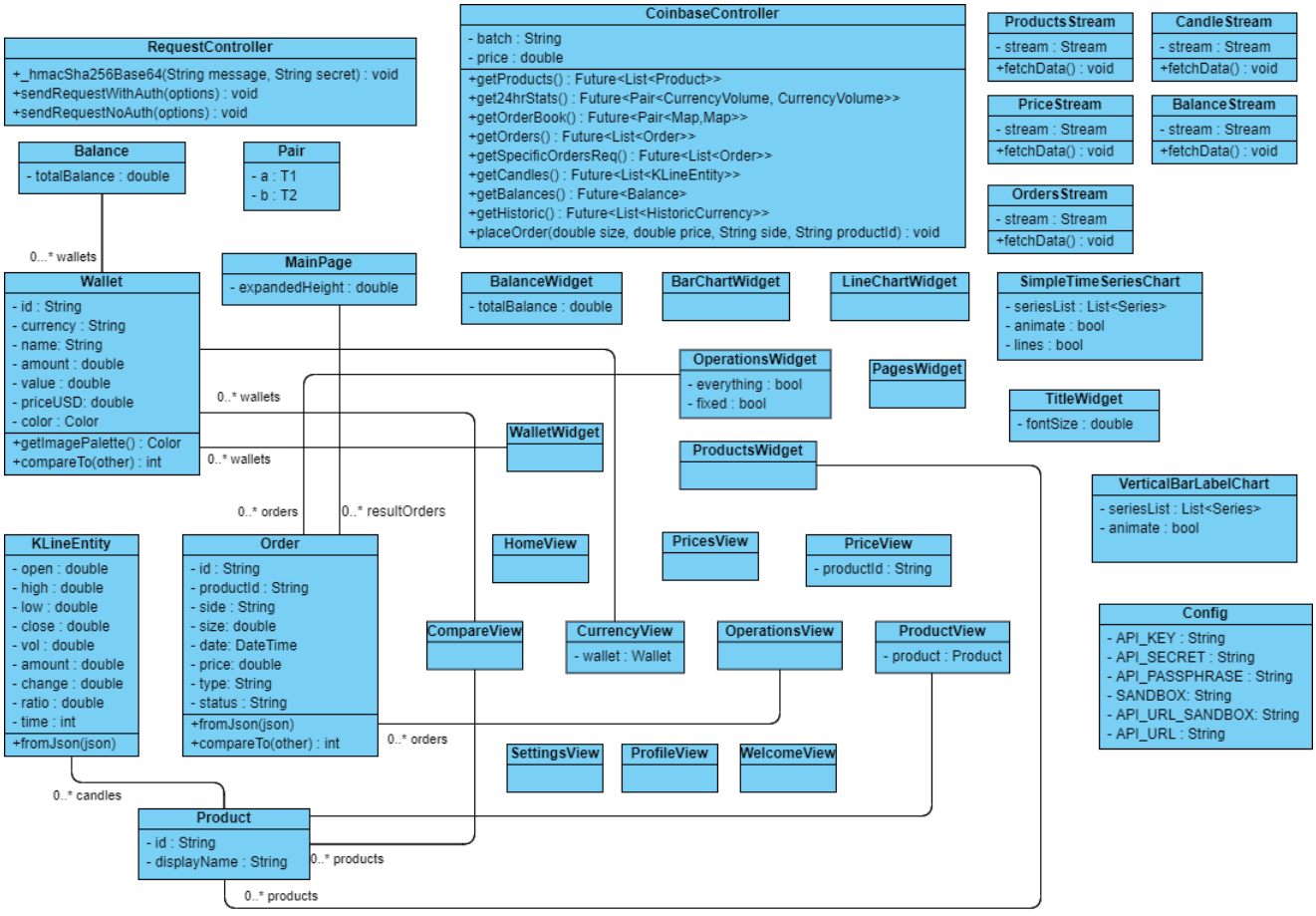

Figura 3.17: Diagrama de clases de la aplicación.

3.5. Comunicación con el *exchange*

Mediante la *API* de *Coinbase Pro* se obtienen y se procesan los datos los cuales se obtienen llamando a *Endpoints* mediante peticiones *Request* y para los datos en tiempo real se usa el *WebSocket* que nos proporciona la plataforma.

Realizaremos la APP en el framework basado en *Dart* llamado *Flutter*, desarrollado por la propia *Google*, el cual nos permite el fácil desarrollo de aplicaciones tanto para *Android*, *iOS* e incluso Web, todo funcionando con el mismo código.

3.5.1. Comunicación con la API

Para la comunicación con la *API*, *Coinbase* nos proporciona una serie de URLs en las que utilizar las distintas *APIs*, *Coinbase* está diferenciado por 2 *exchanges*, uno sería *Coinbase* y el otro *Coinbase Pro*, el primero sería el más básico el cual simplemente permite la visualización simple de distintas monedas y realizar algunas conversión entre monedas, la segunda ya ofrece mucha más información tanto en la *API* como en el exchange, pudiendo visualizar cualquier producto y realizar operaciones más complejas cómo ordenes de compra/venta ya sean de tipo *Limit, Market o Stop Limit* personalizando así nuestras operaciones. Nosotros usaremos la *API* de *Coinbase Pro*[10], en concreto la plataforma nos ofrece 2, una sería la versión oficial y otro sería un *Sandbox* en donde tenemos toda libertad para hacer movimientos ya que se trata de una versión en la que elaborar pruebas sin miedo a perder nuestro activo debido a que se trata de dinero ficticio, entonces será una opción excelente para la realización de nuestro proyecto.

Además de usar la *API Sandbox* en la que hacer las operaciones también usaremos la versión oficial para obtener información de los productos debido a que esta ofrece los datos con mayor frecuencia además de tener más tipos de mercados al tener una actualización en tiempo real será mucho más satisfactoria la visualización de la información. La *API* se encuentra dividida en las dos siguientes secciones:

- Pública
  - Productos
    - Obtener productos
      GET /products
    - Obtener un producto
      GET /products/<*product-id*>

```
Response GET /products/BTC-USD
[
    {
        "id": "BTC-USD",
        "base_currency": "BTC",
        "quote_currency": "USD",
        "base_min_size": "0.001",
        "base_max_size": "10000",
        "quote_increment": "0.01",
        "base_increment": "0.00000001",
        "display_name": "BTC/USD",
        "min_market_funds": "10",
        "max_market_funds": "1000000",
        "margin_enabled": true,
        "fx_stablecoin": false,
        "post_only": false,
```

```
            "limit_only": false,
            "cancel_only": false,
            "trading_disabled": false,
            "status": "online",
            "status_message": ""
        }
    ]
```


- Obtener libro de órdenes para producto
  GET /products/<*product-id*>/book
  - Parámetros

Nivel	Default	Descripción
level	1	Selecciona el nivel de detalle

Tabla 3.1: Parámetros del libro de ordenes.

  - Niveles

Nivel	Descripcion
1	Solo las mejores ofertas y demandas
2	50 mejores ofertas y demandas
3	Libro de ordenes completa

Tabla 3.2: Niveles del libro de ordenes.


```
Response GET /products/BTC-USD/book
    {
        "bids": [
            [
                "31491.05",
                "3009.9993649",
                1
            ]
        ],
        "asks": [
            [
                "31491.07",
                "3009.99968293",
                1
            ]
        ],
        "sequence": 349515464
    }
```


- Obtener precio del producto
  GET /products/<*product-id*>/ticker


```
Response GET /products/BTC-USD/ticker
        {
            "trade_id": 30976440,
            "price": "31689.76",
            "size": "0.00031508",
            "time": "2021-06-22T10:32:12.175915Z",
            "bid": "31689.74",
            "ask": "31689.76",
            "volume": "24807.2601692"
```

```
        }
```

- Obtener transacciones del producto
  GET /products/<*product-id*>/trades

```
Response GET /products/BTC-USD/trades
    [
        {
            "time": "2021-06-22T10:32:22.174Z",
            "trade_id": 30976444,
            "price": "31689.76000000",
            "size": "0.00031508",
            "side": "sell"
        },
        {
            "time": "2021-06-22T10:32:17.165Z",
            "trade_id": 30976443,
            "price": "31689.74000000",
            "size": "0.00031555",
            "side": "buy"
        },
    ]
```

- Obtener ratios históricos (velas)
  GET /products/<*product-id*>/candles

Parámetros	Descripción
start	Tiempo de inicio en ISO 8601
end	Tiempo de fin en ISO 8601
granularity	Rango a seleccionar en segundos

