Universitat Oberta de Catalunya

uoc.edu

Criptomonedas.app

Alejandro Daniel Glozman
Máster universitario de Desarrollo de aplicaciones para dispositivos móviles
Trabajo Final de Máster DADM

Eduard Martin Lineros
Carles Garrigues Olivella

27/12/2022

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Criptomonedas.app*
Nombre del autor:	*Alejandro Daniel Glozman*
Nombre del consultor/a:	*Eduard Martín Lineros*
Nombre del PRA:	*Carles Garrigues Olivella*
Fecha de entrega (mm/aaaa):	12/2022
Titulación:	*Máster universitario de Desarrollo de aplicaciones para dispositivos móviles*
Área del Trabajo Final:	*Trabajo Final de Master DADM*
Idioma del trabajo:	*Castellano*
Palabras clave	*Flutter, Firebase, Criptomonedas*

Resumen del Trabajo (máximo 250 palabras): *Con la finalidad, contexto de aplicación, metodología, resultados i conclusiones del trabajo.*

El objetivo de este trabajo es la realización de una aplicación donde los usuarios puedan estar informados constantemente sobre el mercado de las criptomonedas. Además, que los usuarios puedan interactuar con sus cuentas de un exchange de manera de tener la información sobre sus activos, poder consultar y administrar sus ordenes y hasta poder programar el alta de ordenes nuevas.

Para lo cual he decidido utilizar una metodología de desarrollo rápida (RAI) ya que dada la envergadura del proyecto y los tiempos pautados considero que es la mas adecuada. La idea es contar con por lo menos un producto mínimo viable y luego seguir hasta donde se pueda avanzar para llegar con la mejor aplicación posible en los tiempos marcados por el calendario académico.

El resultado será una aplicación amigable, fácil de comprender (dentro de lo complejo que es el mercado de las criptomonedas) donde el usuario pueda estar actualizado con los precios de las 5 mil criptomonedas con mayor capitalización del mercado, pueda tener criptomonedas favoritas para un rápido acceso, pueda hacer conversiones entre criptomonedas y entre criptomonedas y monedas Fiat, pueda visualizar los precios en euros o dólares estadounidenses, pueda dar de alta alertas de precios y pueda gestionar su cuenta de por lo menos un exchange.

Abstract (in English, 250 words or less):

The objective of this work is to create an application where users can be constantly informed about the cryptocurrency market. In addition, users can interact with their exchange accounts to have information about their assets, to be able to consult and manage their orders and even to be able to schedule the registration of new orders.

For which I have decided to use a rapid development methodology (RAI) since, given the size of the project and the scheduled times, I consider it to be the most appropriate. The idea is to have at least one minimum viable product and then continue as far as progress can be made to arrive at the best possible application in the times set by the academic calendar.

The result will be a friendly application, easy to understand (within how complex the cryptocurrency market is) where the user can be updated with the prices of the 5 thousand cryptocurrencies with the highest market capitalization, they can have their favorite cryptocurrencies for a quick access, they can make conversions between cryptocurrencies and between cryptocurrencies and Fiat currencies, they can view prices in euros or US dollars, they can register price alerts and they can manage your gutter from at least one exchange.

Índice

Lista de tablas

Lista de figuras

1. Introducción

1.1 Contexto y justificación del Trabajo

Desde el nacimiento del Bitcoin en 2009, el universo de las criptomonedas ha evolucionado a pasos vertiginosos. En enero de 2021, según una estimación de la empresa crypto.com, el numero de personas que utilizaban criptomonedas ascendía a los 106 millones. Al momento de escribir estas líneas, en el sitio web coinmarketcap.com hay listados 411 exchanges y más de 12 mil monedas digitales que, entre todas, poseen una capitalización que supera al trillón y medio de euros (€1.642.574.880.659).

No obstante, cada día se suman más y más usuarios deseando invertir, realizar trading o interactuar con criptoactivos. Según un artículo de elconfidencialdigital.com [1] el 38 por ciento de los jóvenes españoles quiere invertir en ellas. Pero muchos de estos usuarios cuentan con pocos conocimientos informáticos y/o financieros, no saben dónde obtener información confiable y la mayoría de exchanges son difíciles de entender hasta para los usuarios avanzados.

Los exchanges suelen brindar mucha información de manera poco amigable o no la proporcionan o la proporcionan por tiempo limitado, como el historial de transacciones de cada usuario. Además, las herramientas para operar que brindan son complicadas para usuarios sin experiencia en mercados bursátiles y suelen estar más orientadas al propio negocio del exchange que en los usuarios.

El siguiente Trabajo final de master (TFM), del Máster universitario de Desarrollo de aplicaciones para dispositivos móviles de la Universitat Oberta de Catalunya, propone una aplicación móvil donde poder consultar la información básica como precio, variación y capitalización de las criptomonedas de una manera rápida y sencilla y al mismo tiempo brindar a los usuarios la posibilidad de conectarse con sus cuentas de un exchange, para poder interactuar con el mismo, de una mejor manera que si lo hicieran directamente.

Se considera el TFM a presentar como el mínimo producto viable, la primera versión gratuita, de una aplicación completa para ser subida en diferentes App Store, pero que luego, en futuras actualizaciones, se le podría ir agregando más funcionalidades como también más exchanges con los cuales los usuarios podrían operar.

1.2 Objetivos del Trabajo

El objetivo de este TFM es que los usuarios del mercado de criptomonedas, tanto nuevos como avanzados, puedan tener información que los ayude a tomar mejores decisiones y que cuenten con herramientas que les permitan ejecutar dichas decisiones.

En este sentido, el trabajo se propone desarrollar una primera versión de una aplicación móvil que permita, por un lado, visualizar información genérica del mercado de las criptomonedas de manera amigable y, por otro, brindar a usuarios identificados la posibilidad de interactuar con su *exchange* de una mejor manera a que si lo hiciera directamente.

1.2.1 Definición de los requisitos de la aplicación

En cuanto a los requerimientos funcionales he decidido dividirlos entre importantes y deseables. Los requerimientos funcionales importantes son los mínimos indispensables que contendrá la aplicación del TFM, ya que al desarrollar una aplicación con dichos requerimientos se considera que ya se cuenta con una aplicación completa mínima que refleja el espíritu del producto al que se quisiera llegar con futuras versiones. Los requerimientos funcionales deseables son requerimientos a los cuales se desea llegar en una primera o posterior versión (si hay tiempo suficiente se desarrollarán para el TFM y sino se considerarán como mejoras a realizar en un futuro). No obstante, podría darse el caso en que, por motivos del tiempo escaso y las tecnologías que involucra este trabajo, en etapa de desarrollo puedan intercambiarse entre los requerimientos importantes y los deseables, siempre con la intensión que al final quede una aplicación que pueda percibirse como finalizada en su primera versión. Esto coincide con la metodología de desarrollo rápida que se he seleccionado para la realización de este trabajo como explicaré más adelante.

Requerimientos funcionales importantes

Tabla 1: Requerimientos funcionales importantes

Código	Requerimiento
RFi_001	Que se puedan ver las criptomonedas que hay en el mercado con sus datos como precio, variación, capitalización
RFi_002	Que se puedan filtrar y ordenar las criptomonedas por diferentes criterios
RFi_003	Que se pueda seleccionar las criptomonedas favoritas para una rápida visualización
RFi_004	Que se pueda consultar la cuenta del usuario de un exchange añadiendo y mejorando los informes existentes en dicho exchange si es que fuera posible

RFi_005	Poder operar con el exchange seleccionado pudiendo dar de alta, manejar y consultar ordenes de compra
RFi_006	Poder programar en el exchange seleccionado ordenes de compra que dependan de la ejecución de otras ordenes
RFi_007	Poder eliminar ordenes de compra programadas
RFi_008	El usuario podrá siempre eliminar o modificar del back-end de criptomonedas.app los datos de autentificación de su Exchange

Requerimientos funcionales deseables

Tabla 2: Requerimientos funcionales deseables

Código	Requerimiento
RFd_001	Guardar el historial de transacciones del usuario en su exchange por si el exchange restringe el acceso o elimina los registros
RFd_002	Los usuarios puedan recibir alertas cada cierto tiempo sobre el precio u otro atributo de alguna criptomoneda
RFd_003	Los usuarios puedan recibir alertas cuando el precio u otro atributo de una criptomoneda supera un valor x
RFd_004	Los usuarios puedan compartir la aplicación con amigos o conocidos
RFd_005	Los usuarios anónimos también pueden configurar y recibir alertas
RFd_006	Se podrá visualizar gráficas con la variación de los precios en un determinado periodo de tiempo de las criptomonedas

RFd_007	El usuario podrá crear una cartera ficticia de criptomonedas para ver y analizar su evolución
RFd_008	Las configuraciones que hagan los usuarios autentificados deben persistir y mantenerse en todos los dispositivos en que el usuario se identifique

Requerimientos no funcionales

Código	Requerimiento
RnF_001	La aplicación debe ser intuitiva y en general debe poderse utilizar sin leer instrucciones
RnF_002	La aplicación debe proporcionar mensajes que sean informativos y orientados al usuario final cuando sea necesario
RnF_003	La aplicación debe funcionar cómo mínimo en un teléfono móvil con sistema operativo Android
RnF_004	La aplicación debe están en idioma español
RnF_005	La aplicación se desarrollará en Flutter
RnF_006	La aplicación usará Firebase para bases de datos y autentificación
RnF_007	Que el usuario tenga una buena experiencia de uso incluso cuando no posea conexión a internet
RnF_008	La aplicación debe usarse en modo de orientación portrait cómo mínimo

1.3 Enfoque y método elegido

1.3.1 Enfoque

Se ha decidido realizar la aplicación en dos grandes partes, una donde no se requiere autentificación y la otra donde si se requiere. Todo lo que corresponda con datos público y genéricos de criptomonedas se podrá consultar y personalizar sin necesidad de autentificarse. Para utilizar la sección donde el usuario puede interactuar con un *exchange* deberá estar identificado en la aplicación y proporcionar además la información de autentificación que el exchange solicite para la utilización se su API. Además, se contempla la posibilidad de ofrecer servicio de alertas tanto para usuarios autentificados como para usuarios anónimos.

La idea es que los usuarios deban autentificarse o dar permisos solo cuando ellos deseen hacerlo para usar algún servicio que de otra manera no podría brindarse. He considerado que el mundo de las criptomonedas se ha creado con el espíritu de libertad y anonimato y me ha parecido bien respetar este espíritu. Además, cada vez más usuarios sospechan de las aplicaciones que piden permisos o datos innecesarios y mucho mas en aplicaciones que involucran a las criptomonedas o dinero de por medio.

Para la sección pública y para obtener el precio actual de las criptomonedas se ha optado por usar los datos proporcionados por el sitio web coinmarkercap.com el cual mediante API provee el promedio de los precios actuales de las criptomonedas en los distintos *exchange* existentes. La API de coinmarkercap.com es gratuita para un número reducido de consultas y sobrepasadas las mismas se vuelve de pago, por este motivo se ha decidido no consultar desde la aplicación directamente a la API de coinmarkercap.com, sino que consultará a un servidor web propio el cual a su vez este hará las consultas a la API de coinmarketcap.com. De esta manera no se hará una petición a la API de coinmarketcap.com cada vez que el usuario use la aplicación, o a cada segundo, con la finalidad de evitar pasar los límites de la versión gratuita de dicha API.

Para la sección privada se ha decidido utilizar el exchange cex.io ya que la información que posee se puede mejorar bastante en cuanto a estructuración y generación de informes. Es un exchange bastante intuitivo y fácil de utilizar pero que, por ejemplo, no permite ver en una divisa particular cuanto suma el total de activos que los usuarios poseen.

En cuanto a la funcionalidad de poder programar ordenes que dependan de la ejecución de otras ordenes se han contemplado dos opciones de realización:

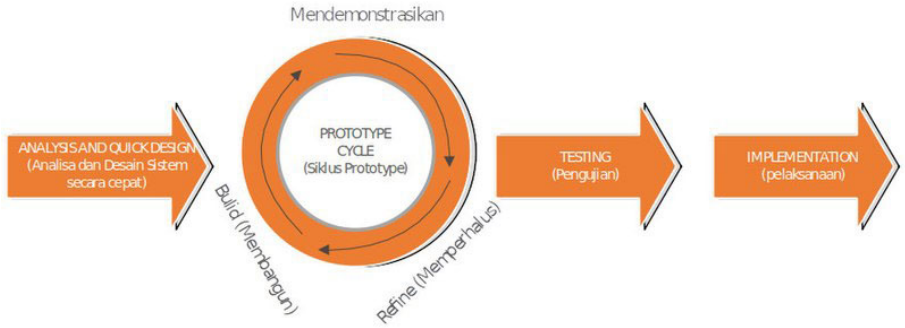
1. El usuario puede seleccionar entre las ordenes puestas en el exchange y luego puede programar en una orden para ser dada de alta en el exchange una vez que sea ejecutada la primera orden.

2. El usuario programa una cola de ordenes con prioridades que se ejecutarán apenas el back-end de criptomonedas.app detecte que se cuenta con los fondos suficientes como para poder dar de alta dicha orden.

Se ha optado por la segunda opción por considerarla más fácil de transmitir a los usuarios nuevos y porque requiere menor dificultad y carga horaria, lo que la convierte en más realista de concretar teniendo en cuenta los plazos del TFM. La primera opción se deja como una funcionalidad a agregar en una posterior actualización de la aplicación.

Para el monitoreo de la cuenta y poder programar ordenes se realizará mediante un script en un servidor propio externo que se ejecutará cada x tiempo. En cada ejecución se verificará si se han dado las condiciones necesarias para proceder al alta de la nueva orden. Si las condiciones necesarias están dadas, el programa dará de alta la orden programada. Se ha contemplado desarrollar esta funcionalidad para que sea ejecutada por la misma aplicación móvil en segundo plano, pero se ha descartado por el echo de que consumiría demasiada batería y ocupar los recursos escasos de los dispositivos móviles como son la CPU y la menoría RAM.

1.3.4 Método de desarrollo



https://commons.wikimedia.org/wiki/File:RAD_Modif.jpg

He optado por un método de desarrollo rápido de aplicaciones (RAD). Es un método que se suele usar mucho en el desarrollo de aplicaciones móviles ya que el mercado es muy cambiante y la competencia abundante. Es un método iterativo dónde la prioridad se enfoca en conseguir primeras versiones rápidas para poder mejorarlas después, a medida que se va pudiendo. En este método se suele priorizar la implementación a la planificación.

Este método va muy bien con mi TFM por el hecho que considero que tengo muy poco tiempo para el desarrollo que me he propuesto hacer teniendo

en cuenta los conocimientos a priori que poseo de algunas de las tecnologías que me he propuesto utilizar.

Como algunas de estas tecnologías nunca las he usado, es que algunas cuestiones relacionadas a ellas que pueda planificar pueden ser erróneas o menos indicadas que otras soluciones. Es por es motivo que realizaré todas las planificaciones y pasos exigidos para la realización del TFM desde los conocimientos que tengo antes de comenzarlo, pero luego, en la etapa de implementación, puede ser que quizá tenga que sacrificar algunas de las decisiones tomas a favor de culminar con un producto mínimo viable tal como indica la metodología RAD.

1.4 Planificación del Trabajo

Para realizar la planificación del proyecto se ha tomado al trabajo final de máster como el proyecto a realizar. Por este motivo como hitos se han tomado las fechas de los diferentes entregables, la entrega final e incluso la defensa del TFM donde se le responden al tribunal evaluador todas aquellas dudas que puedan tener y a las cuales hay que responderlas en un plazo máximo de 24 horas.

Diagrama de hitos

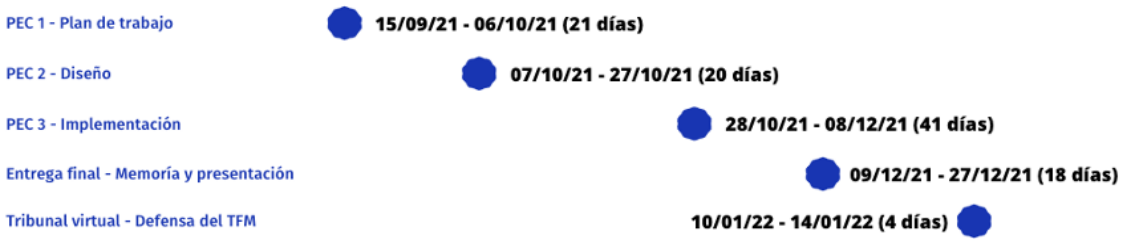
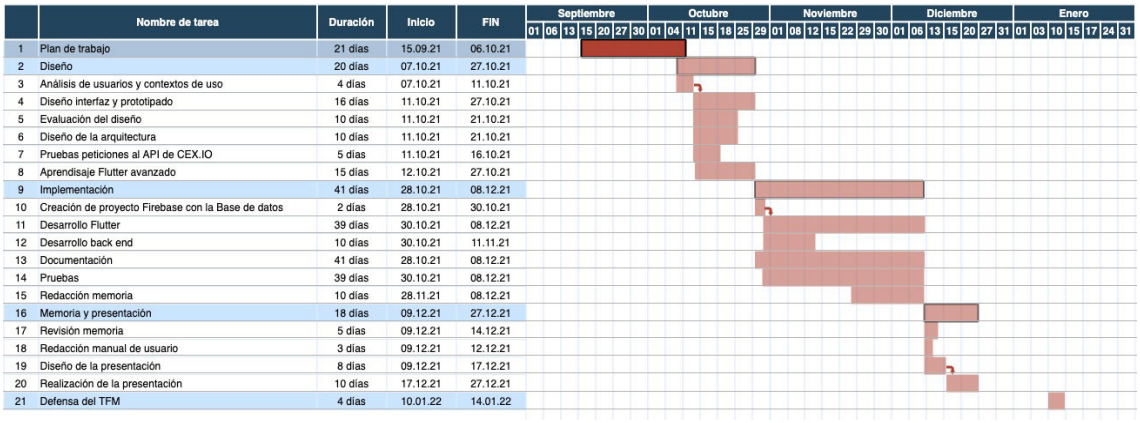


Figura 1: Diagrama de hitos

Para la planificación de horas de trabajo asignadas a tareas se ha planificado una dedicación de 4 horas diarias. Se trabajará de lunes a lunes, por lo que, desde el 6 de octubre de 2021, cuando se está finalizando de escribir este párrafo, restan un total de 81 días o, expresado en otra forma, 324 horas hasta el 27 de diciembre de 2021, día en que se realizará la entrega final del TFM.

Diagrama de Gantt



	Nombre de tarea	Duración	Inicio	FIN
1	Plan de trabajo	21 días	15.09.21	06.10.21
2	Diseño	20 días	07.10.21	27.10.21
3	Análisis de usuarios y contextos de uso	4 días	07.10.21	11.10.21
4	Diseño interfaz y prototipado	16 días	11.10.21	27.10.21
5	Evaluación del diseño	10 días	11.10.21	21.10.21
6	Diseño de la arquitectura	10 días	11.10.21	21.10.21
7	Pruebas peticiones al API de CEX.IO	5 días	11.10.21	16.10.21
8	Aprendisaje Flutter avanzado	15 días	12.10.21	27.10.21
9	Implementación	41 días	28.10.21	08.12.21
10	Creación de proyecto Firebase con la Base de datos	2 días	28.10.21	30.10.21
11	Desarrollo Flutter	39 días	30.10.21	08.12.21
12	Desarrollo back end	10 días	30.10.21	11.11.21
13	Documentación	41 días	28.10.21	08.12.21
14	Pruebas	39 días	30.10.21	08.12.21
15	Redacción memoria	10 días	28.11.21	08.12.21
16	Memoria y presentación	18 días	09.12.21	27.12.21
17	Revisión memoria	5 días	09.12.21	14.12.21
18	Redacción manual de usuario	3 días	09.12.21	12.12.21
19	Diseño de la presentación	8 días	09.12.21	17.12.21
20	Realización de la presentación	10 días	17.12.21	27.12.21
21	Defensa del TFM	4 días	10.01.22	14.01.22

Figura 2. Diagrama de Gant

1.4.1 Riesgos y medidas de contingencia

El proyecto de por si es bastante ambicioso y el tiempo muy escaso, por lo que me he visto obligado a acotar a una mínima versión viable. Pero aún de esta manera sigue siendo ambicioso y tiene bastantes riesgos (Es un proyecto ambicioso teniendo en cuenta que lo realiza una sola persona y necesita aprender varias de las tecnologías involucradas en el desarrollo, como Flutter avanzado y la integración con las funcionalidades avanzadas de Firebase).

No obstante, los riesgos pueden dividirse en dos grandes ítems: el diseño y problemas de desarrollo.

En cuanto a diseño el principal riesgo es que no soy diseñador y además tampoco soy un experto maquetando con Flutter. Aunque pueda realizar un prototipo de alta fidelidad aceptable, no se si luego tengo la habilidad en Flutter como para lograr respetarlo fielmente. Para el riesgo de diseño la medida de contingencia será respetar las funcionalidades y la estructura pensada, pero el diseño se acomodaría a lo que pueda salir y reducir la carga horaria en la etapa de desarrollo.