Tabla 3.3: Parámetros de ratios históricos.

```
Response GET /products/BTC-USD/candles
[ time, low, high, open, close, volume ]
    [
        [
            1624357920,
            31689.74,
            31783.31,
            31689.76,
            31783.29,
            0.77443854
        ],
        [
            1624357860,
            31550.01,
            31736.72,
            31550.01,
            31736.7,
            0.53925588
        ],
    ]
```

- Obtener estadísticas de las últimas 24hr
  GET /products/<*product-id*>/stats

```
Response GET /products/BTC-USD/stats
{
    "open": "32345.69",
    "high": "37920",
    "low": "30000",
    "volume": "24808.89764039",
    "last": "31700.03",
    "volume_30day": "12535107.3351462"
}
```

- Monedas
  - Obtener monedas
    GET /currencies
  - Obtener moneda
    GET /currencies/<id>

```
Response GET /currencies/USD
{
    "id": "BTC",
    "name": "Bitcoin",
    "min_size": "0.00000001",
    "status": "online",
    "message": null,
    "details": {
        "type": "crypto",
        "symbol": "",
        "network_confirmations": 6,
        "sort_order": 3,
        "crypto_address_link":
            "https://live.blockcypher.com/btc/address/{{address}}",
        "crypto_transaction_link":
            "https://live.blockcypher.com/btc/tx/{{txId}}",
        "push_payment_methods": [
            "crypto"
        ],
        "min_withdrawal_amount": 0.0001,
        "max_withdrawal_amount": 500,
        "group_types": [
            "btc",
            "crypto"
        ]
    },
    "max_precision": "0.00000001"
}
```

- Tiempo

  GET /time

```
Response GET /time
    {
        "iso": "2021-06-22T10:38:31.312Z",
        "epoch": 1624358311.312
    }
```

- Privada
  - Cuentas o Carteras
    - Listar cuentas

      GET /accounts

Campo	Descripción
id	Account ID
currency	Moneda del cuenta
balance	Total de saldo de la cuenta
holds	Total de saldo no disponible para el uso
available	Total de saldo disponible para el uso
trading_enabled	Si esta el trading activo para la cuenta

Tabla 3.4: Campos de las cuentas.

    - Obtener cuenta

      GET /accounts/<*account-id*>

Campo	Descripción
id	Account ID
balance	Total de saldo de la cuenta
holds	Total de saldo no disponible para el uso
available	Total de saldo disponible para el uso

Tabla 3.5: Campos de una cuenta/cartera.

```
Response GET /accounts/e7c7feb6-a8c1-4132-8b0c-50a84fb63db4
        {
            "id": "e7c7feb6-a8c1-4132-8b0c-50a84fb63db4",
            "currency": "BTC",
            "balance": "51.4310650998693760",
            "hold": "0.0000000000000000",
            "available": "51.431065099869376",
            "profile_id": "a4556006-0d36-44d0-8365-27b401365cf7",
            "trading_enabled": true
        }
```

- Obtener historia de cuenta

  GET /accounts/<*account-id*>/ledger

```
Response GET
    /accounts/e7c7feb6-a8c1-4132-8b0c-50a84fb63db4/ledger
    [
        {
            "id": "307331125",
            "amount": "0.0100000000000000",
            "balance": "51.4310650998693760",
            "created_at": "2021-06-10T10:21:13.006584Z",
            "type": "match",
            "details": {
                "order_id": "ccec0026-29a0-4054-89b8-d618b4ead2c1",
                "product_id": "BTC-USD",
                "trade_id": "30564136"
            }
        },
        {
            "id": "307028257",
            "amount": "0.0100000000000000",
            "balance": "51.4210650998693760",
            "created_at": "2021-06-09T18:37:49.617825Z",
            "type": "match",
            "details": {
                "order_id": "735bbc19-8e49-4a0f-95f4-4fa5fad1c928",
                "product_id": "BTC-USD",
                "trade_id": "30550658"
            }
        },
        ...
    ]
```

- Obtener retenciones
- Ordenes de compra/venta
  - Establecer orden

    POST /orders

Parámetro	Descripción
client_oid	[opcional] ID de orden elegida por usuario
type	[opcional] limit o market (predeterminada limit)
side	buy o sell (compra o venta)
product_id	ID del producto
stp	[opcional] Flag para prevención de autotradeo
stop	[opcional] loss o entry. Requiere stop_price
stop_price	[opcional] Disparador para la orden stop

Tabla 3.6: Parámetros de orden.

```
Request POST /orders
    {
        "size": "0.01",
        "price": "0.100",
        "side": "buy",
        "product_id": "BTC-USD"
```

```
    }
Response POST /orders
    {
        "id": "d0c5340b-6d6c-49d9-b567-48c4bfca13d2",
        "price": "0.10000000",
        "size": "0.01000000",
        "product_id": "BTC-USD",
        "side": "buy",
        "stp": "dc",
        "type": "limit",
        "time_in_force": "GTC",
        "post_only": false,
        "created_at": "2016-12-08T20:02:28.53864Z",
        "fill_fees": "0.0000000000000000",
        "filled_size": "0.00000000",
        "executed_value": "0.0000000000000000",
        "status": "pending",
        "settled": false
    }
```

◇ Cancelar orden

DELETE /orders/<id>

Parámetro	Predeterminado	Descripción
product_id	[opcional]	ID del producto para acelerar el proceso

Tabla 3.7: Parámetros de cancelación de orden.

```
Response DELETE
    /orders/d0c5340b-6d6c-49d9-b567-48c4bfca13d2
    "d0c5340b-6d6c-49d9-b567-48c4bfca13d2"
```

◇ Cancelar todas las ordenes

DELETE /orders

Parámetro	Predeterminado	Descripción
product_id	[opcional]	ID del producto a eliminar ordenes

Tabla 3.8: Parámetros de cancelación de ordenes.

◇ Listar ordenes

GET /orders

Parámetro	Predeterminado	Descripción
status	[open, pending, active, all]	Muestra los que cumple los estados
product_id	[opcional]	ID del producto

Tabla 3.9: Parámetros de listado de ordenes.

```
Response GET /orders
[
    {
        "id": "d0c5340b-6d6c-49d9-b567-48c4bfca13d2",
        "price": "0.10000000",
        "size": "0.01000000",
        "product_id": "BTC-USD",
```

```
            "side": "buy",
            "stp": "dc",
            "type": "limit",
            "time_in_force": "GTC",
            "post_only": false,
            "created_at": "2016-12-08T20:02:28.53864Z",
            "fill_fees": "0.0000000000000000",
            "filled_size": "0.00000000",
            "executed_value": "0.0000000000000000",
            "status": "open",
            "settled": false
        },
    ...
    ]
```


◇ Obtener orden

GET /orders/<id>


```
Response GET
    /orders/68e6a28f-ae28-4788-8d4f-5ab4e5e5ae08
    {
        "id": "68e6a28f-ae28-4788-8d4f-5ab4e5e5ae08",
        "size": "1.00000000",
        "product_id": "BTC-USD",
        "side": "buy",
        "stp": "dc",
        "funds": "9.9750623400000000",
        "specified_funds": "10.0000000000000000",
        "type": "market",
        "post_only": false,
        "created_at": "2016-12-08T20:09:05.508883Z",
        "done_at": "2016-12-08T20:09:05.527Z",
        "done_reason": "filled",
        "fill_fees": "0.0249376391550000",
        "filled_size": "0.01291771",
        "executed_value": "9.9750556620000000",
        "status": "done",
        "settled": true
    }
```


- Ordenes completadas
- Límites
- Depósitos
  - Listar depósitos

    GET /transfers

Parámetro	Requerido	Descripción
type	no	deposit o internal_deposit
product_id	no	ID del producto
before	no	Devuelve los anteriores a partir de una fecha
after	no	Devuelve los posteriores a partir de una fecha
limit	no	Limitar el número de respuesta

Tabla 3.10: Parámetros de listado de depósitos.

- Depositar a través de método de pago
  POST /deposits/payment-method

Parámetro	Descripción
amount	Cantidad del depósito
currency	Tipo de moneda
payment_method_id	ID del método de pago

Tabla 3.11: Parámetros de depósito a través de método de pago.

```
Request POST /deposits/payment-method
    {
        "amount": 10.00,
        "currency": "USD",
        "payment_method_id":
            "bc677162-d934-5f1a-968c-a496b1c1270b"
    }

Response POST /deposits/payment-method
    {
        "id": "593533d2-ff31-46e0-b22e-ca754147a96a",
        "amount": "10.00",
        "currency": "USD",
        "payout_at": "2016-08-20T00:31:09Z"
    }
```

- Depositar a través de cuenta Coinbase
  POST /deposits/coinbase-method

Parámetro	Descripción
amount	Cantidad del depósito
currency	Tipo de moneda
coinbase_account_id	ID de cuenta Coinbase

Tabla 3.12: Parámetros de depósito a través de Coinbase.

```
Request POST /deposits/coinbase-method
    {
        "amount": 10.00,
        "currency": "BTC",
        "coinbase_account_id":
            "c13cd0fc-72ca-55e9-843b-b84ef628c198",
    }

Response POST /deposits/coinbase-method
    {
        "id": "593533d2-ff31-46e0-b22e-ca754147a96a",
        "amount": "10.00",
        "currency": "BTC",
    }
```

- Retiros
  - Listar retiros

    GET /transfers

Parámetro	Requerido	Descripción
type	no	withdraw o internal_withdraw
product_id	no	ID del producto
before	no	Devuelve los anteriores a partir de una fecha
after	no	Devuelve los posteriores a partir de una fecha
limit	no	Limitar el número de respuesta

Tabla 3.13: Parámetros de listado de retiradas.