En cuando a los problemas de desarrollo tienen su raíz en que estoy trabajando con tecnologías que son bastante nuevas para mi, muchas de ellas nunca las he utilizado. El principal problema lo veo en comunicar la base de datos Firestore con el *back-end* para dar de alta las ordenes y el posterior envío de notificaciones push. No se bien ahora mismo que mecanismos de seguridad tenga Firestore y Firebase en cunado a dominios, ip's y cuanto tiempo me puada insumir el aprendizaje de configuración e permisos. Por este motivo se me ocurren tres medidas de contingencia:

1. En caso de no poder acceder desde un servidor externo, intentar implementarlo con Firebase funciones, pero tampoco nunca lo he usado y sigue teniendo el risco de cuan prolongada es la curva de aprendizaje.

2. Dejar solo para la presentación del trabajo final la base de datos de Firestore sin restricciones de acceso, libre para poder acceder desde

cualquier lado. Pero aún así no se cuan posible es mandar las notificaciones push desde un PHP externo y cual es la curva de aprendizaje para lograrlo.

3. No incluir esta funcionalidad en la primera versión y remplazarla con el requerimiento funcional deseado de las alertas. En este último caso considero que la aplicación sigue siendo bastante completa como para considerarla una mínima primera versión viable. Si bien es un plan de contingencia un poco más drástico, lo considero aceptable a esta altura por el tiempo con el que cuento, las tecnologías nuevas que me he propuesto incluir y porque después de todo la aplicación seguiría respetando el concepto de mínimo producto viable para una primera versión.

1.5 Breve sumario de productos obtenidos

Se obtendrán los siguientes productos:

- La aplicación en formato apk para poder ser instalada en un dispositivo móvil con sistema operativo Android
- Una demo del funcionamiento de la herramienta en formato vídeo
- Una memoria que documente, de forma estructurada y auto contenida, el planteamiento, el desarrollo y los resultados del trabajo
- Una presentación que comunique los aspectos más relevantes del trabajo de forma sintética y atractiva

1.6 Breve descripción de los otros capítulos de la memoria

- Diseño:
  Descripción de los usuarios, casos de uso, prototipado, Diagramas UML, diseño de base de datos.
- Implementación:
  Explicación y justificación de la implementación de la aplicación.
- Valoración económica del trabajo:
  Gastos asociados a la implementación y mantenimiento, así como los beneficios económicos que se pudieran obtener en caso de lanzar la aplicación al mercado, tanto en su versión gratuita como en una de pago.
- Conclusiones:
  Aprendizajes que ha dejado el TFM, objetivos alcanzados. objetivos no alcanzados si los hubiere y perspectivas o ampliaciones futuras.

- Glosario
- Bibliografía
- Anexos:
    - Un manual de usuario donde se explicará el uso de la aplicación y explicarán, cuando sea necesario, de dónde se toman los datos que brinda (por ejemplo, que el pecio de las criptomonedas es el promedio de varios *exchanges*).
    - Manual de instalación

2. Diseño Centrado en el Usuario (DCU)

2.1 Investigación y requisitos de usuario

2.1.1 Ficha/s de persona y escenario

En esta sección se mostrarán tres ejemplos representativos del resultado de entrevistas y análisis cualitativo. Las imágenes y nombres son ficticios, pero representan fielmente el espíritu de la información recolectada. Se ha decidido no incluir nombres ni imágenes reales ya que pedir permiso expresamente puede causar incomodidad a las amables personas que nos han dado si tiempo y opinión. Pero además el hecho de incluir las fotos y datos personales reales o ficticios generan exactamente el mismo resultado (representar fielmente la información proporcionada para lograr un mejor producto).

No obstante, es importante que las fichas parezcan reales ya que, en caso de ser necesario presentarlas ante un equipo de desarrollo o clientes, deben inspirar veracidad absoluta y profesionalidad. Poe este motivo, las imágenes se han tomado del sitio thispersondoesnotexist.com [2] y los datos personales de fakenamegenerator.com [3], ambos usan inteligencia artificial para crear la fantasía.

Tabla 3: Ficha de Persona 1

	**Juanjo Ornelas Sanabria** **Teléfono:** 790 635 779 **Fecha de nacimiento:** 16/05/1995 **Edad:** 26 **Correo electrónico:** JuanjoOrnelasSanabria@gmail.com **Profesión:** Diseñador gráfico **Nivel de estudios:** Ciclos Formativos de Grado Superior
Descripción	
Juanjo es Diseñador gráfico, soltero y vive en Sant Antoni, Barcelona. Le gusta mucho la tecnología y siempre estar al tanto de todo lo nuevo que puede ayudarlo a ser un mejor profesional. Le interesa el mundo de las criptomonedas porque ha leído mucho y le han hablado mucho sobre el tema. Cree que puede serle útil como inversión o incluso por su trabajo. Ha comprado ya un poco y las mantiene en un exchange que le han recomendado. Si embargo, le interesa poder tener una aplicación que lo mantenga al tanto sobre los precios y su cartera de criptoactivos.	
Descripción de un escenario	
Juanjo no quiere entrar todos los días o cada cierto tiempo a su exchange para ver como ha evolucionado si cartera de criptoactivos. Desea tener una aplicación donde poder abrirla en el lugar y momento que le plazca y ver tanto los datos del mercado actualizados como la variación de su capital respecto al euro. No importa si es en la oficina, en un bar o reunión familiar,	

quiere poder verificar la información cuando le plazca de manera fácil e inmediata.

Tabla 4: Ficha de Persona 2

Débora Bonilla Riojas
Teléfono: 751 196 973
Fecha de nacimiento: 25/20/1996
Edad: 25
Correo electrónico: DeboraBonillaRiojas@hotmail.es
Profesión: Venta de insumos gastronómicos
Nivel de estudios: Ciclos Formativos de Grado Medio

Descripción

Débora trabaja 8 horas diarias de lunes a viernes, estudia inglés y le gusta reunirse con amigos los fines de semana. En unas de esas tantas salidas con amigos fue como se ha enterado de las criptomonedas. Ha dado sus primeros pasos, pero encuentra que los Exchange son complicados de entender y además nunca encuentra la información que necesita o le molesta que el exchange en el que tiene sus criptomonedas solamente le guarde los informes por tres meses. “Me parece increíble que si quiero ver a cuanto he comprado una criptomoneda hace 5 meses ya no pueda acceder a esa información”, lamenta. Quiere seguir en el exchange que viene usando porque es el que le han recomendado amigos a los que le tiene confianza en esta materia, pero le gustaría que mejore mucho y que le guarde toda la información histórica.

Descripción de un escenario

Domingo a la tarde esta en su casa tranquila, descansando y preparando todo para empezar la semana de buena manera. Mira el precio de su criptomoneda favorita que había comprado hace cinco meses. Entra a su exchange y le informa que solo guarda las operaciones por tres meses. Por suerte recuerda que tiene criptomonedas.app y tranquila desde su sofá, sin abrir el ordenador consulta las operaciones que ha realizado desde que ha creado su usuario dado los permisos necesarios para que criptomonedas.app le guarde dicha información.

Tabla 5: Ficha de Persona 2

Jordana Benítez Vera
Teléfono: 797 625 157
Fecha de nacimiento: 30/06/1982
Edad: 38
Correo electrónico: JordanaBenitezVera@gustr.com
Profesión: Ejecutiva de cuentas en un banco
Nivel de estudios: Ciclos Formativos de Grado Medio

Descripción

Jordana, 47 años, trabaja en las oficinas de un banco de 9 a 18 horas. Al terminar su jornada laboral se va a su departamento de Les Corts, Barcelona, dónde vive con su hija de 20 años. Es una usuaria de criptomonedas medianamente avanzada, se introdujo hace dos o tres años por medio de un cliente del banco. Usa dos exchanges y hardware wallets. Busca una aplicación sencilla que le permita ver los precios, le envía alertas personalizado, le guarde historial de operaciones, pero también desea automatizar las ordenes de compra. Poder programar ordenes que dependan de la ejecución de otras ordenes. Ella esta muy ocupada y le gusta jugar a comprar y vender, pero no puede estar pendiente todo el tiempo, de hecho, casi nunca. Sabe de la existencia de robots de cryptotrading pero los considera demasiado complejos para lo que ella quiere, y además no son transparentes en cuanto a las estrategias de trading que utilizan o como estas están programadas. Quiere poder programar ordenes, pero con un mayor control, siguiendo su propia estrategia.

Descripción de un escenario

Jordana tiene un día muy ocupado, como de costumbre. Se encuentra en una reunión asesorando a dos socios de una importante cafetería de la zona sobre diferentes opciones de inversión que les ofrece el banco. Siente vibrar su smartwatch, lo mira disimuladamente y suelta una pequeña e imperceptible sonrisa. Un correo electrónico de criptomonedas.app le avisaba que se ha dado de alta en el exchange la operación de compra anidada que había programado. Sigue en la reunión como si nada hubiera pasado.

2.1.2 Necesidades de los usuarios

Luego de realizada la investigación de usuarios procedo a resumir sus necesidades:

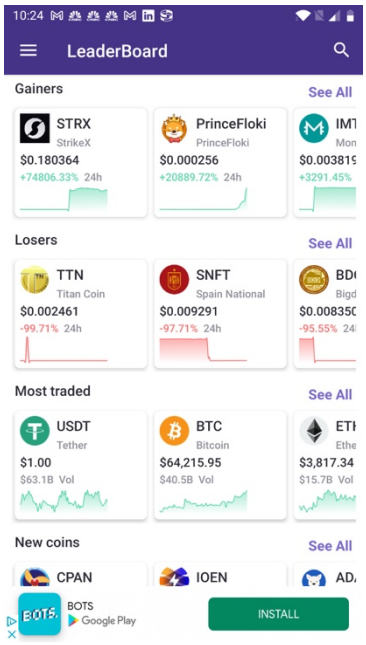
- Los usuarios requieren una aplicación sencilla que les ofrezca la información del mercado y les brinde herramientas para la toma de decisiones
- Esta aplicación debe ser lo menos intrusiva posible y con el mayor anonimato posible
- Debe ser una aplicación que les permita estar concentrados en sus tareas cotidianas sin perder de vista el mercado de las criptomonedas
- Debe permitirles seguir operando con sus exchange de confianza, pero mejorando los informes y servicios que ellos les ofrecen

2.2 Diseño conceptual

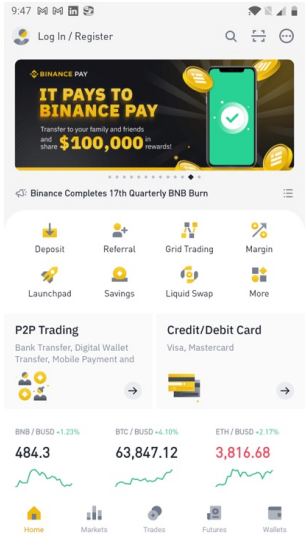
Luego de haber confeccionado la ficha de usuarios, como primer apartado del diseño conceptual he incluido un análisis de la competencia o productos sustitutos. Este análisis me ha permitido rever los requerimientos funcionales y además he podido obtener ideas para resolver soluciones de diseño en general.

2.2.1 Benchmarking en las tiendas de aplicaciones

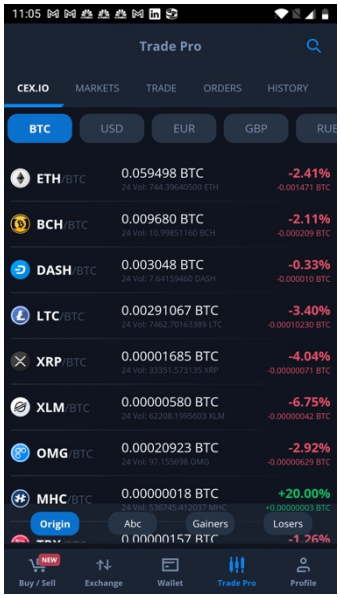
Puntos fuertes y debilidades de soluciones parecidas a criptomonedas.app

Criptomonedas. Bitcoin price. Crypto rate URL de descarga: https://play.google.com/store/apps/details?id=com.supercrypto.cryptocyrrency	
Descargas: 100.000+ Estrellas: 4,6 Reviews: 9.064 en total	
Puntos fuertes	
No es un exchange, gana con publicidad o con los que pagan para hacer desaparecer la publicidad Posee la posibilidad de crear alertas de precios Tiene sección de noticias Tiene posibilidad de ver precios en hasta dos monedas Permite crear porfolio ficticio Buen diseño, intuitivo y simple	
Debilidades:	
En cuanto a alertas solo tiene la posibilidad de precio No permite conectarse a cuentas de exchange de los usuarios y mejorar la información y servicios	

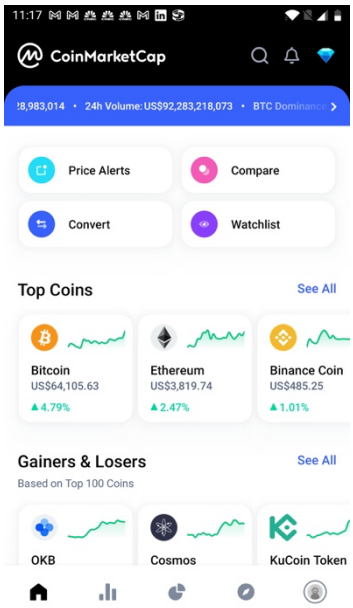
Tiene favoritos, pero a simple vista no resulta intuitivo generarlos Publicidad muy frecuente y molesta	

Binance: Compra Bitcoin & SHIB URL de descarga: https://play.google.com/store/apps/details?id=com.coinbase.android	
Descargas: 10.000.000+ Estrellas: 4,4 Reviews: 417.392 en total	
Puntos fuertes	
El exchange más popular del mundo Tienen mucho poder dentro del mundo de las criptomonedas Dueños de coinmarketcap.com Tiene sus propias criptomonedas dentro del top 10 en cuanto a capitalización y actualmente es la principal competencia de etherium en cuanto a desarrollo de contratos inteligentes. La aplicación permite comprar, vender y todas las opciones mas avanzadas dentro del criptomercado	
Debilidades:	
Aplicación muy compleja con una curva de aprendizaje de las más elevadas dentro del criptomercado Al ser el exchange más popular y tener tanto poder, puede dar la sensación de manipulación y poca transparencia Es un exchange en si mismo, en su aplicación solo promociona e incentiva sus propios productos La aplicación puede resultar agobiante	

hasta para usuarios avanzados	

CEX.IO Exchange de Bitcoin y +80 criptomonedas URL de descarga: https://play.google.com/store/apps/details?id=io.cex.app.prod	
Descargas: 1.000.000+ Estrellas: 3,9 Reviews: 14.371 en total	
Puntos fuertes	
Interesante experiencia de usuario al cambiar el criterio de ordenación de las criptomonedas. Se animan trasladándose a su nueva ubicación Es la aplicación oficial del primer exchange que criptomonedas.app va a incorporar Permite comprar criptomonedas con tarjeta de crédito fácilmente	
Debilidades:	
Es un exchange, no sirve para ver todo el mercado, solo lo que ellos ofrecen Tiene las mismas deficiencias o más que la versión web, no agrega nada más que el echo de ser una aplicación nativa No permite ordenar por capitalización del mercado No permite generar alertas No permite programar ordenes No permite ver ordenes antiguas No permite ver cuanto suman todos los activos que el usuario posee en euros, dólares o alguna criptomoneda específica	

Permite hacer transferencias, lo que en cuanto a criptomonedas.app concierne, es una debilidad porque instalar la app y autentificarse puede ser tan riesgoso como el dispositivo en que se instale sea vulnerable a ataques informáticos, robos, etc.	

CoinMarketCap - rastreador de precios criptográficos URL de descarga: https://play.google.com/store/apps/details?id=io.cex.app.prod	
Descargas: 1.000.000+ Estrellas: 4,3 Reviews: 16.801en total	
Puntos fuertes	
Es la página de referencia en el mercado de criptomonedas en cuanto a toda la información genérica del mercado Es el sistema que criptomonedas.app usará para obtener los datos del mercado para la sección libre de autentificación Permite crear portfolio	
Debilidades:	
Ya no es independiente, fue comprada por binance.com en abril de 2020 Para tener criptomonedas marcadas como favoritas pide autentificación Posee demasiada información de una manera un poco desordenada y abrumadora para ser una aplicación móvil	

Conclusión del benchmarking:

Al realizar el benchmarking hemos podido comprobar que nuestro competidor más directo es la primera aplicación analizada. Es mejorable en cuanto a algunas funcionalidades y no posee la función de mejorar a los

exchange. Me ha aportado junto a la de CoinMarketCap el requisito funcional deseado Rfd007 (que los usuarios puedan confeccionar una cartera ficticia de criptomonedas), el cual queda apuntado para desarrollar en una segunda o tercera versión. Si bien no fue un requisito que me ha aparecido en la etapa de "Investigación y requisitos de usuario", he podido ver en los *rewiews* de dichas aplicaciones que para algunas personas les resulta de utilidad.

En cuanto al diseño, me ha gustado mucho las graficas históricas de precios, lo que posiblemente no podré implementar por no contar con un histórico de precios y las API que lo ofrecen son de pago.

En general, me resultará difícil superarlos en esta primera etapa en cuanto a estética por ser una persona que no domina el arte del diseño. Cabe destacar que las aplicaciones analizadas son creadas y mantenidas por un grupo de trabajo interdisciplinar de grandes expertos y profesionales. Sin embargo, haré hasta lo imposible para lograr un producto mínimo viable que, en futuras versiones, me permita hacer de Criptomonedas.app la mejor aplicación genérica en castellano sobre criptomonedas.

2.2.3 Árbol de navegación

Es el árbol de navegación de la aplicación incluyendo las alertas que podrían dejarse para una segunda versión, pero que podrían implementarse si tengo tiempo o si debo hacer uso del plan de contingencia que las contempla.

El árbol se divide de manera clara en la parte de la aplicación que se puede usar sin autentificación y la parte restringida a usuarios autentificados.

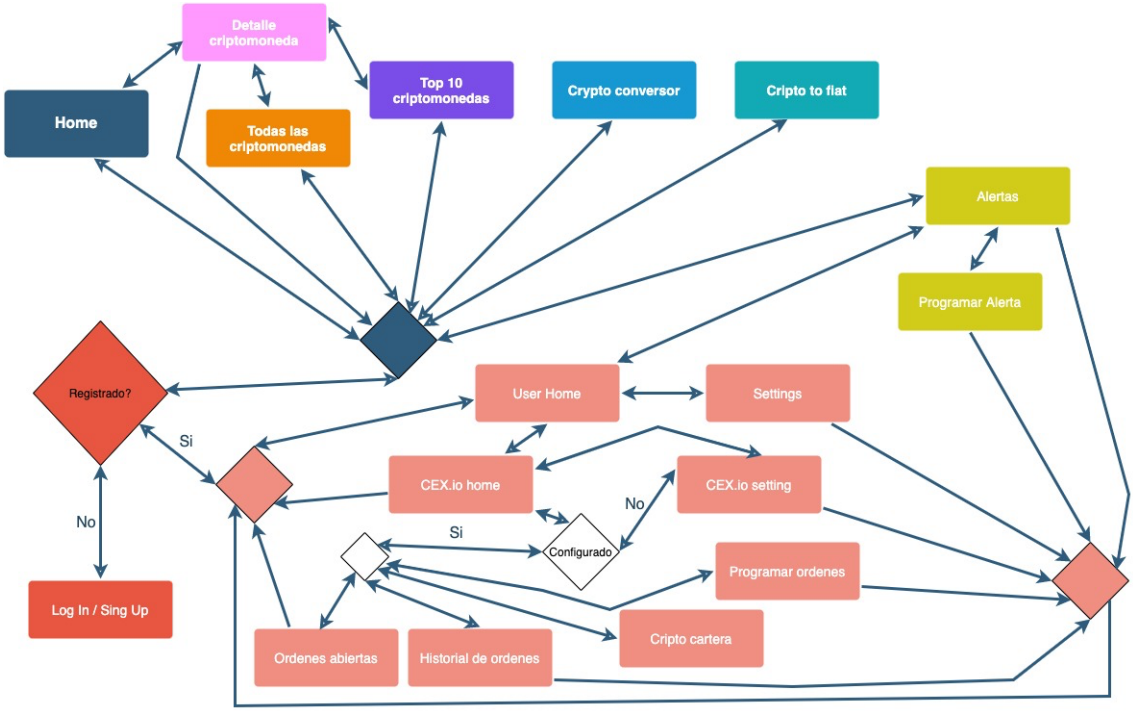


Figura 3: Arbol de navegación

La sección libre incluye al home, el listado de todas las criptomonedas, las 10 criptomonedas más importantes según diferentes criterios, un conversor de precios entre criptomonedas y un conversor entre criptomonedas y divisas. Todas estas secciones se acceden a través de un menú que se encuentra en todas las pantallas, por lo que de cualquier pantalla se puede acceder a cualquier parte de la sección libre. He tomado esta decisión por considerar que todas son herramientas indispensables para tenerlas siempre a mano. Todas las pantallas están a una pantalla de la de registro y autentificación.

En el menú también se incluirá el vinculo con alertas por dos motivos. El primero es que existe la posibilidad de configurar alertas autentificándose de manera anónima a través de Firebase y, si bien en esta etapa no sé si podré implementarlo, sería ideal que así fuera. El segundo motivo es que, aunque no pueda implementar la autentificación anónima, al estar en el menú hace que los usuarios no registrados tengan conocimiento de que existe la funcionalidad y los incentive a registrarse.

Se ha decidido que la pantalla de registro y autentificación sea una sola en vez de dos como había pensado en un primer momento. Considero que son bastante frecuentes las confusiones entre estas secciones y resulta muy molesto, mucho más que en un ordenador, tener que reescribir usuario y clave por confundirse de pantalla. Es una sola pantalla donde la diferencia entre una opción y la otra es un botón que al cambiar mantiene los daos que el usuario pudo haber ya escrito.

A la sección restringida se accede por el home del usuario, esta pantalla contendría una sumarizaciónión de la sección privada y permitirá ir a settings, la sección de alertas y a la pantalla del exchange cex.io. En futuras versiones se agregarían más exchange donde las opciones, informes y configuraciones de cada exchange serían de manera homogénea. Pero para la primera versión se ha optado por empezar con un exchange (como ya se ha mencionado y justificado en la sección anterior).

De la sección del exchange se accede al home de dicha sección y de ahí a sus diferentes secciones o a ingresar los datos de autentificación. Esto se ha pensado así ya que a pesar de que en esta primera versión habrá un solo exchange, se deja estructurado para soportar más de un exchange en posteriores actualizaciones. Considero que cambiar la estructura de una aplicación en futuras versiones no es bueno para usuarios ya fidelizados, les podría parecer molesto y por este motivo he decidido ya desde la primera versión respetar la estructura que la aplicación tendrá en el futuro. Además, esta estructuración permite que la sección privada en un futuro pueda tener secciones que hoy en día no se me han ocurrido o no existen. En caso de que surja una idea nueva para incorporar en futuras versiones fácilmente podrá incorporarse sin modificar el árbol, solo se añadirá para que se pueda acceder desde el home de la sección privada.

2.3 Prototipado

2.3.1 Sketching

Antes del prototipado de alta fidelidad, se ha realizado la técnica de *sketching*. La finalidad ha sido tener una mayor noción de la estructura que debería tener la aplicación antes de iniciar el prototipado. Junto al árbol de navegación me ayudado a ganar tiempo y tener un punto de partida en cuanto a estructura, junto a una lista de las pantallas que debía prototipar.

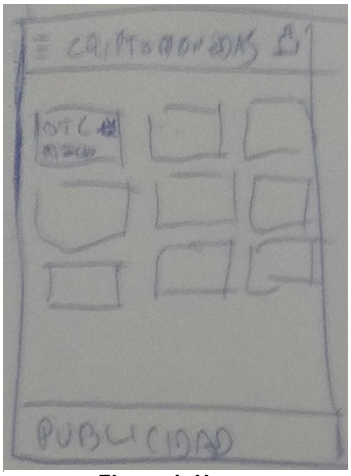


Figura 4: Home

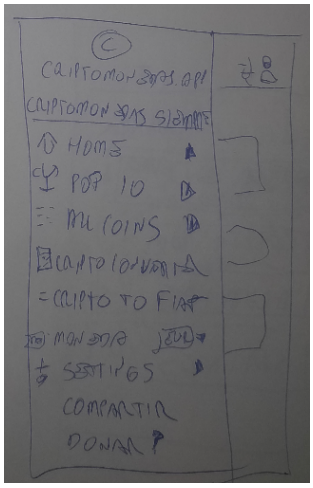


Figura 5: Menú

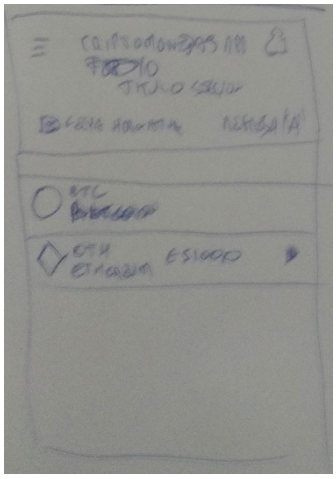


Figura 6: Lista de criptos

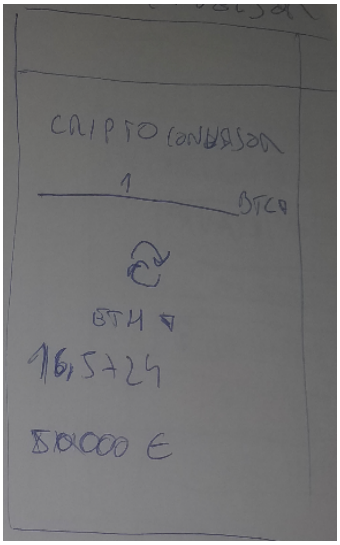


Figura 7: Criptp conversor

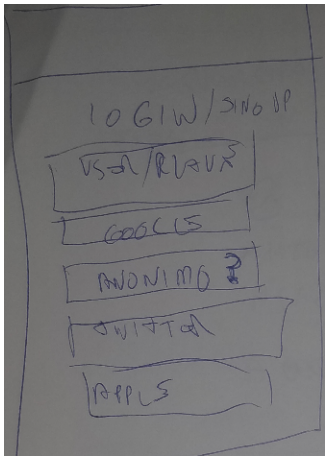


Figura 8: Login

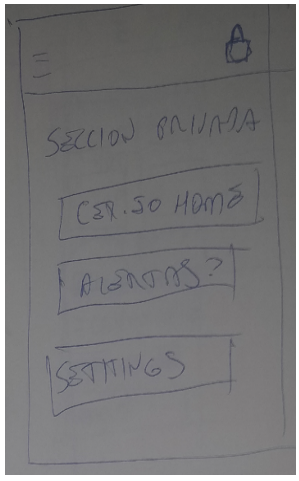


Figura 9: Sección privada

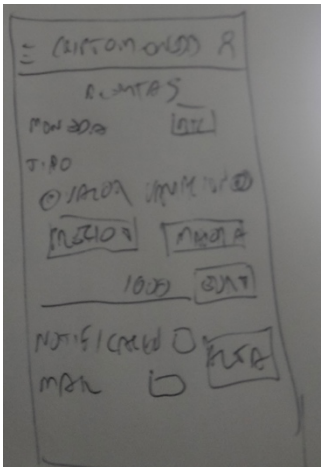


Figura 10: Alertas

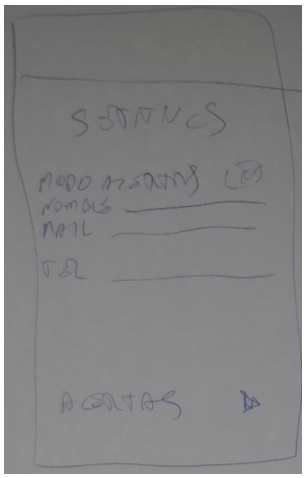


Figura 11: Settings

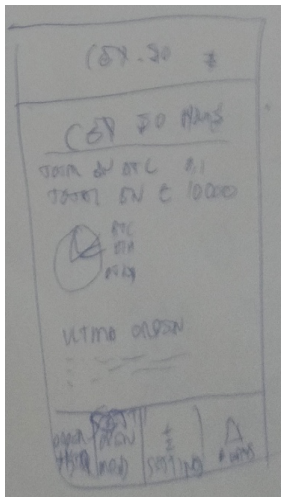


Figura 12: Exchange home

2.3.2 Prototipado

Gracias a la técnica de *sketching* he podido encarar el prototipado de alta fidelidad sin tener que enfrentarme al síndrome de la hoja en blanco. Con el prototipado de alta fidelidad no solo he podido visualizar mejor la estructura y donde colocar cada elemento, sino que he encontrado que el árbol de navegación contenía errores conceptuales que he podido corregir gracias al proceso iterativo de la metodología RAD de desarrollo.

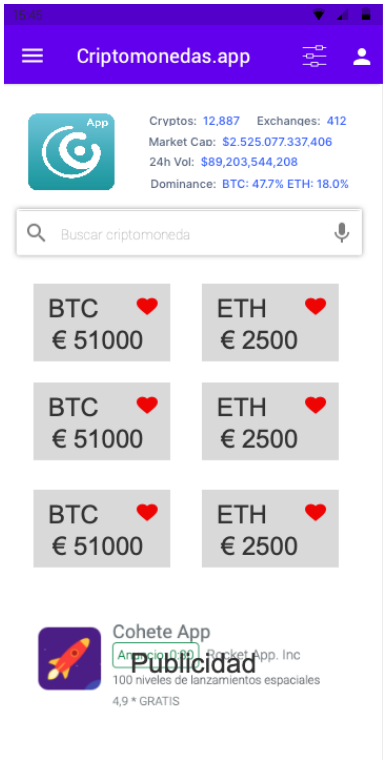


Figura 13: Home

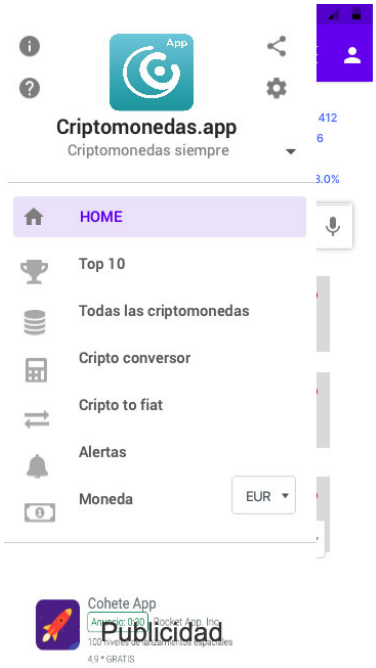


Figura 14: Menú de navegación

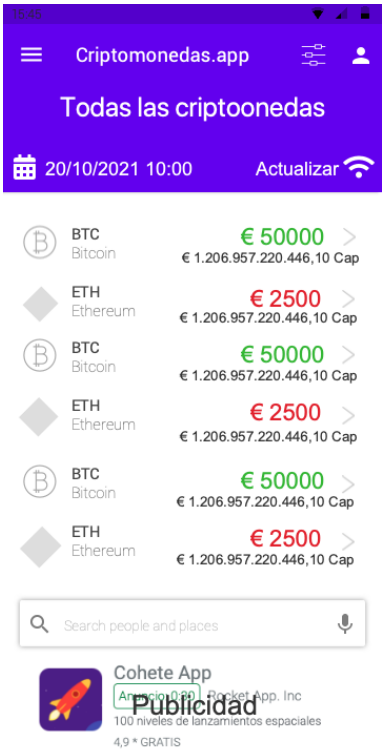


Figura 15: Todas las criptomonedas

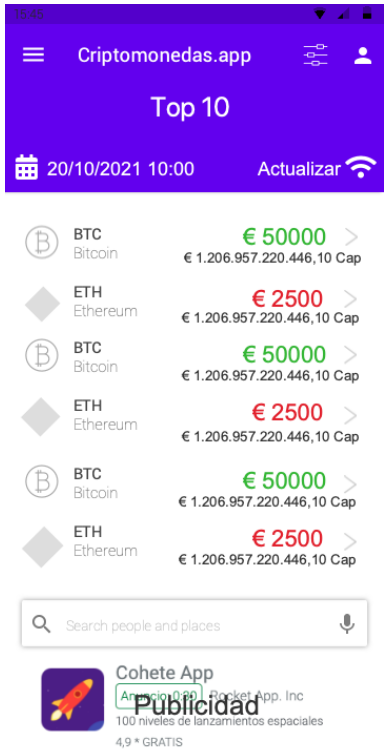


Figura 16: Top ten criptomonedas

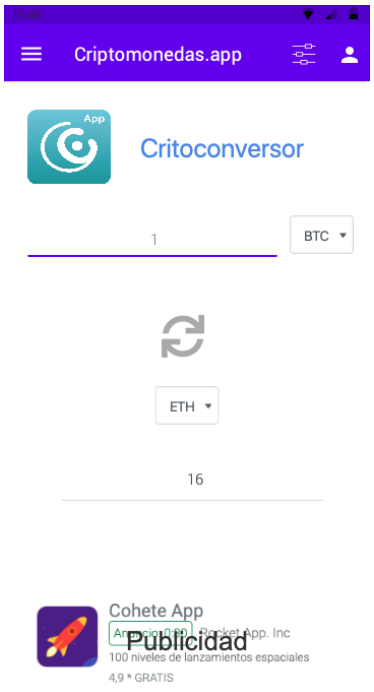


Figura 17: Criptoconversor

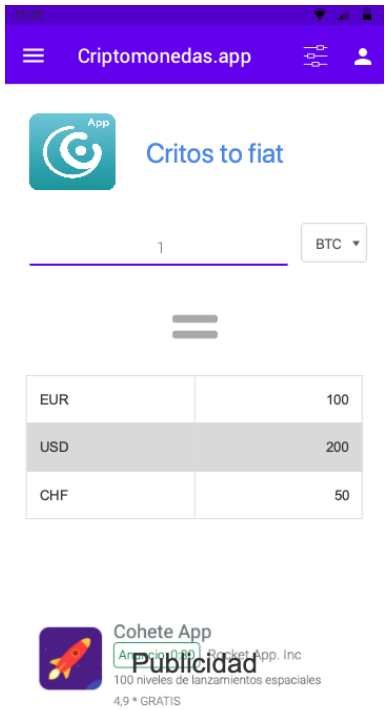


Figura 18: Convertir a fiat

Figura 19: Detalle criptomoneda

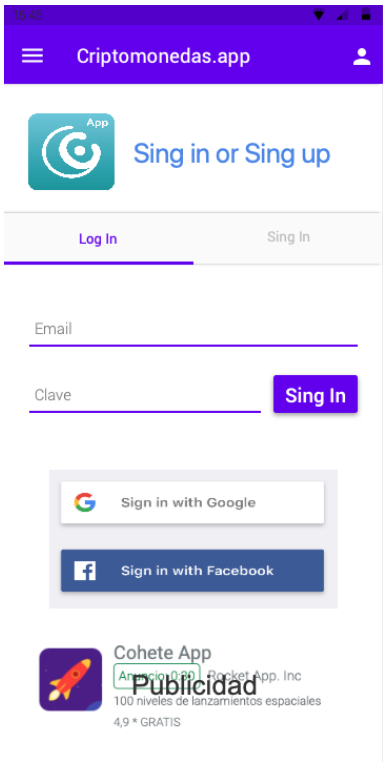


Figura 20: Sing in / Sing up

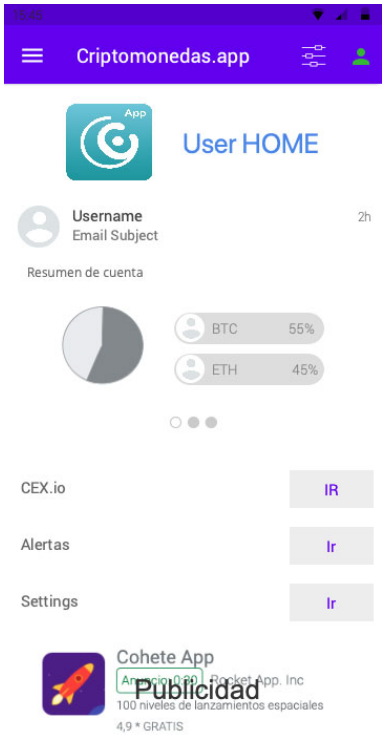


Figura 21: User home

Figura 22: Exchange home

Figura 23: Cripto cartera

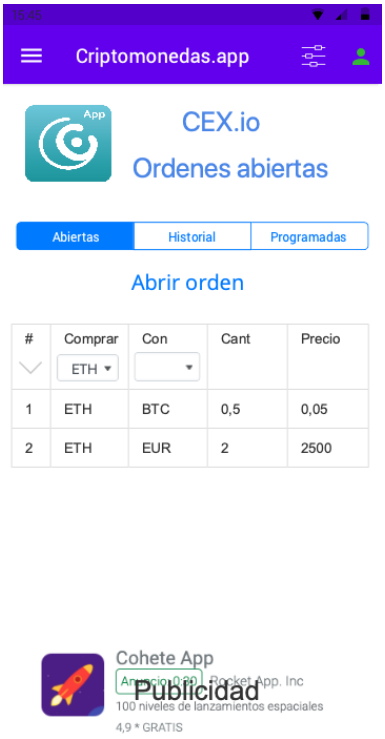


Figura 24: Ordenes abiertas

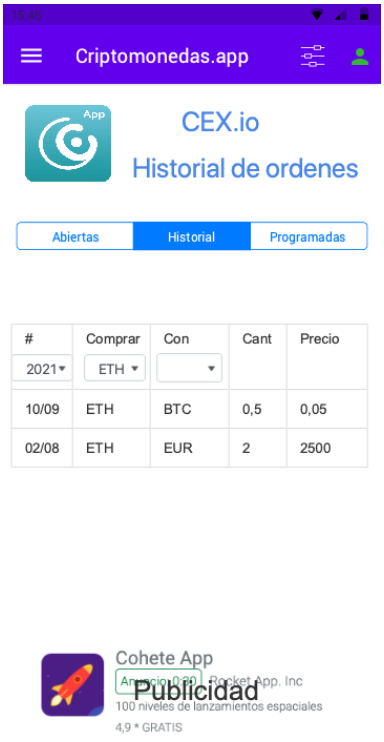


Figura 25: Historial de ordenes

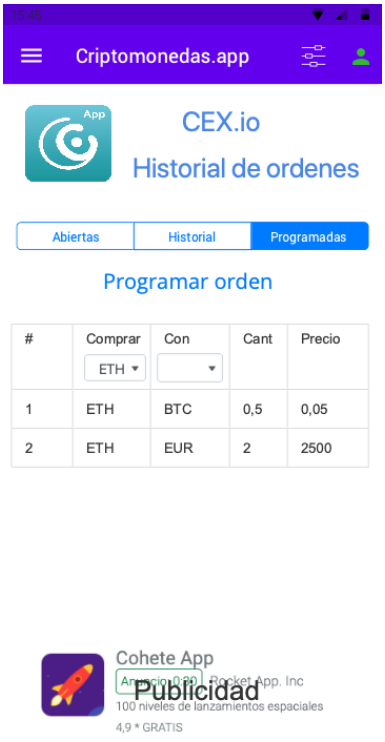


Figura 26: Ordenes programadas

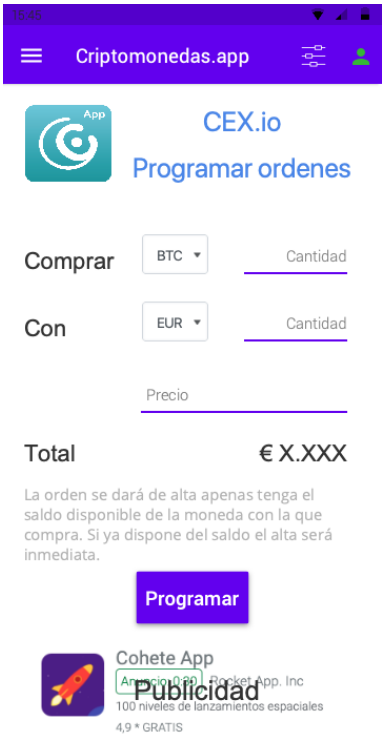


Figura 27: Programar orden

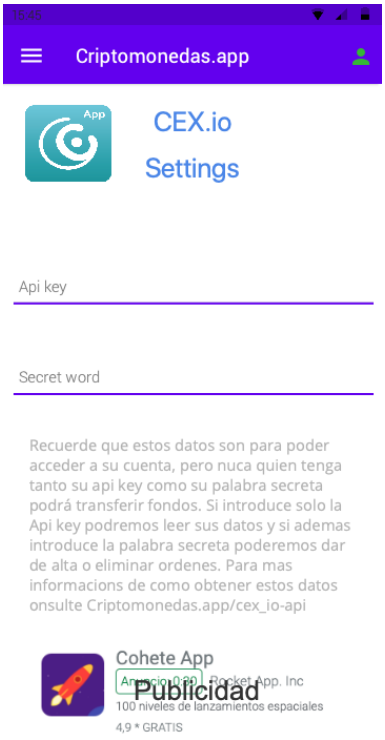


Figura 28: Exchange settings

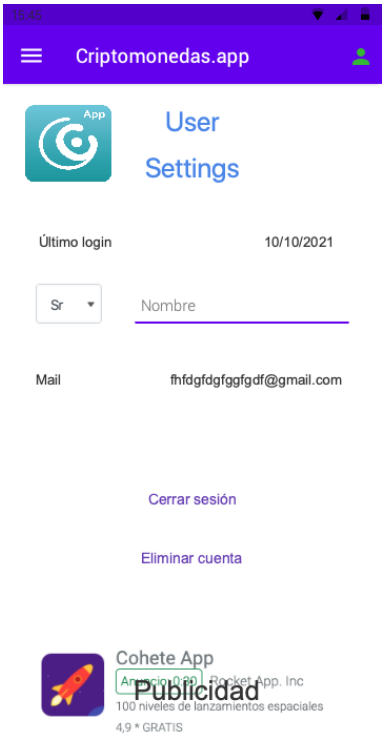


Figura 29: User settings

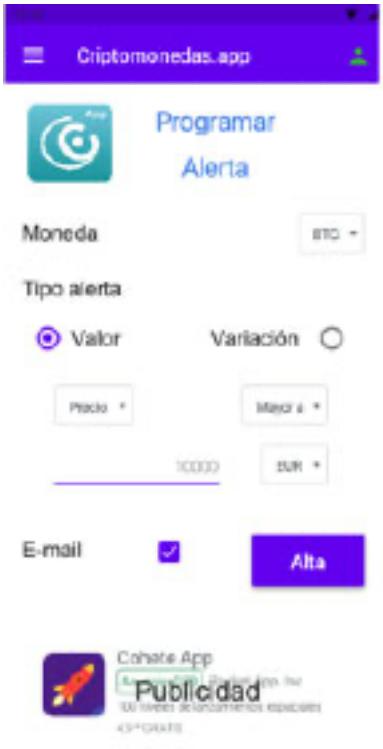


Figura 30: Programar alerta

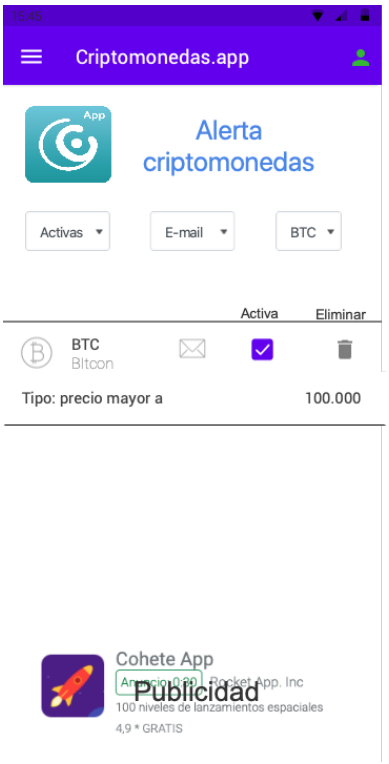


Figura 31: Criptoalertas

Haciendo el prototipado de alta fidelidad se han tomado decisiones que solo se han podido ver durante este proceso. Una de ellas fue, se explica en la sección 2.4, la ruta de navegación para llegar a las alertas. Otra ha sido la configuración de alertas, el hecho de incluir la opción para que puedan llegar por email. Antes se había pensado que se pueda seleccionar no solo el email sino también las notificaciones push, pero he pensado que las notificaciones push estén incluidas siempre en las alertas. Creo que las notificaciones push son importantes que estén siempre sobre todo por tratarse de una aplicación móvil gratuita que ganará con publicidad. Considero que las notificaciones push hacen que se use más la aplicación.