  - Listar un retiro
  - Retirar a método de pago

    POST /withdrawals/payment-method

Parámetro	Descripción
amount	Cantidad del depósito
currency	Tipo de moneda
payment_method_id	ID del método de pago

Tabla 3.14: Parámetros de retirada a través de método de pago.

```
Request POST /withdrawals/payment-method
    {
        "amount": 10.00,
        "currency": "USD",
        "payment_method_id":
            "bc677162-d934-5f1a-968c-a496b1c1270b"
    }

Response POST /withdrawals/payment-method
    {
        "id": "593533d2-ff31-46e0-b22e-ca754147a96a",
        "amount": "10.00",
        "currency": "USD",
        "payout_at": "2016-08-20T00:31:09Z"
    }
```

  - Retirar a cuenta Coinbase

    POST /withdrawals/coinbase-method

Parámetro	Descripción
amount	Cantidad del depósito
currency	Tipo de moneda
coinbase_account_id	ID de cuenta Coinbase

Tabla 3.15: Parámetros de retirada a través de Coinbase.

```
Request POST /withdrawals/coinbase-account
    {
        "amount": 10.00,
        "currency": "BTC",
```

```
        "coinbase_account_id":
            "c13cd0fc-72ca-55e9-843b-b84ef628c198",
    }

Response POST /withdrawals/coinbase-account
    {
        "id":"593533d2-ff31-46e0-b22e-ca754147a96a",
        "amount":"10.00",
        "currency": "BTC",
    }
```

- Retirar a dirección de cripto
  POST /withdrawals/crypto

Parámetro	Descripción
amount	Cantidad del depósito
currency	Tipo de moneda
crypto_address	Dirección de cartera cripto

Tabla 3.16: Parámetros de retirada a través de cartera cripto.

```
Request POST /withdrawals/crypto
 {
    "amount": 10.00,
    "currency": "BTC",
    "crypto_address":
        "0x5ad5769cd04681FeD900BCE3DDc877B50E83d469"
}

Response POST /withdrawals/crypto
{
    "id":"593533d2-ff31-46e0-b22e-ca754147a96a",
    "amount":"10.01",
    "currency": "BTC",
    "fee": ".01",
    "subtotal": "10.00",
}
```

- Conversiones a monedas estables
  - Crear conversión
    POST /conversions

Parámetro	Descripción
from	ID de producto válido
to	ID de producto válido
amount	Cantidad a transferir (max 10.000.000)

Tabla 3.17: Parámetros de conversión de monedas estables.

- Métodos de pago
  - Listar métodos de pago
    GET /payment-methods
- Cuentas de Coinbase
  - Listar cuentas de Coinbase
    GET /coinbase-accounts

```
Response GET /coinbase-accounts
    [
        {
            "id":
                "fc3a8a57-7142-542d-8436-95a3d82e1622",
            "name": "ETH Wallet",
            "balance": "0.00000000",
            "currency": "ETH",
            "type": "wallet",
            "primary": false,
            "active": true
        },
        ...
        ]
```

- Comisiones
  - Obtener comisiones actuales

    GET /fees

```
Response GET /fees
    [
        {
            "maker_fee_rate": "0.0015",
            "taker_fee_rate": "0.0025",
            "usd_volume": "25000.00"
        }
    ]
```

- Reportes
  - Crear nuevo reporte

    POST /reports

Parámetro	Descripción
type	fills o account
product_id	ID del producto
account_id	ID de la cuenta
start_date	Fecha de inicio del reporte
end_date	Fecha de fin del reporte
format	pdf o csv
email	[opcional] Dirección de correo al que enviar

Tabla 3.18: Parámetros de creación de reporte.

  - Obtener estado de reporte
- Perfiles
  - Listar perfiles

    GET /profiles

Parámetro	Descripción
active	Devolver solo perfiles activos

Tabla 3.19: Parámetros de listado de perfiles.

```
Response GET /profiles
    [
        {
            "id":
                "86602c68-306a-4500-ac73-4ce56a91d83c",
            "user_id": "5844eceecf7e803e259d0365",
            "name": "default",
            "active": true,
            "is_default": true,
            "created_at":
                "2019-11-18T15:08:40.236309Z"
        }
    ]
```

- Obtener perfil
- Transferir perfil

  GET /profiles/transfer

Parámetro	Descripción
from	ID del perfil inicio
to	ID del perfil destino
currency	Moneda a transferir
amount	Cantidad a transferir

Tabla 3.20: Parámetros de transferencia de perfil.

```
Response POST /profiles/transfer
{
    "from": "86602c68-306a-4500-ac73-4ce56a91d83c",
    "to": "e87429d3-f0a7-4f28-8dff-8dd93d383de1",
    "currency": "BTC",
    "amount": "1000.00"
}
```

Además para la obtención de los precios se usa la *API de Coinbase* básica, teniendo así el precio de cualquiera de las monedas en USD. (*https://api.coinbase.com/v2/prices/$currency-USD/spot*).

Una vez tengamos nuestra cuenta en *Coinbase Pro* para poder usar la *API* necesitaremos crear una clave para acceder a ella, estando en la web de *Coinbase Pro* clickamos en la zona superior derecha y seleccionamos el apartado de *API* (ver figura 3.18).

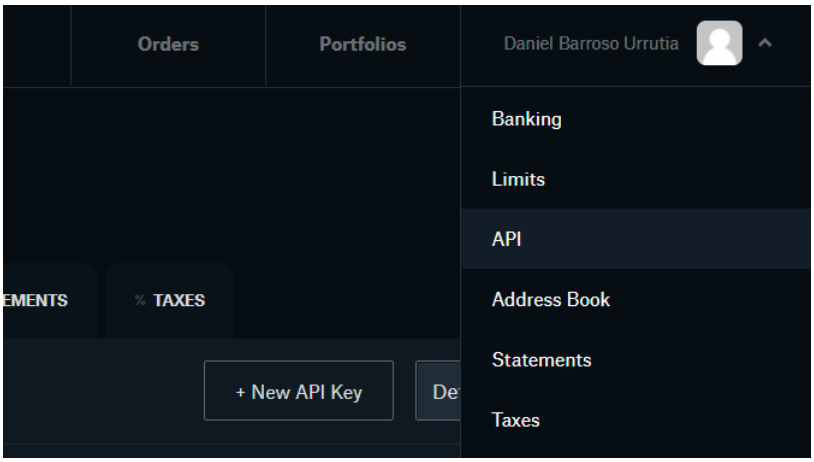

Figura 3.18: Sección de configuración de API.

Dentro de la sección hacemos click en *"New API Key"* y nos aparecerá una nueva ventana en la que podemos introducir los datos para generar nuestra *API*, serían los siguientes.

- Seleccionar portfolio/cartera
- Nombre de la llave [opcional]
- Permisos
  - Vista
  - Transferencia
  - Trade/Intercambio
- Contraseña
- Lista blanca de IPs [opcional]

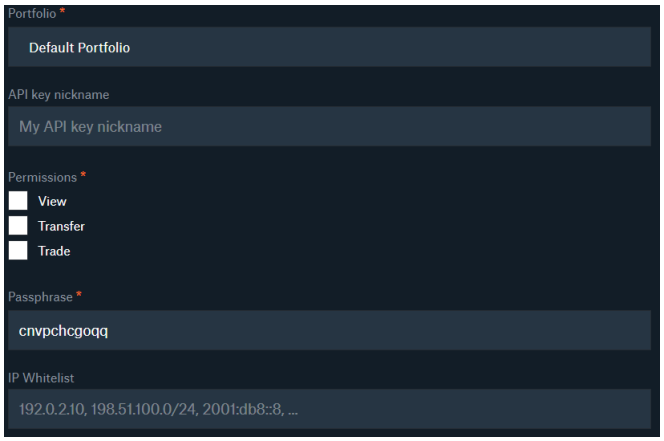

Figura 3.19: Sección de creación de claves para API.