Otra de las cuestiones que puede ver en esta etapa fue que en el árbol de navegación no me había percatado de la configuración del api key del Exchange. Gracias a este proceso pude modificar el árbol para que lo contemple. Tampoco había contemplado la pantalla de criptocartera donde el usuario puede ver de una forma mas clara, que en el propio exchange, su cartera de criptomonedas.

Observaciones:

- Figura 27: Programar orden y alta de orden es la misma pantalla. se considera que la orden abierta es una orden programada que se da de alta al instante ya que el usuario cuenta con el dinero para que así suceda.

2.4 Evaluación

El árbol de navegación me ha servido para empezar el *sketching* que me ha permitido tener una mayor noción de la estructura necesaria y que me ha ayudado a corregir el propio árbol. En este punto he empezado el prototipado de alta fidelidad de manera muy dinámica y el mismo me ha hecho descubrir nuevas pantallas y circuitos de navegación que me han hecho modificar el árbol y agregar pantallas al *skeching* para luego prototiparlas (la pantalla de alta de alertas). Ha sido todo un proceso iterativo.

Una de las modificaciones que he decidido hacer mientras me encontraba realizando el prototipado de alta fidelidad es incluir las alertas en el menú de navegación. Anteriormente se podía llegar a dicha sección desde la pantalla de *settings* del usuario autentificado, pero luego la he quitado de esa sección, he puesto el acceso desde el home del usuario y desde el menú de navegación. He tomado esta decisión por considerar que al estar en el menú de navegación puede ser una manera para que usuarios no autentificados conozcan que existe la funcionalidad y que de la otra forma solo la verían una vez autentificados. Antes las hacia depender de la pagina de configuración por considerarlas conceptualmente una configuración que luego funcionarían por fuera. Pero me he dado cuenta o he decidido que conceptualmente es un servicio que se le brinda a los usuarios.

2.4.1 Definición de los casos de uso

“Los casos de uso de un sistema contienen los requisitos funcionales deseados o existentes, los actores (los actores describen el rol que presentan los usuarios del sistema) y las asociaciones que unen a actores y funcionalidades. Este conjunto determina asimismo las fronteras del sistema, es decir, las funcionalidades del sistema y aquellas que son externas a él.” [4]

Para realizar los casos de uso se han contemplado 5 actores diferentes: Usuario no autentificado, usuario anónimo y usuario autentificado, Los tres heredan de usuario. La diferencia entre usuario identificado y anónimo es que el anónimo no puede identificarse, pero esta autentificado en el sentido que puede configurar alertas y desde Firebase podemos hacérselas llegar como notificaciones push a su dispositivo. En cambio, los usuarios identificados podrían instalar la aplicación criptomonedas.app en múltiples dispositivos y en cada uno en el que se autentifique vera la misma información de su cuenta privada. El usuario 5 es el back-end de criptomonedas.app quien podrá enviar alertas y abrir ordenes en el exchange.

Cabe aclarar nuevamente en esta etapa que algunos de los casos de uso son deseados, como las alertas, que pueden no estar en el resultado final de esta primera versión. O que en proceso de desarrollo haya que aplicar alguna medida de contingencia ya mencionada. Lo importante es poder lograr de un mínimo producto viable en adelante.

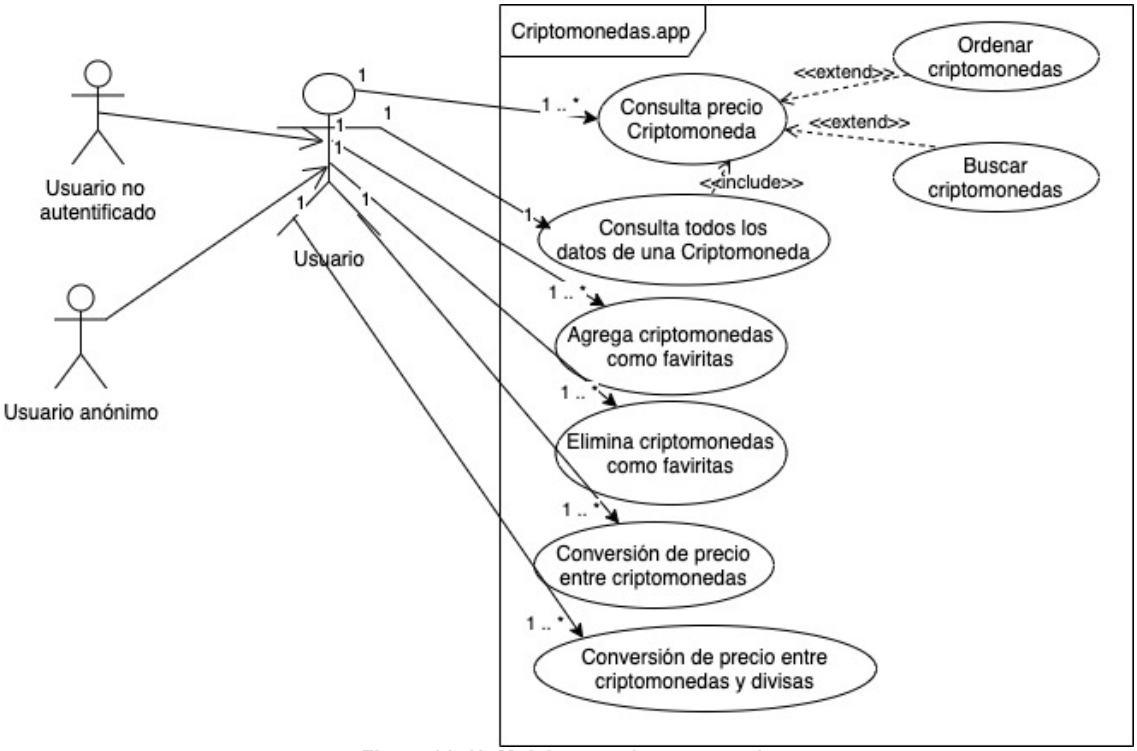


Figura 32: ULM del casos de uso usuario

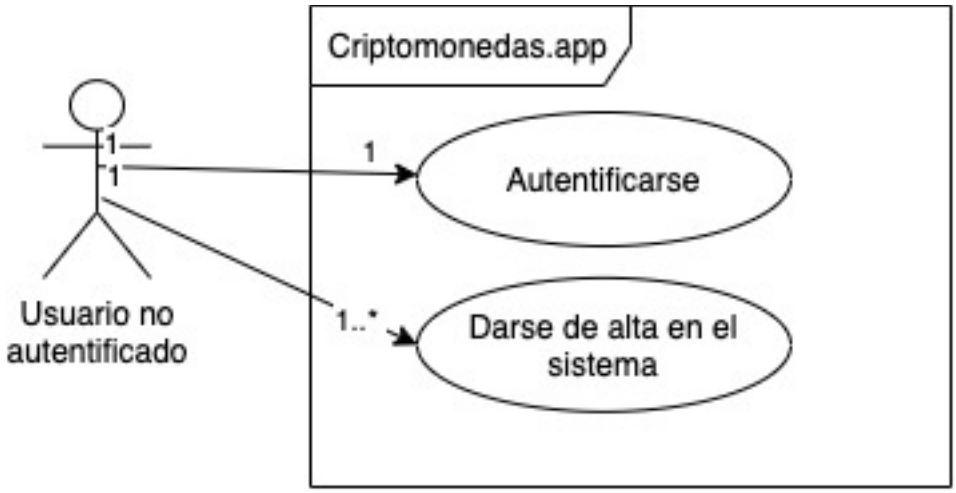


Figura 33: UML del caso de uso usuario no autentificado

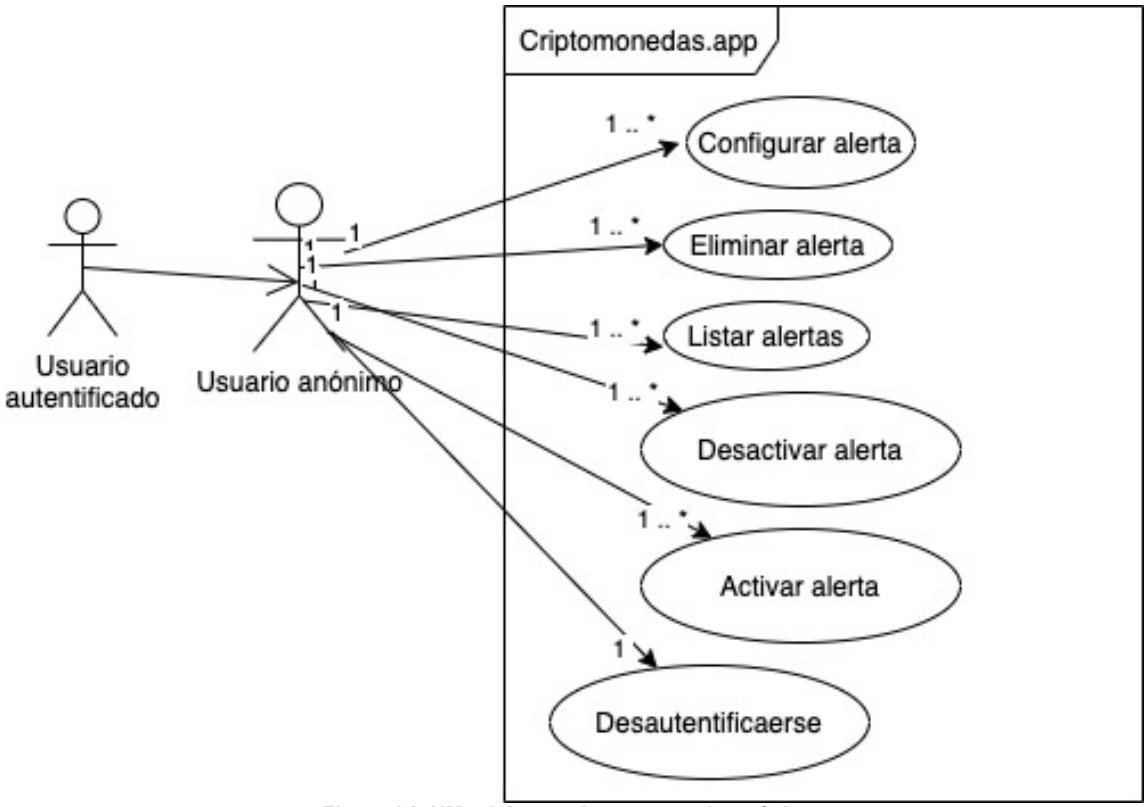


Figura 34: UML del caso de uso usuario anónimo

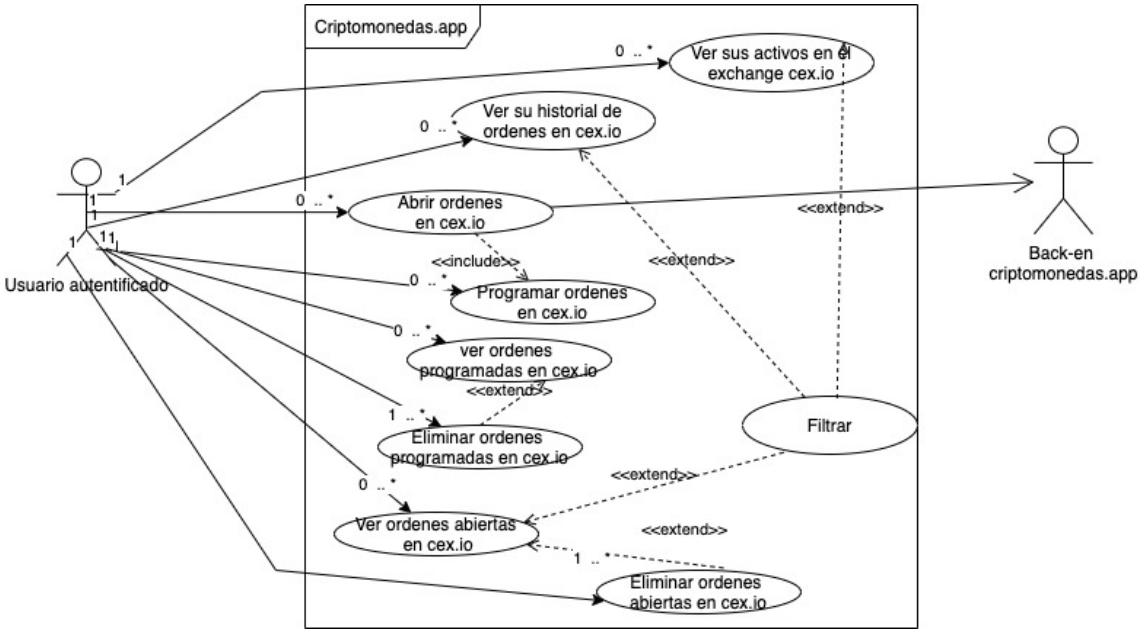


Figura 35: UML del caso de uso del usuario autentificado

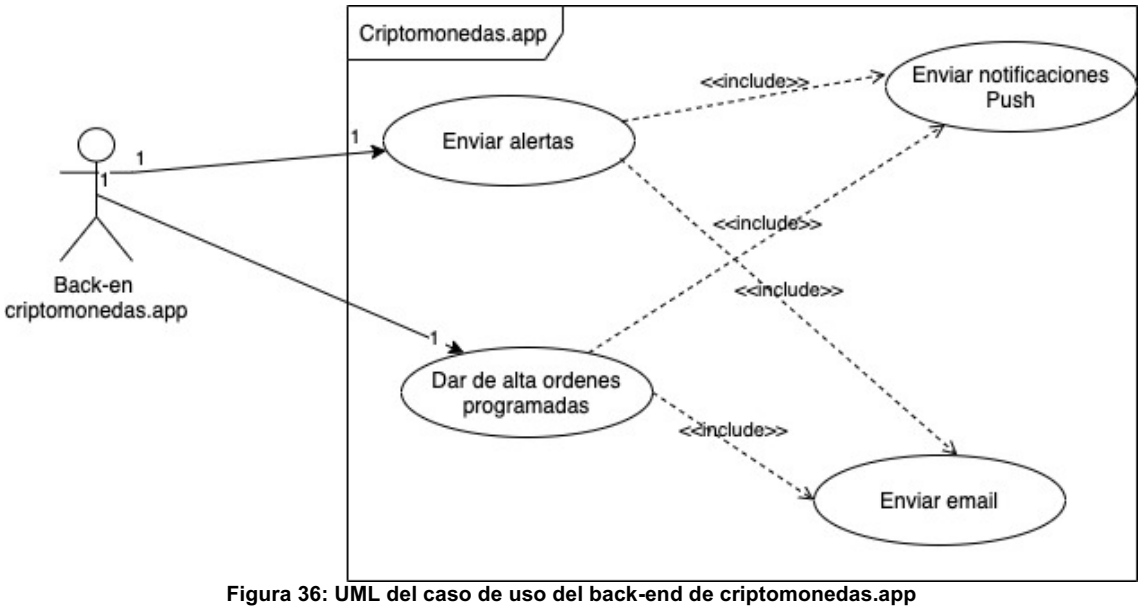


Figura 36: UML del caso de uso del back-end de criptomonedas.app

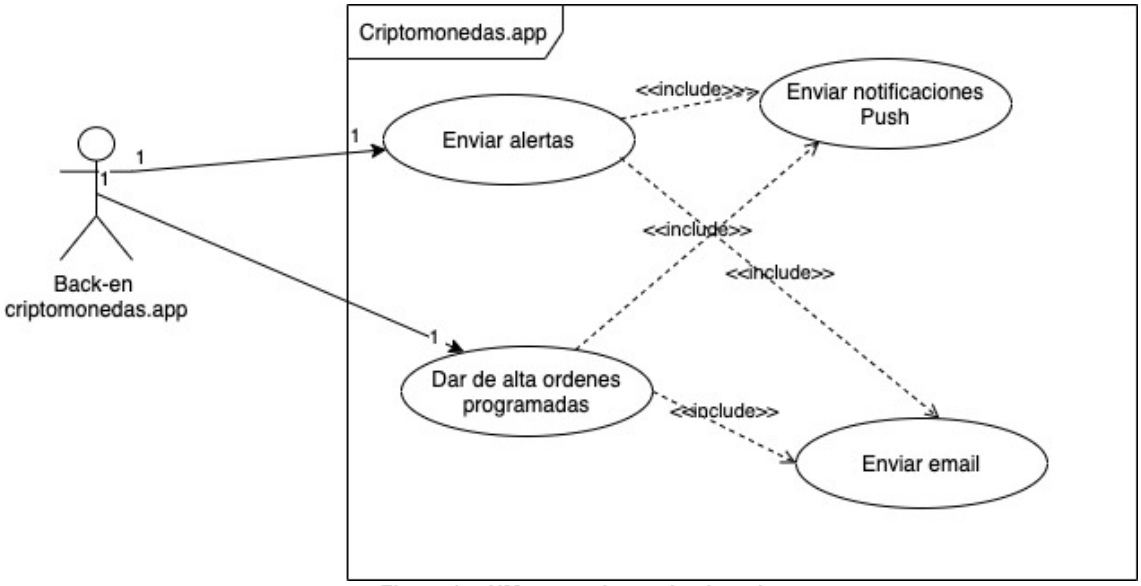


Figura 37: UML caso de uso back-end

Tabla 6: Caso de uso "Consultar datos de criptomonedas**"**

Identificador	**CU-001**
Nombre	**Consultar datos de criptomonedas**
Actor primario	Usuario de la aplicación
Sistema	Criptomonedas.app
Participantes	Usuario de la aplicación
Nivel	Objetivo del actor principal
Condiciones previas	El usuario debe encontrarse en la pantalla home o en la pantalla top 10 o en la pantalla todas las criptomonedas El dispositivo debe tener conexión a internet
Operaciones	
1	Ordenar
2	Buscar
3	Ir a la pantalla todas las criptomonedas
Extensiones	
1.A	¿La criptomoneda que se quiere consultar está a la vista?
1.A.1	Si sí y solo se quiere saber el precio, finalizar
1.A.2	Si sí y se quiere ver mas detalle seguir con la operación 3
1.A..3	Si no, seguir con la operación 2
2.A	¿Se quiere saber solo precio?
2.A.1	Si sí, finalizar
2.A.2	Si no, seguir con la operación 3

Tabla 7: Caso de uso "Añadir criptomoneda a lista de favorites**"**

Identificador	**CU-002**
Nombre	**Añadir criptomoneda a lista de favoritas**
Actor primario	Usuario de la aplicación
Sistema	Criptomonedas.app
Participantes	Usuario de la aplicación
Nivel	Objetivo del actor principal
Condiciones previas	El usuario debe encontrarse en una pantalla donde se listan criptomonedas o en la pantalla de detalle de una criptomoneda La criptomoneda no debe estar marcada como favorita
Operaciones	
1	Pulsar sobre el ícono para que la criptomoneda quede marcada como favorita

Tabla 8: Caso de uso "Eliminar criptomoneda de la lista de favorites**"**

Identificador	**CU-003**
Caso de uso	**Eliminar criptomoneda de la lista de favoritas**
Actor primario	Usuario de la aplicación
Sistema	Criptomonedas.app
Participantes	Usuario de la aplicación
Nivel	Objetivo del actor principal
Condiciones previas	El usuario debe encontrarse en una pantalla donde se listan criptomonedas o en la pantalla de detalle de una criptomoneda La criptomoneda debe estar marcada como favorita
Operaciones	
1	Pulsar sobre el ícono para que la criptomoneda ya no quede marcada como favorita