Después de crear nuestra llave, necesitaremos almacenar 3 datos, los almacenaremos en un archivo *.env* donde guardar nuestras variables de entorno (ver figura 3.20):

- Key (clave pública)

- Secret (clave privada)
- Passphrase (contraseña)

```
.env
1  API_KEY = f5e39c91cc9636966d2a630f259f9cbc
2  API_SECRET = Uhfp4rqy69aeoKotpjV2KoXxXH+It05yHiIjwFwrRTlk115XYGukuoSkFByiFE9+4X0DgIHmIk+btMwnvBxGyg==
3  API_PASSPHRASE = vwvomszp10d
4  SANDBOX = true
```

Figura 3.20: Variables de entorno.

Una vez teniendo las claves para poder hacer peticiones a la *API* privada se deben hacer siguiendo las instrucciones que nos indica la documentación de *Coinbase*. Siguiendo estos pasos:

- Obtener tiempo actual exacto de la API (timestamp).
- Al timestamp le añadimos el método de la request (GET, POST, DELETE, etc), el end-point al que queremos acceder y el body si tiene.
- De esta petición se obtiene la "firma" usando SHA256 y Base64, a partir de nuestra clave secreta.

```
var timestamp = await get('https://api.coinbase.com/v2/time')
    .then((res) => json.decode(res.body))
    .then((res) => res['data']['epoch']);
String query = timestamp.toString() +
    options['method'] +
    options['endPoint'] +
    options['body'];
String signature = _hmacSha256Base64(query, Config.API_SECRET);
var url = Config.API_URL_SANDBOX + options['endPoint'];
```

Figura 3.21: Método de creación de firma para Request.

Teniendo ya la *firma* ya podríamos hacer la petición con la cabecera correspondiente y la url con el *endpoint* al que queremos acceder.

- CB-ACCESS-KEY, Nuestra clave privada.
- CB-ACCESS-SIGN, Firma realizada anteriormente.
- CB-ACCESS-TIMESTAMP, Fecha actual obtenida.
- CB-ACCESS-PASSPHRASE, Contraseña de nuestra clave.

```
response = await get(url, headers: {
  'CB-ACCESS-KEY': Config.API_KEY,
  'CB-ACCESS-SIGN': signature,
  'CB-ACCESS-TIMESTAMP': timestamp.toString(),
  'CB-ACCESS-PASSPHRASE': Config.API_PASSPHRASE,
});
```

Figura 3.22: Creación de petición Request.

3.5.2. Comunicación con *WebSockets*

Además, para la obtención de los datos en tiempo real se ha usado el *WebSocket*, primero nos conectamos a él en la dirección que nos dan (*'wss://ws-feed.pro.coinbase.com'*), una vez conectados se debe elegir el canal o canales que queremos escuchar, los canales son los siguientes:

- Heartbeat

```
Request
    {
        "type": "subscribe",
        "channels": [{ "name": "heartbeat", "product_ids": ["ETH-EUR"] }]
    }
Response
    {
        "type": "heartbeat",
        "sequence": 90,
        "last_trade_id": 20,
        "product_id": "BTC-USD",
        "time": "2014-11-07T08:19:28.464459Z"
    }
```

- Status (Estado)

```
Request
    {
        "type": "subscribe",
        "channels": [{ "name": "status"}]
    }
Response
    {
        "type": "status",
        "products": [
            {
                "id": "BTC-USD",
                "base_currency": "BTC",
                "quote_currency": "USD",
                "base_min_size": "0.001",
                "base_max_size": "70",
                "base_increment": "0.00000001",
                "quote_increment": "0.01",
                "display_name": "BTC/USD",
                "status": "online",
                "status_message": null,
                "min_market_funds": "10",
                "max_market_funds": "1000000",
                "post_only": false,
                "limit_only": false,
                "cancel_only": false
            }
        ],
        "currencies": [
            {
                "id": "USD",
                "name": "United States Dollar",
                "min_size": "0.01000000",
                "status": "online",
```

```
                "status_message": null,
                "max_precision": "0.01",
                "convertible_to": ["USDC"], "details": {}
            },
        ]
    }
```

- Ticker (Cotización). Snapshots del último precio para los productos indicados.

```
Request
    {
      "type": "subscribe",
      "product_ids": ["BTC-USD"],
      "channels": ["ticker"]
    }
Response
    {
        "type": "ticker",
        "trade_id": 20153558,
        "sequence": 3262786978,
        "time": "2017-09-02T17:05:49.250000Z",
        "product_id": "BTC-USD",
        "price": "4388.01000000",
        "side": "buy", // Taker side
        "last_size": "0.03000000",
        "best_bid": "4388",
        "best_ask": "4388.01"
    }
```

- Level2. Snapshots de las ordenes establecidas para los productos indicados.

```
Request
    {
      "type": "subscribe",
      "product_ids": ["BTC-USD"],
      "channels": ["level2"]
    }
Response
    {
      "type": "l2update",
      "product_id": "BTC-USD",
      "time": "2019-08-14T20:42:27.265Z",
      "changes": [
        [
          "buy",
          "10101.80000000",
          "0.162567"
        ]
      ]
    }
```

- User, si se utiliza autenticación se pueden obtener en tiempo real las modificaciones para un propio usuario.
- Matches, se producen cuando se realiza una transacción entre dos ordenes

```
Request
    {
      "type": "subscribe",
      "product_ids": ["BTC-USD"],
      "channels": ["matches"]
    }
Response
   {
       "type": "match",
       "trade_id": 10,
       "sequence": 50,
       "maker_order_id": "ac928c66-ca53-498f-9c13-a110027a60e8",
       "taker_order_id": "132fb6ae-456b-4654-b4e0-d681ac05cea1",
       "time": "2014-11-07T08:19:27.028459Z",
       "product_id": "BTC-USD",
       "size": "5.23512",
       "price": "400.23",
       "side": "sell"
    }
```

- Full, Canal en el que se incluyen todo tipo de snapshots.

```
Request
    {
      "type": "subscribe",
      "product_ids": ["BTC-USD"],
      "channels": ["full"]
    }
```

Pudiendo ser de los siguientes tipos:

- Received (Recibida), se actualiza cuando se recibe una orden válida y está ahora activa, pueden ser tanto *limit* como *market*.

```
Response
    {
        "type": "received",
        "time": "2014-11-07T08:19:27.028459Z",
        "product_id": "BTC-USD",
        "sequence": 10,
        "order_id": "d50ec984-77a8-460a-b958-66f114b0de9b",
        "size": "1.34",
        "price": "502.1",
        "side": "buy",
        "order_type": "limit"
    },
    {
        "type": "received",
        "time": "2014-11-09T08:19:27.028459Z",
        "product_id": "BTC-USD",
        "sequence": 12,
        "order_id": "dddec984-77a8-460a-b958-66f114b0de9b",
        "funds": "3000.234",
        "side": "buy",
        "order_type": "market"
    },
```

- Open (Abierta), se recibe una orden abierta en el libro de ordenes.

```
Response
    {
        "type": "open",
        "time": "2014-11-07T08:19:27.028459Z",
        "product_id": "BTC-USD",
        "sequence": 10,
        "order_id": "d50ec984-77a8-460a-b958-66f114b0de9b",
        "price": "200.2",
        "remaining_size": "1.00",
        "side": "sell"
    },
```

- Done (Finalizada), se recibe una orden que ya no pertenece al libro de ordenes.

```
Response
    {
        "type": "done",
        "time": "2014-11-07T08:19:27.028459Z",
        "product_id": "BTC-USD",
        "sequence": 10,
        "price": "200.2",
        "order_id": "d50ec984-77a8-460a-b958-66f114b0de9b",
        "reason": "filled", // or "canceled"
        "side": "sell",
        "remaining_size": "0"
    },
```

- Match (Acuerdo), se produce cuando se realiza una transacción entre dos ordenes.

```
Response
    {
        "type": "match",
        "trade_id": 10,
        "sequence": 50,
        "maker_order_id": "ac928c66-ca53-498f-9c13-a110027a60e8",
        "taker_order_id": "132fb6ae-456b-4654-b4e0-d681ac05cea1",
        "time": "2014-11-07T08:19:27.028459Z",
        "product_id": "BTC-USD",
        "size": "5.23512",
        "price": "400.23",
        "side": "sell"
    },
```

- Change (Cambio), se recibe cuando se realiza un cambio en una orden ya establecida, por ejemplo, un cambio en la cantidad de esta.

```
Response
    {
        "type": "change",
        "time": "2014-11-07T08:19:27.028459Z",
        "sequence": 80,
        "order_id": "ac928c66-ca53-498f-9c13-a110027a60e8",
        "product_id": "BTC-USD",
        "new_size": "5.23512",
        "old_size": "12.234412",
        "price": "400.23",
        "side": "sell"
```

```
},
```

- Activate (Activar), estas se realizar cuando se establece una orden tipo parada (Stop), la cual activa otra orden a realizar.

```
Response
    {
      "type": "activate",
      "product_id": "test-product",
      "timestamp": "1483736448.299000",
      "user_id": "12",
      "profile_id": "30000727-d308-cf50-7b1c-c06deb1934fc",
      "order_id": "7b52009b-64fd-0a2a-49e6-d8a939753077",
      "stop_type": "entry",
      "side": "buy",
      "stop_price": "80",
      "size": "2",
      "funds": "50",
      "private": true
    }
```

Para este proyecto se han utilizado dos canales, el canal de *ticker* para la obtención de precios y el canal de *level2*, que es el que *Coinbase* recomienda utilizar para actualizar los libros de ordenes ya que se ofrece la información mínima necesaria para ello.

3.6. Diseño de la aplicación

El diseño del aplicativo se ha planteado de una manera para que sea lo más amigable posible, intentando que tenga una visión moderna y colorida para hacer la experiencia de uso más satisfactoria. La aplicación se divide en una serie de vistas:

- **Bienvenida**, pantalla de bienvenida donde se presenta la temática de la aplicación, enseñando el título de la app como el tono que predominará en la misma.

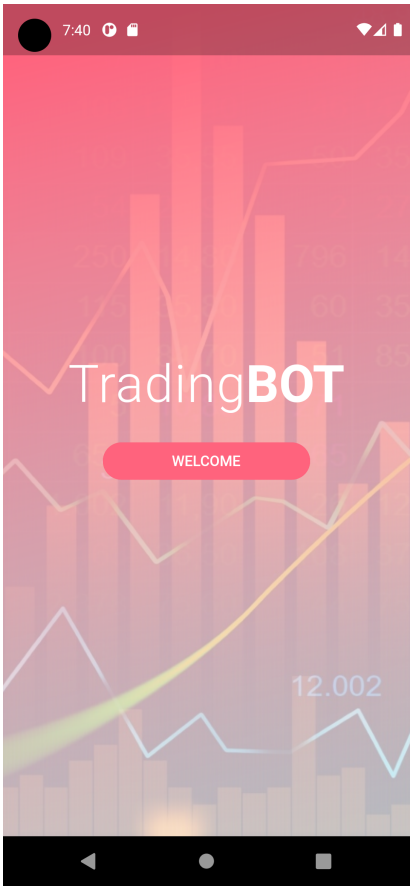

Figura 3.23: Vista principal superior.

- **Principal**, una superior donde y otra inferior en donde podemos seleccionar entre 2 secciones, donde podemos observar información del estado del los distintos mercados y productos.

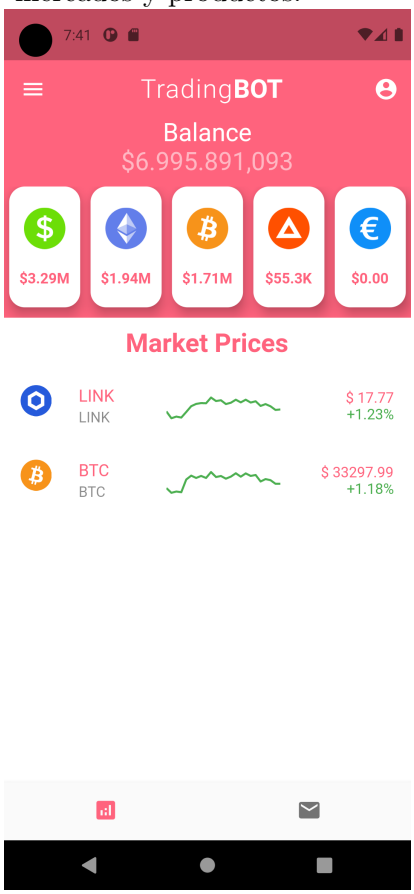

Figura 3.24: Vista principal.

3.6.1. Vista superior.

Se puede observar el balance actual en nuestra cartera en la divisa que seleccionemos en la configuración *(USD, EUR, GBP)* y un listado de la cantidad que disponemos de cada una de las monedas.

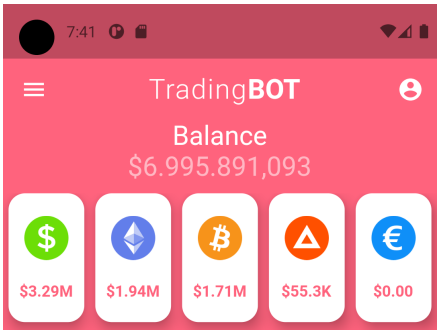

Figura 3.25: Vista principal superior.

Tendría una apartado superior donde como hemos comentado se muestra el balance *total* sumado entre todas monedas que nos pertenecen.

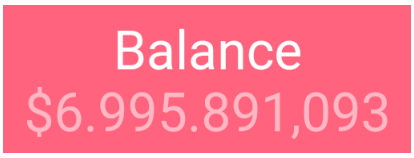

Figura 3.26: Vista principal superior del balance total.

Inmediatamente debajo tendríamos un listado de todas las monedas disponibles, mostrando el icono de la moneda y la cantidad en la divisa para cada una de ellas, pudiendo deslizar horizontalmente para poder observar todas.

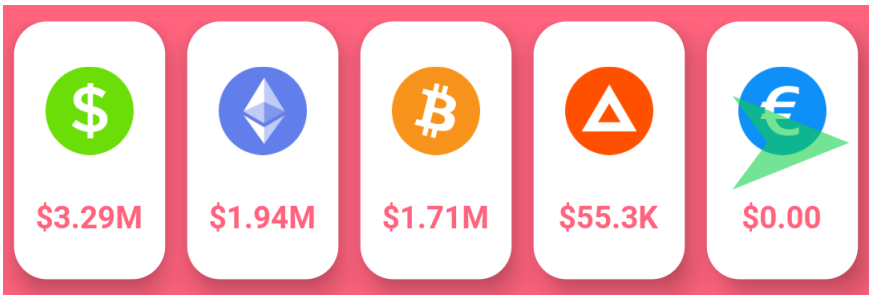

Figura 3.27: Vista principal superior de las carteras.

Desde esta zona superior se puede navegar hacia 2 nuevas vistas, una haciendo click en la zona superior derecha para entrar en la *Vista de Perfil* y la otra haciendo click en cualquiera de las "cartas" de alguna moneda *Vista de Moneda*.

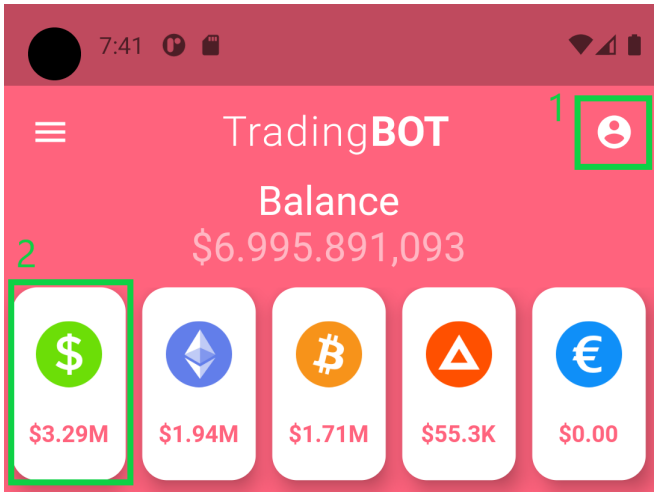

Figura 3.28: Vista principal superior con aclaración.

- Vista de Perfil.
  Esta sería la vista de nuestro perfil el cual podemos ver nuestra imagen de perfil, nombre y un resumen de nuestra cartera mediante un gráfico en forma de *donut* pudiendo ver de forma rápida los porcentajes de cada una de nuestras monedas, además de un listado de todas dando el número detallado.

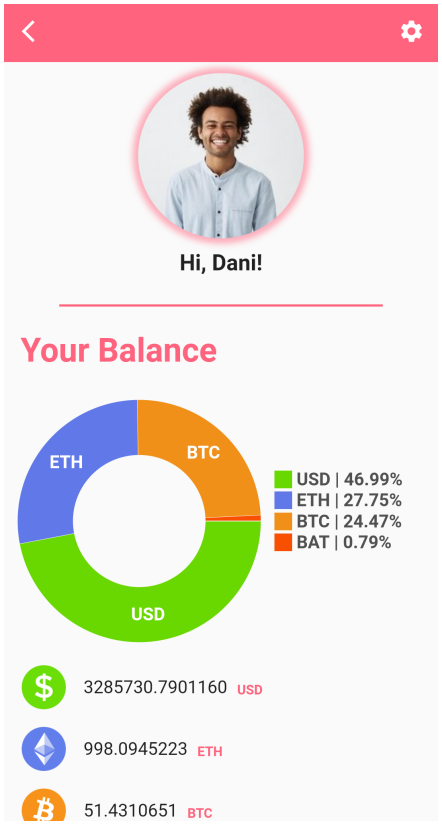

Figura 3.29: Vista de Perfil.