Tabla 9: Caso de uso "Conversión de precio entre criptomonedas**"**

Identificador	**CU-004**
Caso de uso	**Conversión de precio entre criptomonedas**
Actor primario	Usuario de la aplicación
Sistema	Criptomonedas.app
Participantes	Usuario de la aplicación
Nivel	Objetivo del actor principal
Condiciones previas	El usuario debe encontrarse en la pantalla criptoconversor (figura 18)
Operaciones	
1	Selección criptomoneda de la que se quiere averiguar el valor
2	Seleccionar la cantidad de la criptomoneda que se quiere saber el valor
3	Selecciona la criptomoneda en la que se quiere saber el valor de la criptomoneda seleccionada en la operación 1

Tabla 10: Caso de uso "Conversión de precio entre criptomonedas y divisas**"**

Identificador	**CU-005**
Caso de uso	**Conversión de precio entre criptomonedas y divisas**
Actor primario	Usuario de la aplicación
Sistema	Criptomonedas.app
Participantes	Usuario de la aplicación
Nivel	Objetivo del actor principal
Condiciones previas	El usuario debe encontrarse en la pantalla critpo a fiat (figura 19)

Operaciones	
1	Selección criptomoneda de la que se quiere averiguar el valor
2	Seleccionar la cantidad de la criptomoneda que se quiere saber el valor
3	Selecciona la divisa en la que se quiere saber el valor de la criptomoneda seleccionada en la operación 1

Tabla 11: Caso de uso "Darse de alta en el sistema**"**

Identificador	CU-006
Nombre	Darse de alta en el sistema
Actor primario	Usuario no autentificado
Sistema	Criptomonedas.app
Participantes	Usuario no autentificado
Nivel	Objetivo del actor principal
Condiciones previas	El usuario debe encontrarse en la pantalla sing in – sing up (figura 21) Las credenciales con que el usuario va a darse de alta no deben existir en el sistema
Operaciones	
1	Seleccionar el método con el que va a querer autentificarse
1.1	¿Quiere el método de correo electrónico y clave?
1.1.1	Si sí, Ingresar usuario y clave y apretar botón de sing up
1.1.1.2	Confirmar correo electrónico, fin
1.1.2	Si no, apretar el botón del servicio externo de autentificación que desee utilizar y seguir los pasos que el servicio externo le indique

Tabla 12: Caso de uso "Autentificarse en el sistema"

Identificador	**CU-007**
Caso de uso	**Autentificarse en el sistema**
Actor primario	Usuario no autentificado
Sistema	Criptomonedas.app
Participantes	Usuario no autentificado
Nivel	Objetivo del actor principal
Condiciones previas	El usuario debe encontrarse en la pantalla sing in – sing up (figura 21) Las credenciales con que el usuario va a autentificarse deben existir en el sistema
Operaciones	
1	Seleccionar el método con el que va a querer ingresar al sistema
1.1	¿Quiere el método de correo electrónico y clave?
1.1.1	Si sí, Ingresar usuario y clave y apretar botón de sing in
1.1.2	Si no, apretar el botón del servicio externo de autentificación que desee utilizar y seguir los pasos que el servicio externo le indique

Tabla 13: Caso de uso "Configurar alerta"

Identificador	**CU-008**
Caso de uso	**Configurar alerta**
Actor primario	Usuario anónimo
Sistema	Criptomonedas.app
Participantes	Usuario anónimo
Nivel	Objetivo del actor principal
Condiciones	El usuario debe encontrarse en la pantalla programar alerta

previas	(figura 31) El usuario debe estar autentificado, aunque sea de manera anónima
Operaciones	
1	Seleccionar la criptomoneda para la que desea configurar la alerta
2	Seleccionar el tipo de alerta (valor o variación)
3	Seleccionar las condiciones según el tipo de alerta seleccionado
4	Selecciona si además de notificaciones push desea notificaciones por correo electrónico
5	Apretar el botón de alta

Tabla 14: Caso de uso "Eliminar alerta**"**

Identificador	**CU-009**
Caso de uso	**Eliminar alerta**
Actor primario	Usuario anónimo
Sistema	Criptomonedas.app
Participantes	Usuario anónimo
Nivel	Objetivo del actor principal
Condiciones previas	El usuario debe encontrarse en la pantalla alerta criptomonedas (figura 20) El usuario debe estar autentificado, aunque sea de manera anónima El usuario debió haber dado de alta la alerta que desea eliminar
Operaciones	
1	Buscar la alerta que desea eliminar
2	Presionar el icono de eliminar

3	Confirmar la acción de eliminar
4	Selecciona si además de notificaciones push desea notificaciones por correo electrónico

Tabla 15: Caso de uso "Listar alertas**"**

Identificador	**CU-010**
Caso de uso	**Listar alertas**
Actor primario	Usuario anónimo
Sistema	Criptomonedas.app
Participantes	Usuario anónimo
Nivel	Objetivo del actor principal
Condiciones previas	El usuario debe encontrarse en la pantalla alerta criptomonedas (figura 20) El usuario debe estar autentificado, aunque sea de manera anónima
Operaciones	
1	Aplicar los filtros pertinentes para obtener el listado de alertas deseado

Tabla 16: Caso de uso "Desactivar alerta**"**

Identificador	**CU-011**
Caso de uso	**Desactivar alerta**
Actor primario	Usuario anónimo
Sistema	Criptomonedas.app
Participantes	Usuario anónimo
Nivel	Objetivo del actor principal

Condiciones previas	El usuario debe encontrarse en la pantalla alerta criptomonedas (figura 32) El usuario debe estar autentificado, aunque sea de manera anónima El usuario debió haber dado de alta la alerta que desea desactivar La alerta debe encontrarse en estado activa
Operaciones	
1	Buscar la alerta que desea desactiva
2	Presionar en el elemento indicado para desactivar una alerta
3	Confirmar la acción de desactivación

Tabla 17: Caso de uso "Activar alerta**"**

Identificador	**CU-012**
Caso de uso	**Activar alerta**
Actor primario	Usuario anónimo
Sistema	Criptomonedas.app
Participantes	Usuario anónimo
Nivel	Objetivo del actor principal
Condiciones previas	El usuario debe encontrarse en la pantalla alerta criptomonedas (figura 32) El usuario debe estar autentificado, aunque sea de manera anónima El usuario debió haber dado de alta la alerta que desea desactivar La alerta debe encontrarse en estado desactivada
Operaciones	
1	Buscar la alerta que desea activa
2	Presionar en el elemento indicado para activar una alerta

Tabla 18: Caso de uso "Desautentificarse**"**

Identificador	**CU-013**
Caso de uso	**Desautentificarse**
Actor primario	Usuario anónimo
Sistema	Criptomonedas.app
Participantes	Usuario anónimo
Nivel	Objetivo del actor principal
Condiciones previas	El usuario debe encontrarse en la pantalla de user home (figura 22) El usuario debe estar autentificado, aunque sea de manera anónima
Operaciones	
1	Pulsar el elemento indicado para des autentificarse
2	Confirmar que se desasea la des autentificación y no ha sido un accidente

Tabla 19: Caso de uso "Ver activos que posee en el exchange cex.io**"**

Identificador	**CU-014**
Caso de uso	**Ver activos que posee en el exchange cex.io**
Actor primario	Usuario autentificado
Sistema	Criptomonedas.app
Participantes	Usuario autentificado
Nivel	Objetivo del actor principal
Condiciones previas	El usuario debe estar autentificado El usuario debe haber ingresado los datos necesarios para que criptomonedas.app pueda acceder a los datos de su cuenta en el exchange Los datos ingresados para que criptomonedas.app acceda a la

	información de la cuenta del usuario en el exchange tienen que ser válidos El usuario debe encontrarse en la pantalla home de cex.io (figura 23)
Operaciones	
1	Seleccionar la divisa en que quiere ver cuando suman sus activos
2	Seleccionar la criptomoneda en que sea ver cuanto suman sus activos

Tabla 20: Caso de uso "Ver histórico de ordenes del exchange cex.io**"**

Identificador	**CU-015**
Caso de uso	**Ver histórico de ordenes del exchange cex.io**
Actor primario	Usuario autentificado
Sistema	Criptomonedas.app
Participantes	Usuario autentificado
Nivel	Objetivo del actor principal
Condiciones previas	El usuario debe estar autentificado El usuario debe haber ingresado los datos necesarios para que criptomonedas.app pueda acceder a los datos de su cuenta en el exchange Los datos ingresados para que criptomonedas.app acceda a la información de la cuenta del usuario en el exchange tienen que ser válidos El usuario debe encontrarse en la pantalla historial de ordenes del exchange (figura 26) El usuario debe tener ordenes que se hayan ejecutado anteriormente en la cuenta del exchange a la que ha dado su acceso
Operaciones	
1	Aplicar los filtros para visualizar las órdenes históricas que desea

Tabla 21: Caso de uso "Programar orden en exchange cex.io"

Identificador	**CU-016**
Caso de uso	**Programar orden en exchange cex.io**
Actor primario	Usuario autentificado
Sistema	Criptomonedas.app
Participantes	Usuario autentificado
Nivel	Objetivo del actor principal
Condiciones previas	El usuario debe estar autentificado El usuario debe haber ingresado los datos necesarios para que criptomonedas.app pueda dar de alta ordenes en su cuenta del exchange Los datos ingresados para que criptomonedas.app acceda a la información de la cuenta del usuario en el exchange tienen que ser válidos El usuario debe encontrarse en la pantalla programar orden (figura 28)
Operaciones	
1	Seleccionar criptomoneda o divisa a comprar
2	Seleccionar con qué criptomoneda o divisa quiere comprar
3	Ingresar la cantidad de la criptomoneda o divisa que desea comprar o la cantidad de la criptomoneda o divisa con la que desea comprar.
3	Seleccionar el precio al que quiere realizar la compra
4	Apretar en el elemento indicado para dar de alta la orden
5	Confirmar el alta de la orden

Tabla 22: Caso de uso "Ver ordenes programadas en exchange cex.io"

Identificador	**CU-017**
Caso de uso	**Ver ordenes programadas en exchange cex.io**

Actor primario	Usuario autentificado
Sistema	Criptomonedas.app
Participantes	Usuario autentificado
Nivel	Objetivo del actor principal
Condiciones previas	El usuario debe estar autentificado El usuario debe haber ingresado los datos necesarios para que criptomonedas.app pueda acceder a los datos de su cuenta en el exchange Los datos ingresados para que criptomonedas.app acceda a la información de la cuenta del usuario en el exchange tienen que ser válidos El usuario debe encontrarse en la pantalla ordenes programadas del exchange (figura 27) El usuario debe tener ordenes programadas para el exchange
Operaciones	
1	Aplicar los filtros para visualizar las órdenes programadas que desea

Tabla 23: Caso de uso "Eliminar ordenes programadas en exchange cex.io**"**

Identificador	**CU-018**
Caso de uso	**Eliminar ordenes programadas en exchange cex.io**
Actor primario	Usuario autentificado
Sistema	Criptomonedas.app
Participantes	Usuario autentificado
Nivel	Objetivo del actor principal
Condiciones previas	El usuario debe estar autentificado El usuario debe haber ingresado los datos necesarios para que criptomonedas.app pueda acceder a los datos de su cuenta en el exchange Los datos ingresados para que criptomonedas.app acceda a la

	información de la cuenta del usuario en el exchange tienen que ser válidos El usuario debe encontrarse en la pantalla ordenes programadas del exchange (figura 27) El usuario debe tener ordenes programadas para el exchange
Operaciones	
1	Aplicar los filtros para visualizar las órdenes programadas que desea eliminar
2	Pulsar el elemento indicado para eliminar la orden
3	Confirmar la eliminación

Tabla 24: Caso de uso "Ver ordenes abiertas en exchange cex.io**"**

Identificador	**CU-019**
Caso de uso	**Ver ordenes abiertas en exchange cex.io**
Actor primario	Usuario autentificado
Sistema	Criptomonedas.app
Participantes	Usuario autentificado
Nivel	Objetivo del actor principal
Condiciones previas	El usuario debe estar autentificado El usuario debe haber ingresado los datos necesarios para que criptomonedas.app pueda acceder a los datos de su cuenta en el exchange Los datos ingresados para que criptomonedas.app acceda a la información de la cuenta del usuario en el exchange tienen que ser válidos El usuario debe encontrarse en la pantalla ordenes abiertas del exchange (figura 25) El usuario debe tener ordenes programadas para el exchange
Operaciones	
1	Aplicar los filtros para visualizar las órdenes programadas que desea eliminar

2	Pulsar el elemento indicado para eliminar la orden
3	Confirmar la eliminación

Tabla 25: Caso de uso "Eliminar ordenes abiertas en exchange cex.io**"**

Identificador	**CU-020**
Caso de uso	**Eliminar ordenes abiertas en exchange cex.io**
Actor primario	Usuario autentificado
Sistema	Criptomonedas.app
Participantes	Usuario autentificado
Nivel	Objetivo del actor principal
Condiciones previas	El usuario debe estar autentificado El usuario debe haber ingresado los datos necesarios para que criptomonedas.app pueda acceder a los datos de su cuenta en el exchange Los datos ingresados para que criptomonedas.app acceda a la información de la cuenta del usuario en el exchange tienen que ser válidos El usuario debe encontrarse en la pantalla ordenes abiertas del exchange (figura 25) El usuario debe tener ordenes abiertas en el exchange
Operaciones	
1	Aplicar los filtros para visualizar las órdenes abiertas que desea eliminar
2	Pulsar el elemento indicado para eliminar la orden
3	Confirmar la eliminación

Tabla 26: Caso de uso "Abrir orden en el exchange cex.io**"**

Identificador	**CU-021**

Caso de uso	**Abrir orden en el exchange cex.io**
Actor primario	Usuario autentificado
Sistema	Criptomonedas.app
Participantes	Usuario autentificado, back end de criptomonedas.app
Nivel	Objetivo del actor principal
Condiciones previas	El usuario debe estar autentificado El usuario debe haber ingresado los datos necesarios para que criptomonedas.app pueda dar de alta ordenes en su cuenta del exchange El usuario debe encontrarse en la pantalla programar orden (figura 28) El usuario debe tener en su cuenta del exchange el capital disponible que desea utilizar para realizar la compra
Operaciones	
1	Seleccionar criptomoneda o divisa a comprar
2	Seleccionar con qué criptomoneda o divisa quiere comprar
3	Ingresar la cantidad de la criptomoneda o divisa que desea comprar o la cantidad de la criptomoneda o divisa con la que desea comprar.
3	Seleccionar el precio al que quiere realizar la compra
4	Apretar en el elemento indicado para dar de alta la orden
5	Confirmar el alta de la orden
6	Al back-end de criptomonedas.app abre la orden en el exchange

Tabla 27: Caso de uso "Abrir orden en el exchange de pare de un usuario**"**

Identificador	**CU-022**
Caso de uso	**Abrir orden en el exchange de pare de un usuario**

Actor primario	Back-end de la aplicación
Sistema	Criptomonedas.app
Participantes	Back-end de la aplicación
Nivel	Objetivo del usuario
Condiciones previas	Deben existir registros de ordenes programadas dadas de alta por usuarios en la base de datos Las credenciales ingresadas por el usuario al sistema deben seguir siendo validas El usuario debe tener saldo suficiente en su cuenta para abrir la orden
Operaciones	
1	Se consulta la base de datos por las ordenes que hay programadas
2	Se revisa una por una las ordenes programadas
3	Se verifica en cada alerta si las condiciones están sucediendo
3.1	Si no, fin
3.2	Si sí, se envía notificación push y se mira si además el usuario ha pedido ser notificado por correo electrónico
3.2.1	Si no, fin
3.2.1	Si sí, se envía la notificación por correo electrónico

Tabla 28: Caso de uso "Enviar alertas a usuarios**"**

Identificador	**CU-023**
Caso de uso	**Enviar alertas a usuarios**
Actor primario	Back-end de la aplicación
Sistema	Criptomonedas.app

Participantes	Back-end de la aplicación
Nivel	Objetivo de usuario
Condiciones previas	Deben existir registros de alertas dadas de alta por usuarios en la base de datos Las alertas deben tener estado activo La condición que el usuario a puesto para que se dispare la alerta tiene que estar sucediendo
Operaciones	
1	Se consulta la base de datos por las alertas activas
2	Se revisa una por una las alertas activas
3	Se verifica en cada alerta si las condiciones están sucediendo
3.1	Si no, fin
3.2	Si sí, se envía notificación push y se mira si además el usuario ha pedido ser notificado por correo electrónico
3.2.1	Si no, fin
3.2.1	Si sí, se envía la notificación por correo electrónico

Observaciones de los caos de uso:

En el caso de uso “Abrir orden en el exchange cex.io” se ha tomado la decisión de que sea un caso de uso de las ordenes programadas que se abrirá por parte del exchange en la primera ejecución del script encargado en el bask-end. Se podía haber hecho directamente por la aplicación y sería mejor aún, pero al tratarse de una versión gratuita y teniendo en cuenta que la lógica para realizar esta acción ya esta hecha en el back-en y no en la aplicación móvil, me pareció bueno hacerlo de esta manera y en todo caso ofrecerlo más adelante como una ventaja de una versión de pago de la aplicación.

2.4.2 Diseño de la arquitectura

1) El diagrama UML correspondiente al diseño de la base de datos.

Si bien se ha optado por utilizar una base de datos NoSQL, se ha confeccionado el diagrama UML correspondiente al diseño de la base de datos

relacional. Se ha realizado de esta manera porque siempre es posible necesitar migrar o replicar en el dispositivo de los usuarios las bases de datos que se encuentran en la nube. Esto último, con la intención de mejorar la experiencia de usuario al máximo incluso en momentos en que no hay conexión a internet (requisito no funcional RnF007). Es muy probable que no se llegue a realizar esta replicación para la primera versión, pero si es muy deseable que esté implementado para la próxima actualización.