Haciendo click en la sección superior derecha (icono engranaje), navegariamos a la vista en la cual podriamos cambiar la configuración acerca de nuestra cuenta, divisa a elegir (USD, EUR, GBP), notificaciones, cerrar sesión, etc...

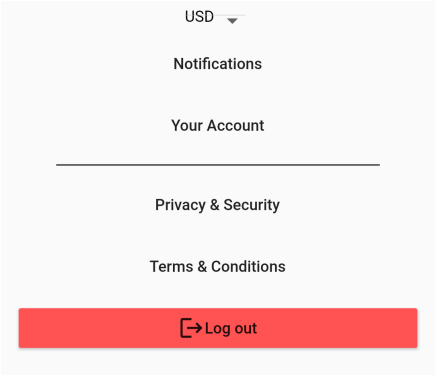

Figura 3.30: Vista de Configuración.

- Vista de Moneda.
  Esta vista está dedicada a poder visualizar el historial de cada moneda en nuestra cuenta, en la parte superior podemos observar un gráfico de lineas en el cual podemos visualizar el histórico de la cantidad que hemos tenido a lo largo del tiempo. Inmediatamente más abajo veremos un listado de las operaciones que se han realizado con esta moneda, pudiendo observar el *producto* utilizado si la operación ha sido de compra o venta, la cantidad y la fecha y hora en el cual se realizó.

Figura 3.31: Vista de Moneda.

3.6.2. Vista inferior.

En donde podemos seleccionar entre 2 secciones, donde podemos observar información del estado del los distintos mercados y productos.

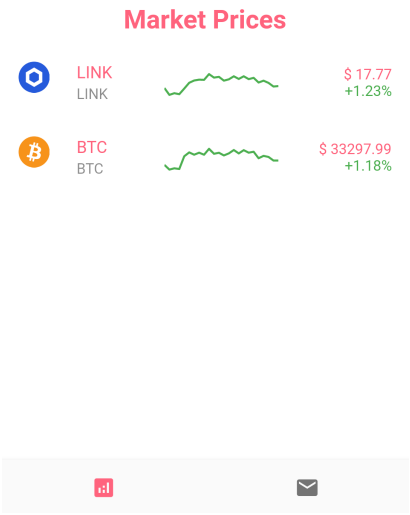

Figura 3.32: Vista principal inferior de precios de mercado.

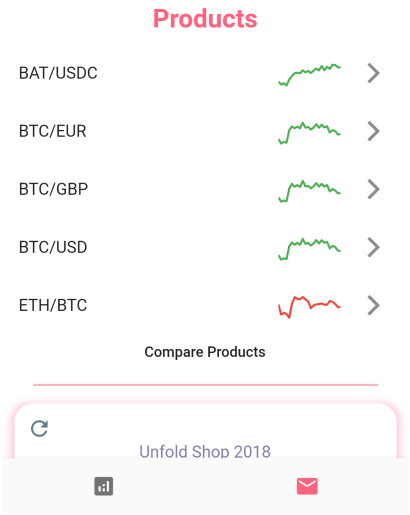

Figura 3.33: Vista principal inferior de productos.

- Vista de precios de mercado
  En esta vista se mostrará un listado en donde podemos observar de forma rápida los precios de mercado para los productos disponibles para nuestra divisa, cada elemento de esta lista contiene. Se puede observar la moneda, un gráfico lineal simplificado mostrando el precio de las últimas 24hr, verde o rojo en función de la diferencia de precio, además se muestra el precio actual y el porcentaje de cambio del precio en las últimas 24hr.

Figura 3.34: Sección de elemento en precios de mercado.

Si hacemos click en alguna de ellas accedemos a otra vista específica para el producto en el cual podemos ver de forma más detallada el precio.

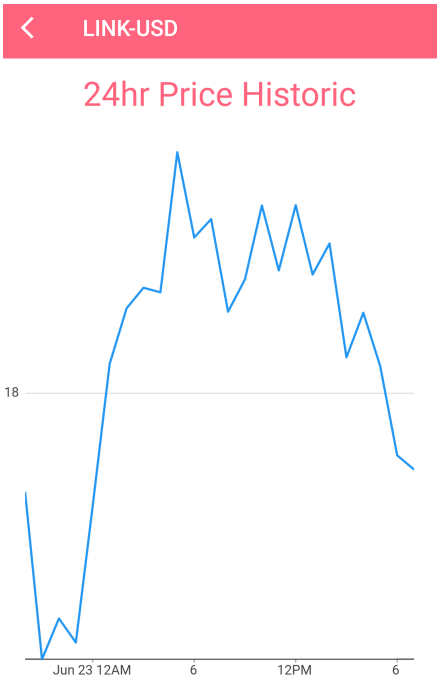

Figura 3.35: Vista de precio de mercado en las últimas 24hr.

- Vista de productos.
  Dentro de la vista principal como ya hemos comentado tenemos otra sección en donde encontraremos información detallada acerca de los productos. Primeramente, encontramos un listado de todos los productos disponibles en donde más adelante podremos comparar distintos variables entre ellos.

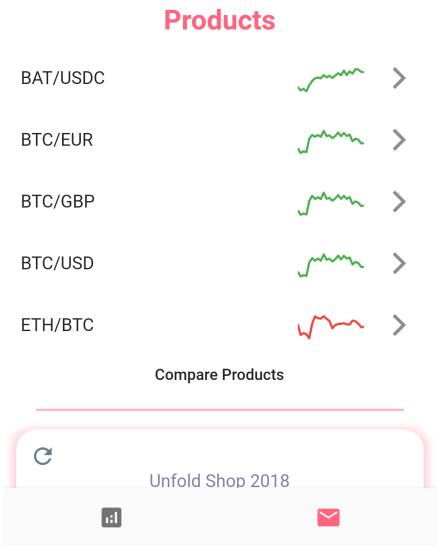

Figura 3.36: Vista de Productos.

Deslizando hacia abajo encontramos dos gráficos en los cuales podemos observar las ganancias o perdidas que se han ido consiguiendo, tanto mensuales como semanales.

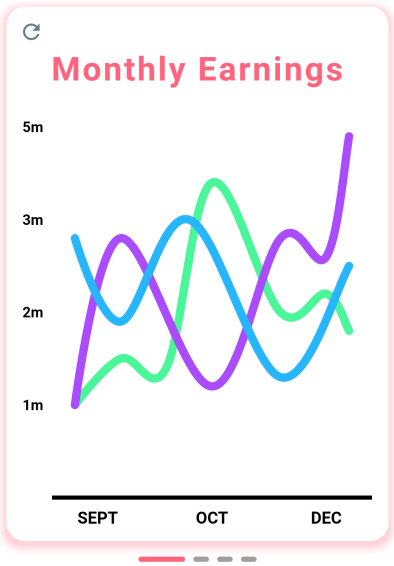

Figura 3.37: Sección de ganancias mensuales.

Figura 3.38: Sección de ganancias semanales.

Debajo tendríamos las 5 operaciones más recientes que se han realizado. Haciendo click en *View more* entramos en la *Vista de operaciones* en la cual podemos ver todas las operaciones que se han realizado, ordenadas de más reciente a más antiguas.

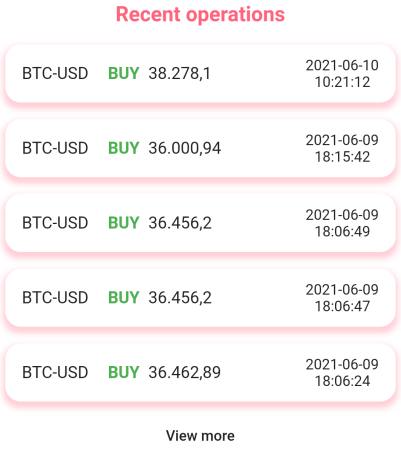

Figura 3.39: Sección de operaciones recientes.

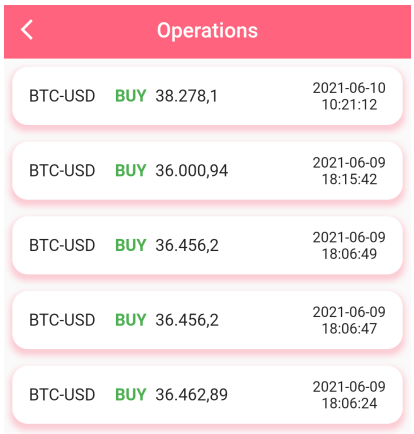

Figura 3.40: Vista operaciones totales.