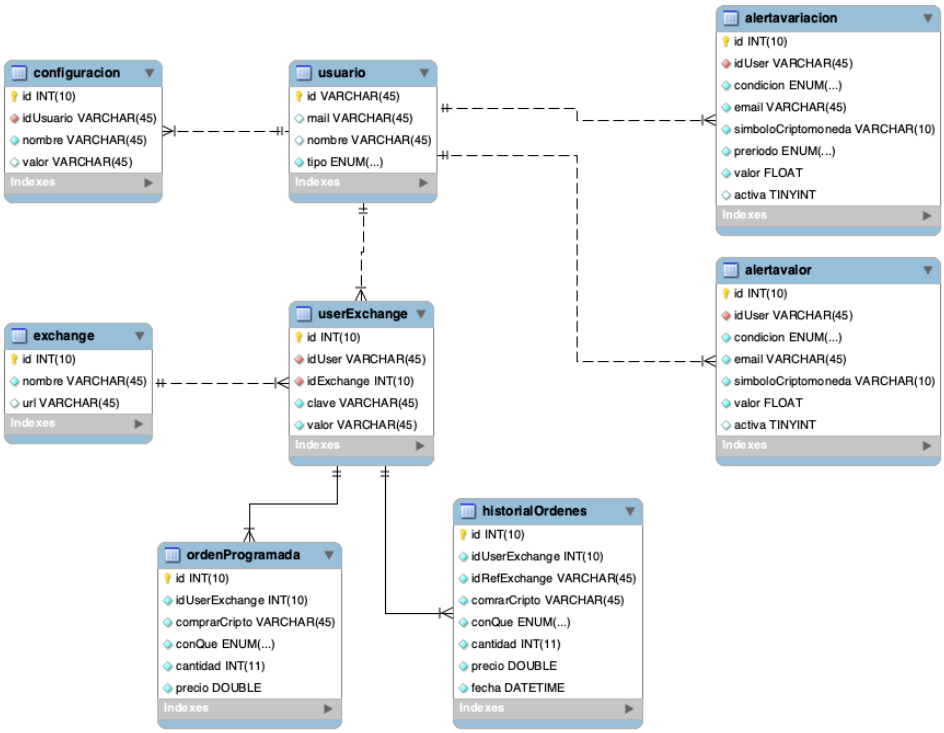


Figura 38: Diagrama UML correspondiente al diseño de la base de datos

La estructura de tablas de la base de datos se puede crear ejecutando las siguientes sentencias en una base de datos MySQL:

```
CREATE TABLE `usuario` (
  `id` varchar(45) NOT NULL,
  `mail` varchar(45) DEFAULT NULL,
  `nombre` varchar(45) DEFAULT NULL,
  `tipo` enum('anonimo','identificado') NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `id_UNIQUE` (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;

CREATE TABLE `exchange` (
  `id` int unsigned NOT NULL AUTO_INCREMENT,
  `nombre` varchar(45) NOT NULL,
  `url` varchar(45) DEFAULT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `id_UNIQUE` (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;

CREATE TABLE `userExchange` (
`id` int unsigned NOT NULL AUTO_INCREMENT,
  `idUser` varchar(45) NOT NULL,
```

```
  `idExchange` int unsigned NOT NULL,
  `clave` varchar(45) NOT NULL,
  `valor` varchar(45) NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `id_UNIQUE` (`id`),
  UNIQUE KEY `UC_usr_exch_val` (`idUser`,`idExchange`),
  CONSTRAINT `userexchange_ibfk_1` FOREIGN KEY (`idUser`) REFERENCES `usuario` (`id`) ON DELETE
CASCADE ON UPDATE CASCADE,
  CONSTRAINT `userexchange_ibfk_2` FOREIGN KEY (`idExchange`) REFERENCES `exchange` (`id`) ON DELETE
CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;

CREATE TABLE `ordenProgramada` (
  `id` int unsigned NOT NULL AUTO_INCREMENT,
  `idUserExchange` int unsigned NOT NULL,
  `comprarCripto` varchar(45) NOT NULL,
  `conQue` enum('mayor','menor') NOT NULL,
  `cantidad` int NOT NULL,
  `precio` double NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `id_UNIQUE2` (`id`),
  CONSTRAINT `userexchange_ibfk_3` FOREIGN KEY (`idUserExchange`) REFERENCES `userExchange` (`id`) ON
DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;

CREATE TABLE `historialOrdenes` (
  `id` int unsigned NOT NULL AUTO_INCREMENT,
  `idUserExchange` int unsigned NOT NULL,
  `idRefExchange` varchar(45) NOT NULL,
  `comrarCripto` varchar(45) NOT NULL,
  `conQue` enum('mayor','menor') NOT NULL,
  `cantidad` int NOT NULL,
  `precio` double NOT NULL,
  `fecha` datetime NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `id_UNIQUE3` (`id`),
  UNIQUE KEY `idRefExchane_UNIQUE` (`idRefExchange`),
  CONSTRAINT `userexchange_ibfk_4` FOREIGN KEY (`idUserExchange`) REFERENCES `userExchange` (`id`) ON
DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;

CREATE TABLE `configuracion` (
  `id` int unsigned NOT NULL AUTO_INCREMENT,
  `idUsuario` varchar(45) NOT NULL,
  `nombre` varchar(45) NOT NULL,
  `valor` varchar(45) DEFAULT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `id_UNIQUE` (`id`),
  UNIQUE KEY `UC_Empresa` (`nombre`,`idUsuario`),
  KEY `idUsuario` (`idUsuario`),
  CONSTRAINT `configuracion_ibfk_1` FOREIGN KEY (`idUsuario`) REFERENCES `usuario` (`id`) ON DELETE
CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;

CREATE TABLE `alertavariacion` (
  `id` int unsigned NOT NULL AUTO_INCREMENT,
  `idUser` varchar(45) NOT NULL,
  `condicion` enum('mayor','menor') NOT NULL,
  `email` varchar(45) NOT NULL,
  `simboloCriptomoneda` varchar(10) NOT NULL,
  `preriodo` enum('1h','24h','7d','30d') NOT NULL,
  `valor` float NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `id_UNIQUE` (`id`),
  UNIQUE KEY `UC_usr_aler_iniq` (`idUser`,`condicion`,`simboloCriptomoneda`,`preriodo`,`valor`),
  CONSTRAINT `alertavariacion_ibfk_1` FOREIGN KEY (`idUser`) REFERENCES `usuario` (`id`) ON DELETE
CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;

CREATE TABLE `alertavalor` (
  `id` int unsigned NOT NULL AUTO_INCREMENT,
  `idUser` varchar(45) NOT NULL,
  `condicion` enum('mayor','menor') NOT NULL,
  `email` varchar(45) NOT NULL,
  `simboloCriptomoneda` varchar(10) NOT NULL,
```

```
  `valor` float NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `id_UNIQUE` (`id`),
  UNIQUE KEY `UC_usr_aler_iniq` (`idUser`,`condicion`,`simboloCriptomoneda`,`valor`),
  CONSTRAINT `alertavalor_ibfk_1` FOREIGN KEY (`idUser`) REFERENCES `usuario` (`id`) ON DELETE CASCADE
ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
```

Se ha creado una tabla *configuración* para poder guardar cualquier tipo de configuración del usuario tan solo agregando un nombre en clave y un valor. Se ha tomado esta decisión para dejar asentado que habrá una tabla que contendrá las configuraciones de los usuarios.

Como he optado por un método de desarrollo rápido de aplicaciones es posible que las configuraciones que el usuario vaya a poder hacer cambien o aumenten en la etapa de desarrollo. Una vez que sea un poco más estable y esté más seguro de las configuraciones que se puedan realizar, cambiaría la estructura de esta tabla para que contenga las columnas exactas correspondientes a las configuraciones que el usuario podría realizar. Cabe recordar además que la base seleccionada es de tipo NoSQL y que por el momento no habrá necesidad de usar esta estructura. Además, es muy probable que esta funcionalidad no quede implementada en esta primera versión ya que corresponde con un requisito funcional deseado (RFd_008) pero no importante.

Lo que he querido marcar es que un usuario puede tener muchas alertas de tipo variación (tabla alertaVariación) y muchas alertas de tipo precio (tabla alertaPrecio). También un usuario puede tener muchas configuraciones (hasta que la tabla quede mas especificada como se explicó en el párrafo anterior). Luego existen muchos exchanges (vale recordar que en esta versión solo existirá cex.io, pero la idea es soportar cada vez más y que la información de todos este organizada y mostrada de manera homogénea) donde los usuarios pueden tener una cuenta, pero el usuario puede tener cuenta en muchos exchanges.

De esta última relación es que ha surgido la existencia de la tabla *userExchange*. Luego existe la tabla *ordenProgramada* y *historialOrdenes* relacionadas con la tabla *userExchange* ya que cada orden pertenece a la cuenta de un usuario en un Exchange. *ordenProgramada* contiene las ordenes que el back-end tiene que verificar cada cierto tiempo y dar de alta en caso de que haya el dinero suficiente en la cuenta de dicho usuario. *historialOrdenes* guarda las ordenes ejecutadas y corresponde con el requerimiento funcional deseado RFd_001.

2) El diagrama UML correspondiente al diseño de las entidades y clases.

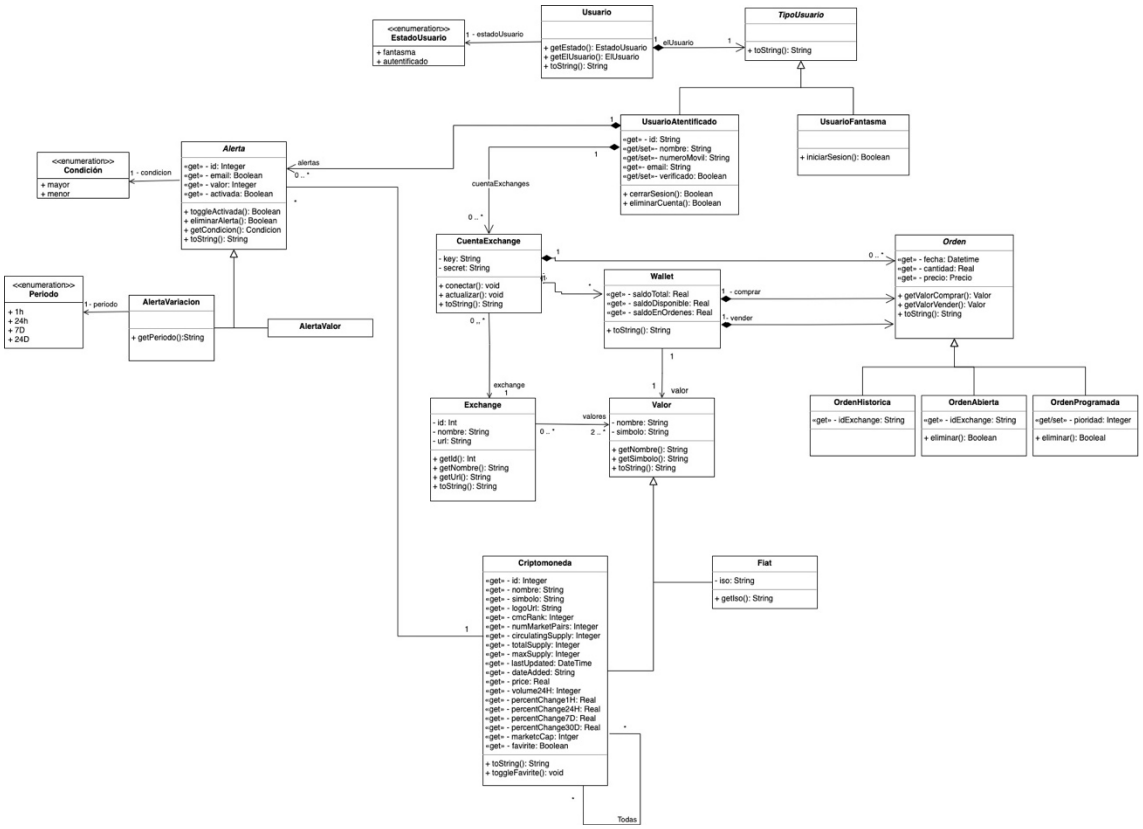


Figura 39: Diagrama UML de clases

Observaciones del diagrama de clases:

La clase OrdenAbierta corresponde a las ordenes abiertas en el Exchange, no existe en la base de datos o back-end de criptomonedas.app como las otras clases de ordenes ya que se consultan directamente en el exchange que es donde residen y donde el usuario puede cancelarlas también. En cambio, OrdenHistorica es una orden que ya ha sido ejecutada y que ya no puede cancelarse porque ya ha sido ejecutada. Y la clase OrdenProgrmada solo existe en criptomonedas.app y no en el Exchange ya que el Exchange no permite la programación de ordenes que se den de alta según condiciones que puedan variar en el futuro.

3) Un diagrama explicativo de la arquitectura del sistema (siguiendo el paradigma MVC, SOA, EDA, etc.)

Si bien me he propuesto que la aplicación funcione en móviles Android (Requerimiento no funcional RnF_003) en modo de orientación portrait (Requerimiento no funcional RnF_003) por cuestiones practicas debido al tiempo de desarrollo, la tecnología de desarrollo es multiplataforma (Flutter). Por este motivo he seleccionado una arquitectura Modelo-Vista-Controlador o MVC (Model-View-Controler) ya que me permite separar la vista de la lógica de la aplicación y los datos. MVC me permite que en posteriores versiones pueda implementar diseños de vista acordes a cualquier tipo de dispositivo y orientación si tocar la lógica o los datos.

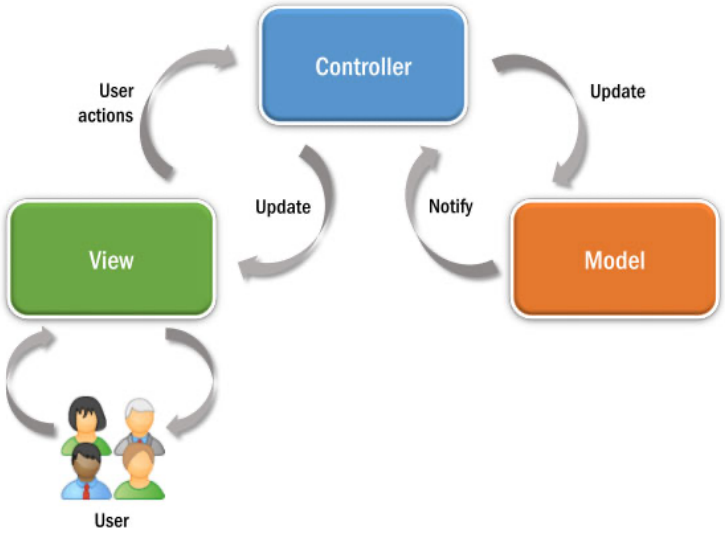


Figura 40: Modelo vista controlador (https://commons.wikimedia.org/wiki/File:Model-View-Controller_architectural_pattern.svg)

El modelo MVC tiene los siguientes Componentes:

- La vista (view): Es la interfaz de la aplicación, es lo que ve el usuario y el componente por el cual interactúa con la aplicación.

- El controlador (controller): Es la lógica de la aplicación,

- El modelo (model): Son los datos de la aplicación que se procesan mediante el controlador y se muestran en la vista.

En este caso el modelo viene de diferentes fuentes, una base de datos Firebase, la memoria interna del dispositivo del usuario, una API para interactuar con el exchange cex.io y una API para obtener toda la información actualizada de las criptomonedas de coinmarketcap.com. Luego estos datos de fuentes tan variadas son tratados y procesados por el controlador que se los pasará a la componente vista para que el usuario pueda verlos y interactuar con los mismos.

3. Implementación

3.1 Entorno de desarrollo

Como entorno de desarrollo he usado Android Studio [6]. He seleccionado este IDE por la única razón que en medio de esta etapa hubo una actualización de versión de Flutter (a pasado de la 2.5.2 a la 2.5.3) y al ser el IDE oficial ya contaba con la posibilidad de depurar código de la nueva versión. Había comenzado usando Visual Studio Code pero en la actualización algunas extensiones, como la de depuración, han dejado de funcionar.

En cuanto a Flutter he usado la versión 2.5.3

Figura 41: Versión de Flutter

La aplicación la he probado en un móvil físico Google Pixel 3ª con Android 12 [7] y un emulador Nexus 6 con Android 11. Finalmente, para esta primera versión de la aplicación, se ha desarrollado para dispositivos Android y en modo portrait cumpliendo así con los requisitos no funcionales RnF_003 y RnF_008. La mínima versión de Android con la que se puede usar la aplicación desarrollada es la denominada kitkat ya que es la mínima versión soportada por Flutter 2.5.3.

3.2 Tecnologías

Como se ha mencionado en el apartado anterior se ha usado Flutter 2.5.3 para el desarrollo. Pero también se ha usado:

- Firebase Authentication [8], para el manejo de usuarios identificados.
- Firebase Firestore [9], como base de datos
- Firebase Cloud Messaging [10], para el envió de mensajes a usuarios
- Python 3.7.6 como lenguaje de back-end
- CronTab [11], para la ejecución periódica del back-end

Para la ejecución de Python y CronTab utilizo un ordenador MacBook Pro (13-inch, Mid 2012) conectado a internet las 24 horas, pero se puede utilizar cualquier ordenador con Python 3.7.6 instalado, que permita ejecutar archivos Python de manera periódica y que posea conexión a internet ininterrumpida.

3.1.1 Dependencias

Como he mencionado la aplicación la he desarrollado con Flutter ya que me interesaba aprender esta tecnología y hacer la aplicación multidispisitivo con un único desarrollo. Finalmente he tenido tiempo para testearla y asegurar el correcto funcionamiento en móviles Android, pero queda preparada para con un poco más de tiempo testearla en Tablet, iOS y Web tanto para dispositivos móviles, como tablets y ordenadores y ajustar las adaptaciones necesarias tanto en landscape como en portrait.

Para desarrollar la aplicación he usado las siguientes dependencias:

```
firebase_core: "^1.7.0"
firebase_auth: "^3.1.3"
cloud_firestore: ^3.1.0
firebase_messaging: "^11.1.0"
google_sign_in: "^5.1.1"
flutter_signin_button: ^2.0.0
email_validator: '^2.0.1'
crypto: ^3.0.1
syncfusion_flutter_charts: ^19.3.54
syncfusion_flutter_datagrid: ^19.3.54
syncfusion_flutter_treemap: ^19.3.55-beta
http: ^0.13.4
provider: ^6.0.1
font_awesome_flutter: ^9.1.0
line_awesome_flutter: ^2.0.0
shared_preferences: ^2.0.8
share: ^2.0.4
google_mobile_ads: ^1.0.0
google_fonts: ^2.1.0
flutter_facebook_auth: ^3.5.6
```

Firebase Core, Firebase Auth, Cloud Firestore, Firebase Messaging para interactuar con Firebase.

Google Sign In para poder realizar el login con Google sign in a travez de Firebase.

Flutter Signin Button para crear un botón de Google Sign In In standard.

Crypto para poder interactuar con la Api del Exchange Cex.io

Syncfusion Flutter Charts, Syncfusion Flutter Datagrid, Syncfusion Flutter treemap para las gráficas estadísticas y manejo de tablas.

Http para las peticiones Http.

Provider para la utilización del patrón provider recomendado por Flutter.

Font Awesome Flutter y Line Awesome Flutter para disponer de multitude de íconos.

Shared preferences para poder almacenar los settings en el dispositivo de los usuarios.

Share para que los usuarios puedan recomendar la aplicación de manera rápida a través de Intents en Android y UIActivityViewController en iOS.

Para añadir publicidad y poder monetizar la aplicación uso el paquete google_mobile_ads y para usar fuentes mas amigables google_fonts.

Para la autentificación con Facebook se ha usado el paquete flutter_facebook_auth.

3.1.2 Autentificación

En cuanto a la autentificación de los usuarios se ha decidido dejar únicamente la de Google Auth. Todas las autentificaciones se manejan a través de Firebase por facilidad e integración de seguridad con Firestore. Sin embargo, en el código se deja implementado el método de autentificación por Facebook que se ha probado y funciona correctamente, pero se ha quitado para realizar la entrega del TFM porque Facebook requiere que la aplicación sea revisada una vez que se da de alta en las App Store. Como la aplicación no se ha subido a ninguna App Store, no se ha podido validar y por ende solo puede usarse este método con mi email, que es el email con el que se ha dado de alta toda la configuración en Facebook Developers. En este sentido, en la pantalla de login solo se ha comentado el botón que permitiría autentificarse con este método, pero se deja implementado para descomentar el botón una vez que la aplicación quede validada por Facebook.con en un futuro.

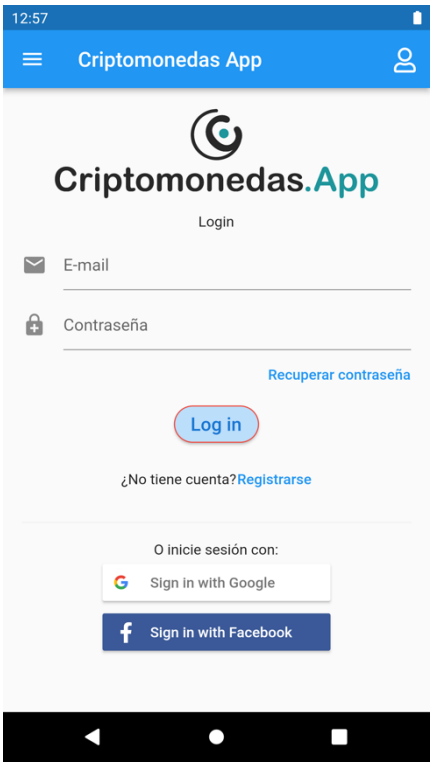


Figura 42: Inicio de sesión con Facebook y de usuario y contraseña que en la entrega quedan comentados

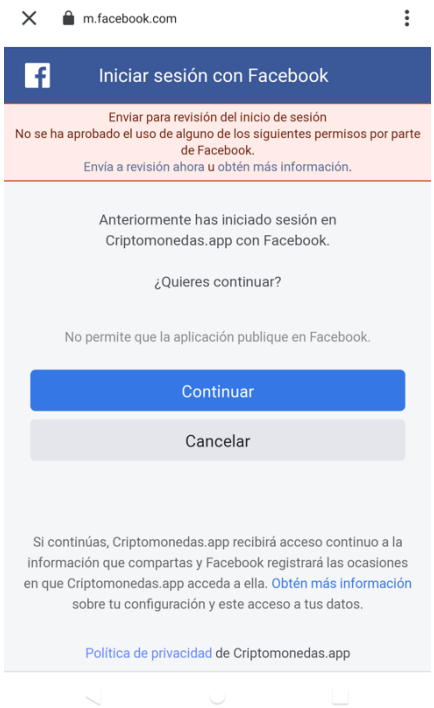


Figura 43: Facebook exige que la aplicación sea revisada

No se ha usado la autentificación con Apple.com ya que no estoy inscripto aún en el programa de desarrolladores de Apple. Sin estar inscripto en dicho programa, Apple no permite probar la autentificación con su método.

Se deja funcionado, pero comentado, el método de autentificación con email y contraseña ya que no he implementado el cambio o recuperación de contraseña. No lo he implementado porque para poder enviar emails con el dominio criptomonedas.app es necesario poder contratar un servidor SMTP que permita enviar emails de tal manera que no sean marcados como spam. Esto se puede implementar muy fácilmente por ejemplo pagando una suscripción en Google Workspace, pero para el TFM no he contratado dicho servicio. Por esta misma razón, se deja para versiones posteriores el uso de otros tipos de autentificación como anónima y SMS.

He dejado la autentificación de Google por ya que es el método más extendido entre los usuarios y rápido para poder testear y llevar adelante el desarrollo de la aplicación en el proceso de implementación. Y porque fácilmente podrá ser usado por terceras personas que no hayan estado involucradas en el proceso de desarrollo de la aplicación, no requiere validaciones ni envíos de correo electrónico de confirmación.

3.1.3 Base de datos

Como se había mencionado en el apartado 2.4.2 se ha optado por una base de datos NoSQL, más precisamente Firebase Firestore. Se ha optado por esta tecnología varios motivos:

- El primero es seguir aprendiendo a utilizar esta tecnología -ya vista en otras asignaturas del master- por considerarla muy dinámica, rápida y segura,
- Por el hecho de que posee una seguridad muy buena y me permite desentenderme de su administración o la administración de otro motor de base de datos que si bien poseo los conocimientos como para hacerlo, solo tenia tiempo para ocuparme del desarrollo de la aplicación en si.
- La seguridad es muy buena para el desarrollo móvil y se complementa perfectamente con Firebase *Auth*

Si bien en el apartado 2.4.2 se ha expuesto el diseño de la base de datos para un modelo relacional, las bases de datos NoSQL se estructuran en base a colecciones y documentos. Se estructuran de una manera totalmente diferente. En Firestore en particular, se puede estructurar la base de datos de tal manera que junto a los permisos configurados adecuadamente pueda gestionarse la seguridad fácilmente logrando que cada usuario solo pueda acceder a su propia información y no a la de otros usuarios. Todo esto gestionado y administrado por Google. En este sentido, he usado las siguientes reglas de permisos:

```
rules_version = '2';
service cloud.firestore {
  match /databases/{database}/documents {
    // Allow only authenticated content owners access
    match /usuarios/{userId}/{documents=**} {
      allow read, write: if request.auth != null && request.auth.uid == userId
    }
        match /{document=**}{
                allow read, write: if false
                }
  }
}
```

Dichas reglas junto a la manera que he estructurado el árbol de colecciones y documentos son la base principal de la seguridad. Básicamente hacen que a los documentos por debajo de Usuarios solo pueda acceder el usuario cuyo id coincida de manera exacta con el id de documento que se encuentra justo por de bajo de la colección usuarios y por delante de todas la colecciones y documentos correspondientes a la información privada de cada usuario.

La estructura de la base de datos en Firestore ha quedado de la siguiente manera:

Ruta /usuarios/{userId}/	
email	Email del usuario
nombre	Nombre del usuario
teléfono	Teléfono del usuario
tokens	Tokens de los dispositivos con los que el usuario se ha conectado y con los que a través de Firebase Cloud Messaging se le enviarán las notificaciones a los usuarios.

Ruta /usuarios/{userId}/alertas_precio**/{alertId}/**	
activa	Indica si la alerta esta marcada como activada o no. Si esta activada se le mandará la notificación al usuario en caso de que se cumplan las condiciones

	de la alerta. Al ejecutarse la alerta quedará marcada como desactivada.
campo	En esta primera versión solo es posible poder poner una alerta para el campo precio de las criptomonedas, como se explica mas adelante en la sección de alertas, Se deja preparado este campo porque se espera poder implementar alertas para todos los otros campos en posteriores versiones de la aplicación.
condición	Puede se mayor o menor e indica que la alerta será ejecutada una vez que el dato del campo indicado en campo sea mayor o menor al monto indicado en el campo monto.
email	Booleano que indica si en caso de ejecutarse la alerta se le avisará, o no, al usuario mediante un correo electrónico, en esta primera versión esta opción no está disponible.
fiat	Código ISO 4217 de la divisa en que se encuentra el campo monto
moneda	Símbolo de la criptomoneda que el usuario desea comprobar con la alerta.
monto	Dato numérico que se comparará con el campo indicado en campo y que se validará con la condición indicada en condición

Se ha reservado la siguiente estructura para las alertas de tipo porcentaje que se incorporarán en una próxima versión de la aplicación:

Ruta /usuarios/{userId}/alertas_**porcentaje/{alertId}/**	
activa	Indica si la alerta esta marcada como activada o no. Si esta activada se le mandará la notificación al usuario en caso de que se cumplan las condiciones de la alerta. Al ejecutarse la alerta quedará marcada como desactivada.

campo	En esta primera versión solo es posible poder poner una alerta para el campo precio de las criptomonedas, como se explica mas adelante en la sección de alertas, Se deja preparado este campo porque se espera poder implementar alertas para todos los otros campos en posteriores versiones de la aplicación.
email	Booleano que indica si en caso de ejecutarse la alerta se le avisará, o no, al usuario mediante un correo electrónico, en esta primera versión esta opción no está disponible.
periodo	Periodo en que el porcentaje del campo indicado en campo ha variado en un porcentaje igual o mayor al indicado en el campo porcentaje. Dicho periodo puede ser 1h (una hora), 24h (un día), 7d (una semana) o 30d (un mes)
moneda	Símbolo de la criptomoneda que el usuario desea comprobar con la alerta.
porcentaje	Porcentaje de variación que el usuario desea verificar en caso de que se alcance en el campo y periodo indicados

Ruta /usuarios/{userId}/exchanges**/Cex.io/**	
key	Dato brindado por el exchange Cex.io y necesario para que los usuarios puedan concertarse a sus cuentas mediante API.
secret	Dato brindado por el exchange Cex.io y necesario para que los usuarios puedan concertarse a sus cuentas mediante API.
userID	Dato brindado por el exchange Cex.io y necesario para que los usuarios puedan concertarse a sus

	cuentas mediante API.

Ruta /usuarios/{userId}/exchanges/Cex.io/ordenes_programadas//{idOrden}/	
cant1	Cantidad de unidades a comprar del activo indicado en el campo comprar
cant2	Cantidad de unidades del activo indicado en el campo con que se usarán para comprar las unidades Cant1 del activo indicado en el campo comprar
comprar	Símbolo del activo que se desea comprar
con	Símbolo del activo que se desea usar para comprar el activo indicado en el campo comprar
precio	Valor de una unidad del activo indicado en el campo comprar expresado en el activo indicado en el campo con
montoOk	Porcentaje de variación que el usuario desea verificar en caso de que se alcance en el campo y periodo indicados
precioOK	Precio (desde el punto de vista del exchange) de la operación
symbol1Ok	Símbolo dominante (desde el punto de vista del exchange) del par con el que se desea operar en el exchange
symbol1Ok	Símbolo segundario (desde el punto de vista del exchange) del par con el que se desea operar ene l exchange
tipoOk	Tipo de operación (desde el punto de vista del exchange) que puede ser buy o sell en el caso del exchange Cex.io.

En esta última colección de documentos se ha decidido agregar los campos según la aplicación criptomonedas.app desea homogenizar la manera de mostrar las operaciones de los exchanges y además los campos según el exchange trata las operaciones. Esto se ha realizado de esta manera por el hecho de que en la implementación he decido mostrar las operaciones de compra y venta de la manera tradicional y no mediante pares como lo hacen muchos o la mayoría de los exchanges.

En general los exchange trabajan por pares (ej. BTC/USD) donde el primer símbolo es el dominante y el segundo el secundario, los precios los muestran en la moneda dominante y la operación es de compra o de venta a partir de esta moneda. Personalmente considero que esta forma confunde bastante y me parece mejor ver el precio respecto al activo que se desea utilizar para comprar, sea cual sea (explicado mejor en la sección de desarrollo correspondiente a la pantalla programar orden).

Por este motivo igualmente e preferido poner los dos tipos de metodologías en la base de datos para que cuando el *back-end* consulte los documentos no tenga que volver a realizar cuantas de homogenización. O por si en versiones posteriores de Criptomonedas.app se tenga que repensar la manera de mostrar las ordenes, que no haya que realizar todas las cuentas nuevamente.

Por otro lado, en caso de que en un futuro se incorpore el soporte para poder interactuar con mas exchange, los mismos se incorporarán bajo el siguiente formato de ruta:

/usuarios/{userId}/exchanges**/{nombreExchange**}/

Cada uno de estos documentos contendrá los campos necesarios para que los usuarios puedan autentificarse mediante api a cada exchange correspondiente.

Al haber seleccionado una metodología RAD de desarrollo, como se indica en la sección 1.3.4, se ha dejado el código fuente parcialmente preparado para poder incluir nuevos exchange a partir de la obtención de su información de una base de datos (explicado más adelante en la sección correspondiente a desarrollo). En este primer momento los datos necesarios para conectarse a CEX.io se incluyen en el código fuente de la aplicación por ser un solo exchange, pero en versiones posteriores se obtendrán directamente desde Firestore.

3.1.4 Apis

Para la realización de la aplicación se han usado 3 APIS diferentes.

- Una Api correspondiente al exchange CEX.io [16]

- Una Api para la obtención general del mercado de criptomonedas que se usa en la sección para usuarios no autentificado.

- Una Api para la obtención de conversión de divisas

El api de Cex.io se utiliza para interactuar con las cuentas de los usuarios. Permite obtener el saldo de cada criptomoneda que el usuario tiene en dicho exchange, permite consultar el histórico de ordenes ejecutadas, las ordenes abiertas y los pares de criptomonedas que el exchange permite que los usuarios puedan operar.

El api general se ha usado el api de CoinMarketCap.com [17] permite obtener todos los datos relativos al mercado actual de las criptomonedas. Se usa la opción gratuita, con restricciones, a través de un servidor externo que actualiza los datos cada una hora. Esto quiere decir que los datos de la sección general libre de autentificación se actualizan cada una hora, lo que me parece aceptable por tratarse de una versión gratuita de la aplicación. Además, que al usar un servidor externo al api permite poder cambiar de proveedor de información o acortar los tiempos de actualización de información sin que los usuarios tengan que actualizar la versión de la aplicación.

Por ultimo se utiliza el api de https://free.currencyconverterapi.com/ para la obtención de las cotizaciones de las diferentes divisas tipo dinero FIAT [18]. Al igual que el api anterior se utiliza a través de un servidor intermedio que permite poder cambiar de api en cualquier momento sin que los usuarios tengan que actualizar la aplicación. La información proveniente de este api se actualiza cada 12 horas.

3.1.5 Back end

Figura 44: Python Software Foundation

En cuanto al *back-end* he avaluado varias posibilidades antes de decidirme por Python. Al principio pensaba realizarlo en PHP ya que había encontrado un desarrollo para conectarse a la API del exchange CEX.io en Github.com, pero en etapa de implementación he descubierto que Firebase no posee un SDK de *Firebase Admin* [12] oficial para dicho lenguaje (si bien he visto que existe una implementación para PHP no oficial, preferí usar una oficial).

Básicamente lo que necesitaba del *back-end* era, como se explicara mas adelante poder acceder a la base de datos de *Firestore* como administrador y poder enviar mensajes push a los usuarios que así lo soliciten. En este sentido me quedaban dos posibilidades, seleccionar un lenguaje que tuviera un SDK oficial de *Firebase Admin* [12] o utilizar *Cloud Functions for Firebase* [13]. Mi primera intención ha sido usar *Cloud Functions for Firebase* ya que me entusiasmaba probarlo, porque tiene una seguridad muy buena y se complementa perfectamente con *Firebase Firestore* y *Firebase Auth.* Pero requiere utilizar un el plan *Blaze* o de facturación de Firebase, y el plan que utilizo es *Spark* [14] que es totalmente gratuito.

Viendo los lenguajes que poseen un SDK oficial me quedaba la opción de Node.js o Python que son lenguajes que ya poseo un conocimiento previo como par poder usarlos con el fin que requería, pero me he decantado por Python ya que considero que junto a Cron es mucho más sencillo de configurar y usar que Node.js, ya que Node.js requiere de un servidor web como Express.

Al tratarse de un *back-end* que no involucra el desarrollo de una aplicación móvil en si misma, he intentado hacerlo lo más rápido y sencillo posible como para que la aplicación quede funcionando. Cumple perfectamente su función de dar soporte a la aplicación ejecutando las funciones pertinentes como para cumplir con el requerimiento funcional **RFi_006** y los requerimientos funcionales deseados **RFd_002 y RFd_003**.

Adjunto el código de Python 3.7.6.

3.3 Desarrollo

Observaciones generales:

La aplicación al iniciarse aprovecha la Splash Screen para cargar los datos iniciales y que luego la experiencia de usuario sea fluida. Me ha parecido mejor hacer esperar un segundo al usuario al inicio (o si tiene una mala conexión un poco más), pero que luego puede usar la aplicación de manera fluida.

Los precios de las criptomonedas en la parte de usuarios no autentificados y alertas se actualizan cada una hora ya que para el TFM utilizo la versión gratuita del API. Al no traer los datos directos del API sino de un dominio intermedio, puedo cambiar este comportamiento sin necesidad de que los usuarios finales actualicen la aplicación.

La aplicación solo puede iniciarse mientras el dispositivo posea conexión a internet. Como depende en su totalidad de datos provenientes de internet que deben estar actualizados, si el usuario intenta abrir la aplicación son conexión se le avisará y brindará la posibilidad de volver a intentarlo. Luego si al estar utilizando la aplicación el usuario pierde la conexión no podrá seguir utilizándola, le aparecerá una pantalla avisándole de la situación y no se le irá hasta recuperar la conexión.

Como se ha desarrollado la versión Android (requerimiento no funcional RnF_003), la aplicación no contempla el botón back ya que en Android dicha acción se realiza a través de los dispositivos, no de la UI de las aplicaciones.

3.3.1 Pantalla por pantalla

Splash Screen

Esta pantalla es la que el usuario ve primero al iniciar la aplicación. Se usa para traer los datos básicos necesarios para el buen funcionamiento de la aplicación y que, de no cargarse aquí, el usuario tendría que esperar este mismo segundo mientras navega la aplicación. Ya que las pantallas de inicio, tanto la Home general como la Home de usuarios autentificados muestran datos actuales traídos de internet.

Figura 45: Splash screen

La sensación de estar en pantallas y no poder usarlas inmediatamente en mi opinión perjudican mucho la experiencia de usuario mientras que esta solución me ha parecido amigable y que será agradecida por parte de los usuarios. He contemplado traer los datos de manera asíncrona, y de hecho de esta manera lo hacia desde el principio, pero la naturaleza de la aplicación requiere que los datos ya estén apenas se cargan las pantallas.

Home:

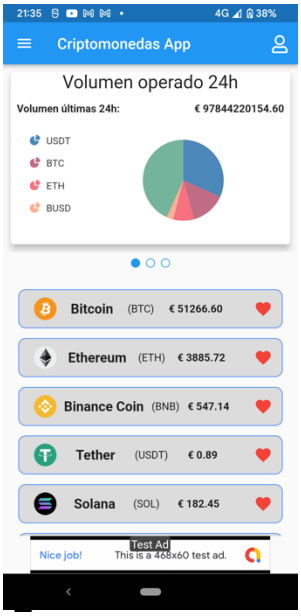


Figura 46: Home

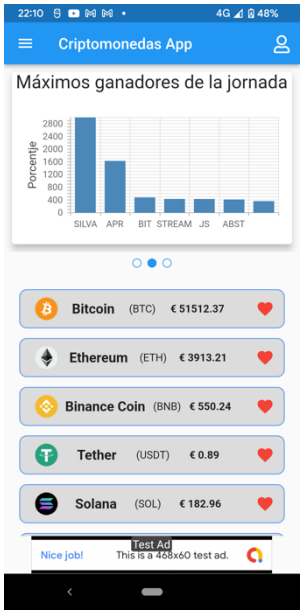


Figura 47: Home

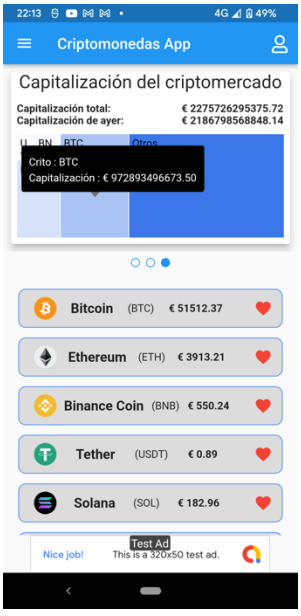


Figura 48: Home

Es la pantalla de inicio de los usuarios no identificados. Los usuarios identificados pueden acceder a través del menú de navegación.

En esta pantalla se muestra un resumen general del mercado actual de las criptomonedas (Volumen operado ultimas 24 horas, criptomonedas con mayores ganancias en las últimas 24 horas y una grafica con la capitalización total del criptomercado y la porción equivalente de las criptomonedas con mayor capitalización respecto al total.) y las criptomonedas que el usuario tenga marcadas como favoritas (requerimiento funcional importante RFi_003).

En este caso las criptomonedas favoritas se guardan en los dispositivos por lo que no se ha implementado la persistencia multidispositivo para usuarios autentificados. Se puede hacer fácilmente como se ha hecho con la sección privada, pero por cuestiones de tiempo queda para una próxima versión.

Por defecto si el usuario no tiene criptomonedas marcadas como favoritas se le cargan las top 10 como favoritas automáticamente. Esto me a parecido interesante para que el usuario no vea la pantalla vacía. Si el usuario quita todas las criptomonedas favoritas le aparece un cartel que le avisa que no tiene más favoritas (requerimiento no funcional RnF_002) y le da la opción de volver a cargar el top ten como favoritas. Aunque no lo hiciera, si deja la aplicación sin favoritas, la próxima vez que abra la aplicación le volverán a parecer el top ten como favoritas. Este comportamiento podría cambiarse si una vez subida la aplicación en las App Store se pude tomar conocimiento de que a los usuarios les molesta.

Al seleccionar cualquier activo se redirige a la pantalla de detalle de criptomoneda correspondiente.

Detalle de criptomoneda:

Es la pantalla donde se pueden ver todos los datos correspondientes a una criptomoneda en detalle. Falta agregar los gráficos de datos históricos, pero al ser una aplicación gratuita, y al no encontrar una API gratuita para la obtención de datos, no lo he podido implementar para esta primera versión. Ya estoy recopilando información con un spider para poder contar con esta información sin necesidad de contratar el servicio de terceros. Lo implementaré en una próxima versión.

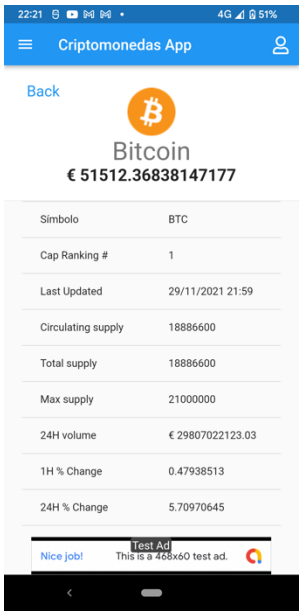


Figura 49: Detalle cripto

A esta pantalla se puede acceder desde varios lugares de la aplicación, sobre todo al seleccionar una criptomoneda de los listados de la sección pública. Pero, además, al recibir una alerta de variación de precio, si el usuario tiene la aplicación en primer plano le aparecerá un mensaje con la opción de acceder a esta página de detalle, correspondiente a dicha

moneda. Y si el usuario tiene la aplicación terminada o suspendida, le llagará la alerta correspondiente y al pulsarla, se le abrirá la aplicación y lo traerá a esta pantalla.

En la sección privada no he implementado el acceso a esta sección ya que los precios pueden variar de un exchange a otro.

Top ten:

Esta pantalla es un listado rápido del top ten de criptomonedas, En principio al entrar aparecen ordenadas por el campo capitalización, pero luego el usuario puede optar por ver el top ten según mayor precio, según menor precio, y según las mayores ganancias en las últimas 24 horas.

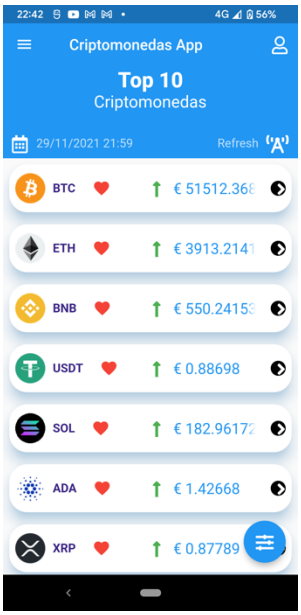


Figura 50: Top ten

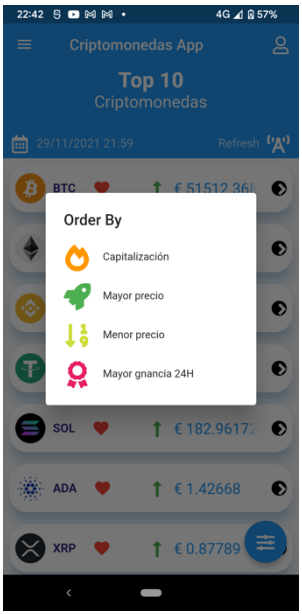


Figura 51: Filtrando Top ten

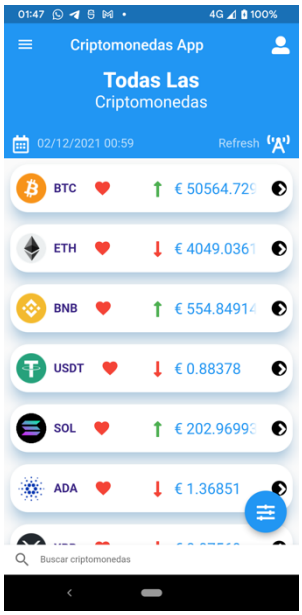


Figura 52: Todas las cripto

Todas las criptomonedas:

Es esta pantalla se pueden ver todas las criptomonedas (Requerimientos funcionales importantes RFi_001 y RFi_002). A diferencia de la pantalla Top ten, tiene un buscador para filtrar por nombre de símbolo las criptomonedas que se muestran.

Cripto conversor:

En principio había decidido soportar la conversión solo entre el top 10 de criptomonedas según su capitalización. Pero a ultimo momento he pensado que de la manera en que se había diseñado era muy fácil poder implementar la conversión entre cualquier criptomoneda de las soportadas en la aplicación.