Si volvemos a la zona superior y seleccionamos cualquier producto, navegamos a la *Vista del producto* específica.

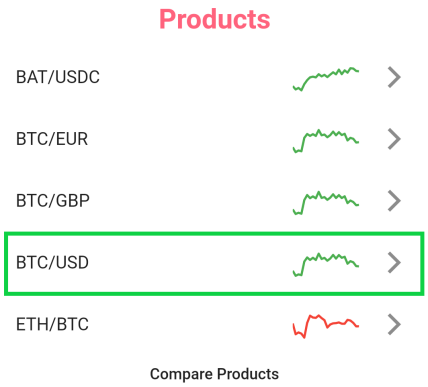

Figura 3.41: Selección de elemento en Vista de productos.

Una vez dentro podemos ver una vista con información mucho más detallada acerca del producto seleccionado, en la zona superior podemos ver el precio que tiene este producto en tiempo real.

Figura 3.42: Vista de Producto.

Esta vista se divide en tres apartados:

- Gráfico de velas
  En este gráfico trata de un *gráfico de velas japonesas* en las que cada *vela* estaría distinguida entre mínimo, máximo (siendo la mecha) y apertura y cierre (siendo el cuerpo), representando verde o rojo en función del precio de cierre de la anterior vela. Cada vela se ejecutan a lo largo de un tiempo establecido, concretamente en nuestra aplicación podemos elegir entre 1 minuto, 5 minutos, 15 minutos, 1 hora, 6 horas o 1

día. Además, se representan 3 lineas de tendencia de media aritmética para las 5, 10 y 20 últimas velas respectivamente.

Figura 3.43: Gráfico de velas.

Si mantenemos pulsado en alguna vela podemos ver los datos de la vela más específicamente, también podemos desplazarnos por la linea de tiempo del gráfico a la vez de poder hacer zoom.

Figura 3.44: Gráfico de velas detallado.

- Libro de órdenes
  Este apartado se distingue en dos partes (compra y venta), teniendo la zona de compra a la izquierda en verde y la zona de venta a la derecha en rojo. Se muestra en tamaño de mercado de cada orden establecida. Además, para hacer de una manera mucho más intuitiva y simple se añade un gráfico de barras en el fondo indicando los "pesos" de los precios en proporción a las demás, pudiendo ver rápidamente hacia donde se inclina el mercado.
  Deslizando hacia la derecha podemos ver un histórico de las transacciones completadas, incluyendo el tamaño de mercado y la fecha, ordenando de más reciente a más antiguas.

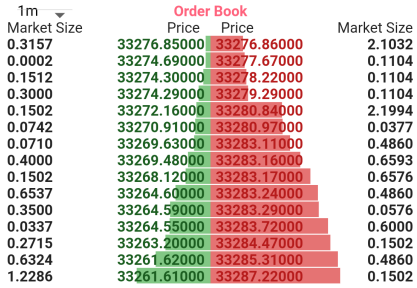

Figura 3.45: Sección de libro de ordenes.

Figura 3.46: Sección de histórico de transacciones.

- Compra/venta
  En la zona inferior de la vista encontramos el apartado donde podemos realizar las operaciones de compra y venta. Primeramente, podemos selecciones si deseamos que sea una operación de venta o de compra, pudiendo ver la cantidad que tenemos de la moneda para operar. Justo debajo seleccionaremos el precio al cual queremos que se realice la operación e debajo de esta tendremos una barra o slider en el cual debemos seleccionar la cantidad que deseamos utilizar en la orden, una vez introducido todos los datos, haciendo click en *Place order*, ejecutaríamos la operación la cual aparecerá en la lista de abajo donde se muestran las operaciones abiertas (hasta que esta se complete o cancele), pudiendo ver cada una de las ordenes abiertas y cancelarlas individualmente.

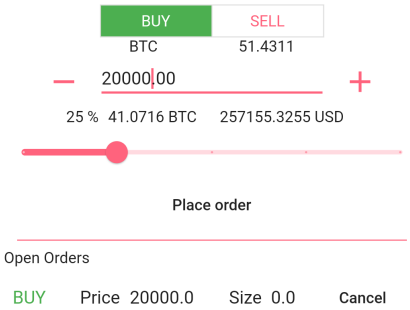

Figura 3.47: Sección de compra/venta de producto.

Volviendo a la *Vista de productos* si hacemos click en el botón *Compare products*, navegamos a la *Vista de comparación* en la cual podremos contrastar distintas variables entre distintas monedas (máx 10), en concreto:

- Precio

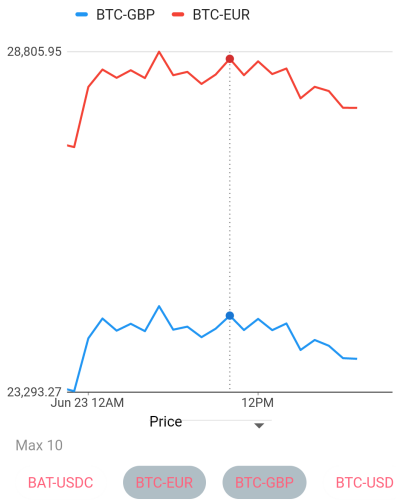

Figura 3.48: Vista de comparación de precios (lineal).

- Volumen

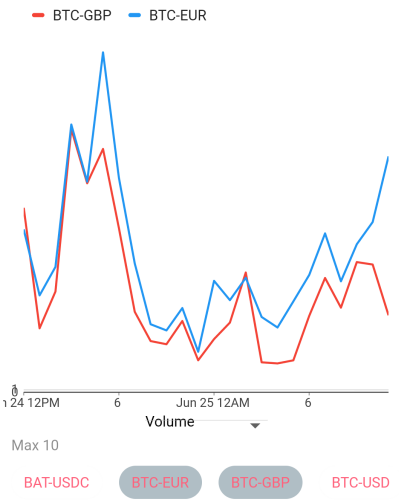

Figura 3.49: Vista de comparación de volúmenes (lineal).

- Volumen últimas 24hr

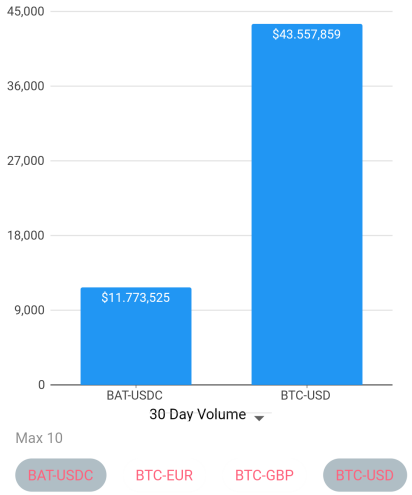

Figura 3.50: Vista de comparación de volúmenes en las últimas 24hr (barras).

- Volumen últimos 30 días

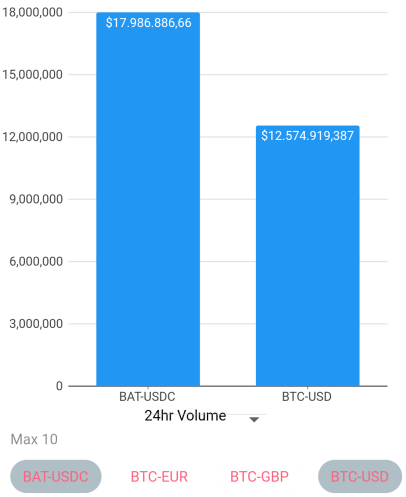

Figura 3.51: Vista de comparación de volúmenes en los últimos 30 días (barras).

Capítulo 4

Conclusiones y trabajo futuro.

Finalizado el proyecto nos disponemos a realizar un análisis del mismo, comparar con proyectos similares e interpretar las posibles mejoras a realizar en un tiempo futuro.

4.1. Conclusión.

Como se vió al inicio de este Trabajo de Fin de Grado se presentó la creación de una aplicación móvil. Primero con el objetivo de cumplir el mismo se realizó una análisis de las necesidades que podrían tener los usuarios para mejorar la experiencia en las plataformas de intercambio de criptomonedas, una vez se realizó esta fase se decidió hacer una *app* multiplataforma para una gran cantidad de usuarios con un solo desarrollo, se plantearon una serie de tecnologías y herramientas a utilizar, según se explicó en el apartado de *Estado del arte* se decidió usar *Flutter* justificando su elección a la vez de *Coinbase* como *exchange*, a todo esto se le añade el uso de distintas metodologías para tener un desarrollo del software adecuado y correcto.

Como conclusión, el proyecto se ha completado de manera satisfactoria, habiendo cumplido los objetivos planteados en primer lugar además de hacerlo de un modo eficiente y funcional. Buscando simplificar la tarea de visualización de mercados de criptomonedas, teniendo una aplicación móvil en la cual ver de forma simple pero completa toda la información, siendo conscientes de que este proyecto puede siempre ser mejorado añadiendo todo tipo de nuevas funcionalidades.

He de destacar que he aprendido ha utilizar el framework de *Flutter* que plantea una forma de desarrollar software innovadora y fresca, que me supondrá un gran añadido a mi portfolio personal y todo ha sido más sencillo debido a la gran cantidad de documentación y tutoriales que ofrecen aún siendo una tecnología relativamente joven, también me ha permitido aprender a gestionar mi tiempo al desarrollar este tipo de proyecto aunque es cierto que se ha tenido que adaptar a los problemas que se han tenido durante la pandemia, sin embargo he conseguido organizarme a las tesituras que han ido surgiendo.

4.2. Publicación del código fuente.

Cómo se ha explicado a lo largo de esta memoria, se ha ido realizando el proyecto haciendo uso de *GitHub* como control de versiones para tener copias del código y poder inspeccionar versiones anteriores. Se ha trabajado con en un repositorio[33] *GitHub* utilizando distintas ramas teniendo una rama principal *master*, una de desarrollo *dev* y las distintas ramas en función de las distintas funciones a implementar (ver Figura 4.1).

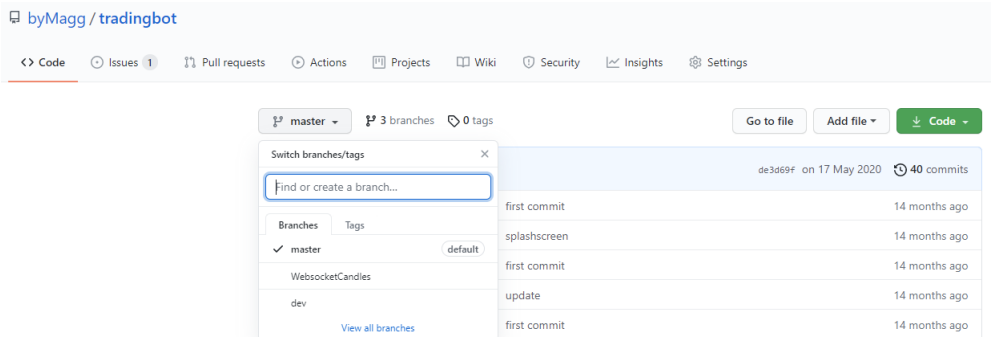

Figura 4.1: Visualización del repositorio GitHub.

4.3. Trabajos futuros.

Para futuros avances en el proyecto se podrían plantear varias opciones, se podría llegar a implementar más tipos de divisas, factores de autenticación biométrica, incluir autenticación con OAuth en lugar de creación de claves, quizás se podría añadir ordenación y filtración de las operaciones por distintos tipos o productos , se puede adaptar un modo oscuro, personalización de paletas en gráficos, implementación de productos favoritos, añadir plataformas de pago y retirada. Incluso sería una gran funcionalidad la de implementar ordenes de tipo *Stop/Loss* pudiendo introduccir un porcentaje de perdida para asegurar las ganancias, también se podría incluir un asistente o BOT que realice operaciones automáticamente en función de distintas técnicas de predicción pudiendo usar *Machine Learning*.

Bibliografía

[1] Caetano Richard. *Learning Bitcoin Embrace the new world of fiance by leveraging the power of crypto-currencies using Bitcoin and the Blockchain.* Packt, 2015.

[2] E. Windmill. *Flutter in Action.* Manning Publications, 2020.

[3] F. Zammetti. *Practical Flutter: Improve your Mobile Development with Google's Latest Open-Source SDK.* Apress, 2019.

[4] A. Biessek. *Flutter for beginners : an introductory guide to building cross-platform mobile applications with flutter and Dart 2.* Packt, 2019.

[5] L. Richardson y col. *RESTful Web APIs: Services for a Changing World.* O'Reilly Media, Inc., 2013.

[6] A. M. Antonopoulos. *Mastering Bitcoin: unlocking digital cryptocurrencies.* O'Reilly Media, Inc., 2014.

[7] Daniel Drescher. *Blockchain basics: a non-technical introduction in 25 steps.* Kube, N., 2018.

[8] J. Buford, H. Yu y E. K Lua. *P2P networking and applications.* Morgan Kaufmann, 2009.

[11] D. Gourley y col. *HTTP: the definitive guide.* O'Reilly Media, Inc., 2002.

[12] A. Lombardi. *WebSocket: lightweight client-server communications.* O'Reilly Media, Inc., 2015.

[30] Brian Vanderjack. *The Agile edge : managing projects effectively using Agile Scrum.* Business Expert Press, 2015.

[31] L. . P. J. F. Debrauwer. *Patrones de diseño en Java : los 23 modelos de diseño : descripción y soluciones ilustradas en UML 2 y Java, Barcelona.* Eni, 2013.

Webgrafía

[9] Century Coin Company Inc. *About HashCash.* URL: http://www.hashcash.com/about.

[10] Coinbase. *Documentación de Coinbase Pro.* URL: https://docs.pro.coinbase.com/.

[13] Binance. *Página oficial de Binance Exchange.* URL: https://www.binance.com/es.

[14] Coinbase. *Página oficial de Coinbase Exchange.* URL: https://www.coinbase.com/es/.

[15] XTB. *Página oficial de XTB.* URL: https://www.xtb.com/es.

[16] eToro. *Página oficial de eToro.* URL: https://www.etoro.com/es/.

[17] Vivid. *Página oficial de Vivid.* URL: https://vivid.money/es-es/.

[18] TradingView. *Página oficial de TradingView.* URL: https://es.tradingview.com/.

[19] Google. *Página oficial de Google Skia.* URL: https://skia.org/.

[20] Facebook Inc. *Página oficial de React Native.* URL: https://reactnative.dev/.

[21] Ionic. *Página oficial de Ionic Framework.* URL: https://ionicframework.com/.

[22] Mozilla. *¿Qué es HTML?.* URL: https://developer.mozilla.org/es/docs/Web/HTML.

[23] Mozilla. *¿Qué es CSS?.* URL: https://developer.mozilla.org/es/docs/Web/CSS.

[24] Mozilla. *¿Qué es JavaScript?.* URL: https://developer.mozilla.org/es/docs/Learn/JavaScript/First_steps/What_is_JavaScript.

[25] Google. *Página oficial de Angular.* URL: https://angular.io/.

[26] Evan You. *Página oficial de Vue.js.* URL: https://vuejs.org/.

[27] Android Studio FAQs. *¿Qué es WebView?.* URL: https://androidstudiofaqs.com/conceptos/android-system-webview.

[28] Microsoft. *¿Qué es Xamarin?.* URL: https://docs.microsoft.com/es-es/xamarin/get-started/what-is-xamarin.

[29] Xamarin Help. *How Xamarin AOT works?.* URL: https://www.xamarinhelp.com/xamarin-android-aot-works/.

[32] Dart Code. *Documentación oficial de Dart.* URL: https://dart.dev/guides.

[33] GitHub. *Página oficial de GitHub.* URL: https://github.com/.

[34] Microsoft. *Documentación oficial de Visual Studio Code.* URL: https://code.visualstudio.com/docs.

[35] ikhoshabi.com. *Documentación oficial de fl_chart.* URL: https://pub.dev/packages/fl_chart.

[36] Comunidad. *Documentación oficial de k_chart.* URL: https://pub.dev/packages/k_chart.

[37] Google. *Documentación oficial de flutter charts.* URL: https://pub.dev/packages/charts_flutter.

UNIVERSIDAD DE ALMERÍA

Resumen/Abstract

En este Trabajo de Fin de Grado se ha hecho uso de tecnologías como Flutter, además de la API del Exchange de criptomonedas Coinbase Pro, para poder tener un visualizador de los mercados que nos proporciona la plataforma, ayudando a tener una visión general del progreso que se realiza en esta.

Palabras clave: Flutter, Coinbase, Coinbase Pro, Dart, Criptomonedas, Aplicación Móvil.

In this End-of-Degree Project, technologies such as Flutter have been used, in addition to the cryptocurrency exchange Coinbase Pro API, to be able to have a visualizer of the stock markets that this platform provides , helping to have an overview of the progress done.

Keywords: Flutter, Coinbase, Coinbase Pro, Dart, Cryptocurrencies, Mobile App.