Al realizar el cambio, solo debía cambiar la lista fuente del controlador del conversor, pero no. He cambiado también los componentes dropdownButton por componentes Autocomplete. Si la lista contiene muchas

opciones el componente dropdownButton pierde utilidad porque hay que buscar a ojo e ir muy abajo para poder acceder a algunas criptomonedas. Perdía el sentido de soportar todas las criptomonedas porque en la practica era inutilizable. Ahora con el Autocomplete quedan todas las criptomonedas accesibles. Pero el único inconveniente es que al ser un cambio de último momento debí sacrificar el botón para dar vuelta las criptomonedas involucradas en la conversión. La conversión se realizaba, pero los autocomplete no intercambiaban el contenido y quedaba confuso. No hay una manera rápida de intercambiar el contenido de los Autocomplete por lo que he dejado comentado el IconButton para implementarlo con un poco más de tiempo. He preferido soportar la conversión entre 5000 criptomonedas desde un comienzo a soportar solo 10 con un botón para invertir la conversión. En una segunda versión el botón quedará implementado pero los usuarios desde un comienzo sabrán que en Criptomonedas.app pueden realizar conversiones entre 5000 criptomonedas.

En esta pantalla he decidido que la conversión sea automática al momento en que el usuario cambie cualquier dato del formulario, podría haber puesto un botón, pero he considerado que en dispositivos móviles reducir un paso para realizar cualquier es un beneficioso para todos los usuarios.

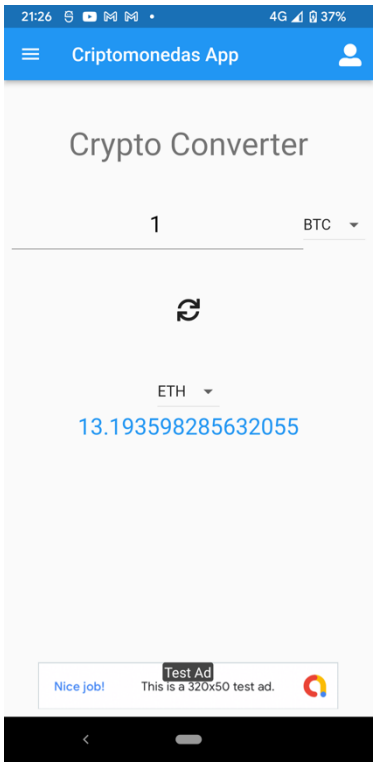


Figura 54: Cipto conversor primera versión

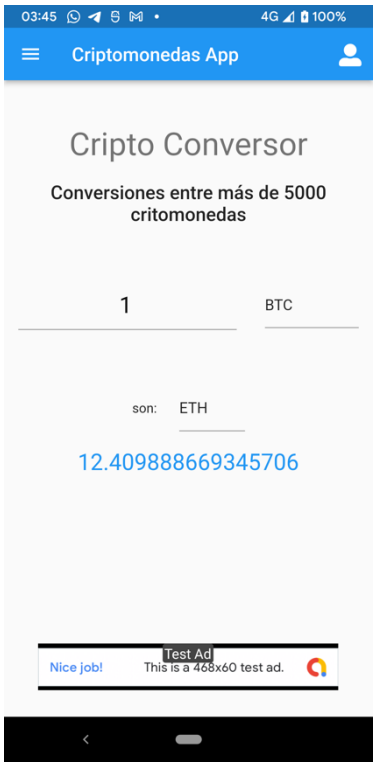


Figura 53: Cipto conversor final

Cripto a Fiat:

Esta pantalla permite convertir el precio de cualquier criptomoneda (de las soportadas por la aplicación) a cualquier activo de tipo FIAT. Solo que entre las opciones aparecen las criptomonedas marcadas como favoritas por el

usuario, para poder usar el criptoconversor con más monedas simplemente hay que agregarlas a favoritos. He tomado esta decisión porque considero que es un incentivo para que el usuario tenga monedas en favoritos. No solo implica ver el precio en la pantalla principal sino la posibilidad de poder convertir a Fiat. Siempre se puede cambiar esta lógica si los usuarios lo manifiestan como una crítica vez se suba la aplicación en las App store.

Esta pantalla permite calcular no solo el precio unitario, sino de cualquier número de unidades.

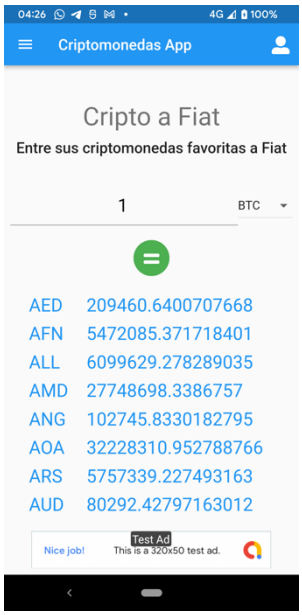


Figura 55: Cripto a fiat

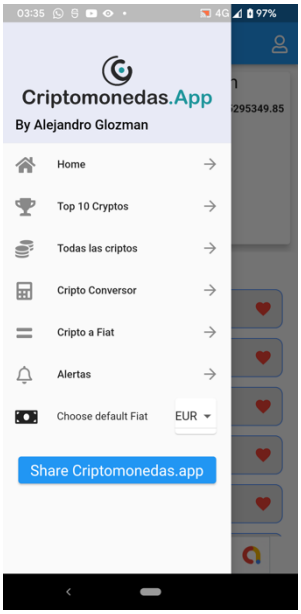


Figura 56: Menú principal

El menú de navegación principal:

El menú de navegación permite navegar entre las diferentes pantallas de acceso libre, sin estar autentificado. Pero además permite cambiar la moneda por defecto en que se muestran todos los precios de la aplicación y permite compartir la aplicación con otras personas. En cuanto a la moneda en que s muestran los precios de la aplicación he construido la aplicación de tal manera que dicho cambio sea instantáneo. Apenas el usuario cambia la moneda en el componente Dropdown automáticamente se realiza la conversión.

A este menú se accede desde cualquier pantalla pulsando en el ícono, denominado coloquialmente menú hamburguesa, que se encuentra arriba a la izquierda en el AppBar.

Además de las secciones anteriores, a través de este menú se puede acceder a las alertas, un servicio restringido a usuarios identificados. Se incluye en este menú porque así se pensó en la etapa de diseño. La idea es que los usuarios identificados puedan acceder rápidamente y los no autentificados sepan que existe el servicio. Creo que es un servicio muy bueno que ayuda a fideliza a los usuarios.

Autentificación:

A esta pantalla se accede desde el AppBar presionando sobre el ícono de persona que se encuentra arriba a la derecha y siempre y cuando el usuario no se encuentre identificado. Si el usuario no se encuentra autentificado, también se accede al presionar alertas del menú de navegación principal. También se accede luego de que un usuario autentificado cierra sesión.

Como se ha comentado anteriormente queda implementado los métodos para autentificarse con usuario y contraseña, FacebookAuth y GoogleSignIn. Sin embargo, he dejado comentado los dos primeros métodos mencionados para entregar el TFM ya que Facebook pide revisión de la aplicación para que se pueda usar su método sin ser el usuario desarrollador, y para confirmar el mail del usuario y contraseña tenía que configurar un servidor SMTP que tiene un costo. Pero he dejado el método de GoogleSignIn que considero es muy extendido y deja que cualquier persona pueda identificarse y usar la aplicación sin complicaciones, incluso si la aplicación aún no se ha subido a las AppStore.

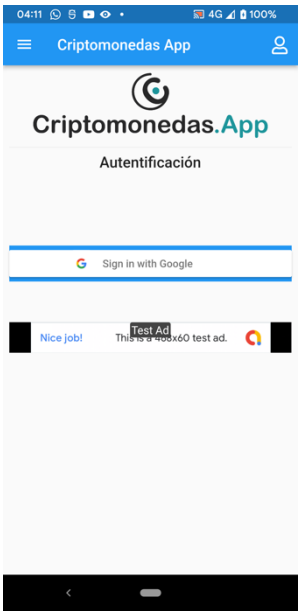


Figura 57: Autentificación

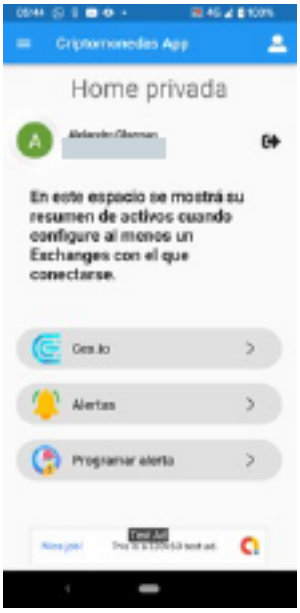


Figura 58: Home autentificados

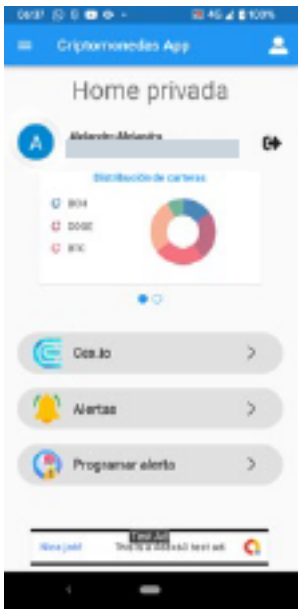


Figura 59: Home autentificados

Home de usuarios autentificados:

Una vez que la autentificación es exitosa se redirige automáticamente a la pantalla Home de usuarios autentificados. A esta pantalla se accede además desde el AppBar presionando sobre el ícono de persona que se encuentra arriba a la derecha y siempre y cuando el usuario ya se encuentre autentificado. Pero también se accede al iniciar la aplicación siempre y cuando la sesión del usuario aún se encuentre activa. Nótese que cuando el usuario se encuentra autentificado el ícono de persona del AppBar se encuentra coloreado y cuando no lo está se encuentra contorneado.

En la pantalla Home de la sección privada en primer lugar aparecen los datos básicos del usuario y la posibilidad de cerrar la sesión. Cabe destacar

que desde el punto de vista del desarrollo se cierra sesión tanto en GoogleSignIn como en FirebaseAuth. Podría cerrar sesión solo con FirebaseAuth pero en ese caso, el usuario al querer volver autentificarse en el mismo dispositivo, no podría cambiar de cuenta de Google, ya que en Google seguiría autentificado y se autentificaría con Firebase directamente si tener que volver a seleccionar en Google la cuenta con la que se desea ingresar.

Luego aparece un resumen de la cuenta (figura 58). En principio aparece solo la información del Exchange Cex.io siempre y cuando el usuario haya ingresado las credenciales correctas para interactuar con su cuenta. La idea es que aquí haya un resumen de todas las cuentas del usuario de los exchanges que la aplicación irá soportando. Si el usuario no ha ingresado las credenciales de Cex.io, se le informa que ahí se le mostrará un resumen de sus cuentas apenas ingrese dichas credenciales (figura 57) como el requerimiento no funcional RnF_002 lo indica.

.

Un poco más abajo aparece la botonera principal de la Home de usuarios autentificados. Se puede ir al espacio dedicado al exchange Cex.io, al espacio dedicado a las Alertas y a programar una alerta. Como se explica en la sección Dar soporte a más exchanges, queda todo bastante preparado para que implementar nuevos exchanges sea casi tan rápido como adaptar sus api a la estandarización de Criptomonedas.app.

Por otra parte, en el código de la pantalla de home de usuarios autentificados es cuando se configura todo lo necesario para que los usuarios puedan recibir las notificaciones. Se ha configurado aquí porque tal y como está diseñada la aplicación solo los usuarios identificados pueden recibir notificaciones.

Ingresar credenciales de exchange:

Esta pantalla aparece cuando un usuario autentificado quiere entrar a la sección de un exchange en particular y aún no ha ingresado las credenciales. Básicamente es la manera en que los usuarios brindan la información necesaria para que Criptomonedas.app pueda tener acceso a sus cuentas a través del API correspondiente. Esta pantalla también sirve para que los usuarios puedan cambiar credenciales ya cargadas, ya que la manera de hacerlo es eliminar las credenciales existentes y posteriormente cargarlas de cero en esta pantalla. Para saber como generar las credenciales de Cex.io ver anexo correspondiente.

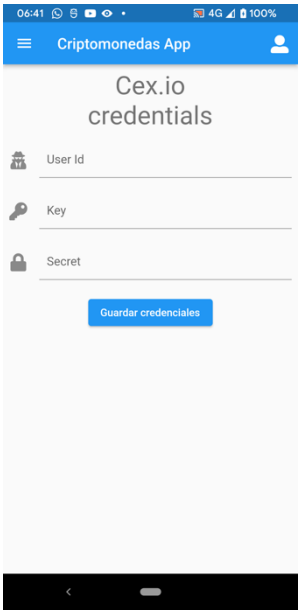


Figura 60: Cargar credenciales

Home de exchange:

El home de exchange corresponde a la pantalla principal de un exchange particular en Criptomonedas.app (en esta primera versión solo existe la de Cex.io). Primero se puede ver en forma de carrusel un resumen de como se compone la Criptocartera del usuario en dicho exchange. Se muestra una gráfica que indica el porcentaje de cada criptomoneda de su Criptocartera respecto al total, siempre tomando el valor actual de cada criptomoneda en relación con el valor actual de todo el porfolio, todos convertidos a la moneda Fiat seleccionada por el usuario como principal en el menú de navegación. Además, abajo se muestra el valor total del porfolio en su equivalente a la moneda Fiat seleccionada por el usuario en el menú principal.

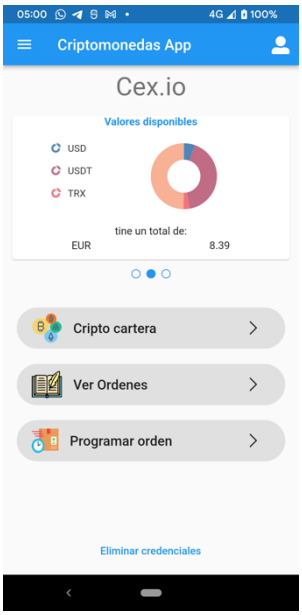


Figura 61: Home de exchange

El carrusel tiene tres secciones en esta primera versión. Todas iguales en cuanto a estética y lógica, pero representan diferentes cosas. La primera es sobre el total del porfolio, la segunda sobre los activos disponibles del porfolio y la tercera sobre los activos bloqueados del porfolio ya que se encuentran reservados en ordenes abiertas.

Por debajo del carrusel se muestra el menú de navegación del exchange y luego un link que le permite al usuario eliminar las credenciales cargadas para este exchange (requerimiento funcional importante RFi_008).

Cripto cartera:

Figura 62: Criptocartera

Figura 63: Criptocartera pool to refresh

Figura 64: Criptocartera secciones

Esta pantalla corresponde con el requerimiento funcional importante RFi_004. Permite ver las criptomonedas que componen el portafolio del usuario en el Exchange. Pero además permite ver su valor correspondiente en la moneda Fiat seleccionada por el usuario tanto de cada criptomoneda como del total. Permite mostrar todos los activos que posee el exchange o solo los que el usuario tiene un saldo mayor a cero. Luego permite ver solo lo disponible o solo lo reservado en ordenes.

Es importante recalcar que, al seleccionar mostrar solo con saldo, corresponde a las criptomonedas con saldo superior a cero en total, por eso que en las secciones de Disponible o en Ordenes pueden aparecer saldos en 0. Se ha hecho de esta manera para remarcar que de esas monedas tiene saldo, pero quizá no disponible o quizá no en ordenes. Es una manera más de visualizar información de una forma rápida y dinámica.

Estos listados permiten hacer pull to refresh para actualizar la información. Es útil por ejemplo en el caso de que el usuario se encuentre en esta pantalla y reciba una alerta de que una orden programada se haya dado de alta o también si el usuario esta hace mucho tiempo en la pantalla y desea actualizar la valorización actual de sus activos respecto a la moneda Fiat seleccionada.

Ordenes:

Se accede a esta pantalla a través de la pantalla principal de la Home del Exchange, pulsando en Ver ordenes. Es una pantalla que corresponde con los requerimientos funcionales importantes RFi_004, RFi_005 y RFi_007

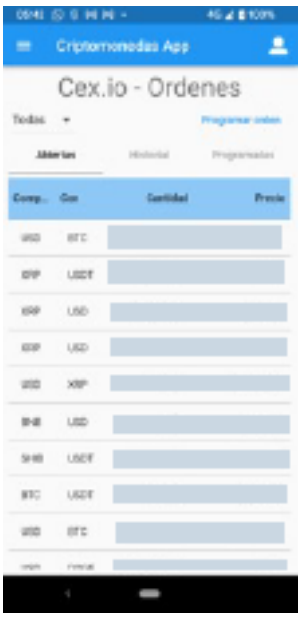


Figura 65: Ordenes abiertas

Figura 66: Eliminar orden abierta

Figura 67: Filtar ordenes

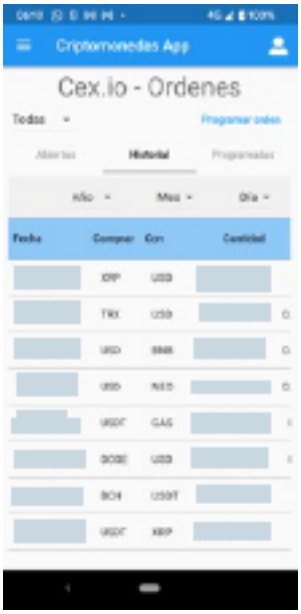


Figura 68: Ordenes històricas

Figura 69: Ordenes programadas

Mediante esta pantalla, se puede listar y eliminar ordenes abiertas en el exchange, listar ordenes históricas ya ejecutadas en el exchange, y listar y eliminar ordenes programadas en el sistema de Criptomonedas.app para ser abiertas en el exchange apenas el usuario cuente de manera disponible con la cantidad del activo que haya configurado como el elemento utilizado para comprar el activo que se desea obtener al ejecutarse la orden programada.

Las ordenes históricas no se pueden eliminar ya que son ordenes ya ejecutadas, un echo irreversible. Pero al tener fecha de ejecución, se pueden filtrar por año, mes y día, en ese orden (figura 67). Una vez puesto el año se puede seleccionar mes y luego día, pero si se ha seleccionado por ejemplo solo un día particular, al filtrar por año se borra la selección del día, se reinicia el filtro. Me ha parecido interesante hacerlo de esta manera como un filtro descendiente que puede ir aumentando en granularidad, otra vez es una decisión que corresponde a un gusto personal y que si a los usuarios les pareciera mejor realizarlo de otra manera se podría cambiar.

Tanto para eliminar ordenes abiertas como programadas (figura 65 y 68), hay que deslizar la fila correspondiente de la orden abierta o programada para la izquierda o derecha y luego apretar el botón de eliminar. Además, las ordenes programadas se actualizan en tiempo real, si el usuario se encuentra en ese listado y una orden programada se elimina en otro dispositivo o se abre en el exchange por el back end automatizado, la orden programa desaparecerá del listado automáticamente. Todas las ordenes, de los tres tipos, pueden filtrarse por criptomonedas por medio del Widget DropdownButton (figura 66)

De esta pantalla al igual que de le Home del exchange se puede acceder directamente a la pantalla para dar de alta ordenes programadas.

Ordenes programadas:

Es la pantalla que corresponde con los requerimientos funcionales importantes RFi_005 y RFi_006.

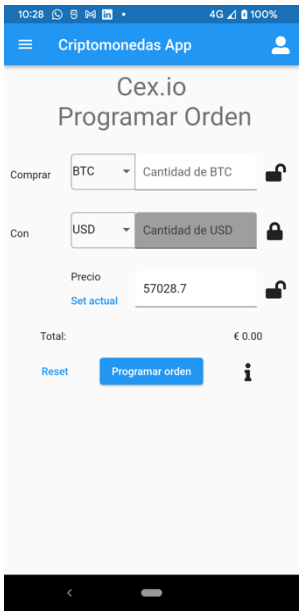


Figura 70: programar ordenes

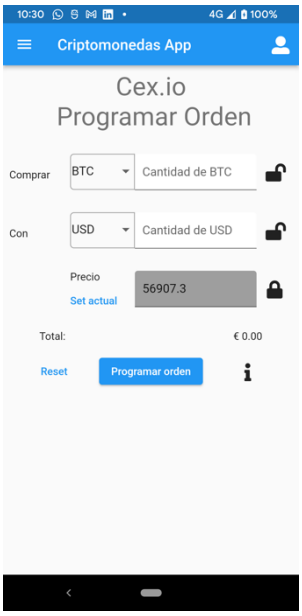


Figura 71: Programar ordenes

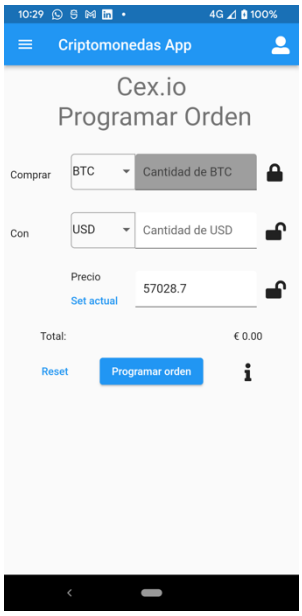


Figura 72: Programar ordenes

En esta pantalla el usuario debe seleccionar primero el activo que desea comprar. Una vez seleccionado el activo a comprar se actualiza el DropdownButton correspondiente a los activos con los que se puede comparar el activo deseado. De esta manera el usuario va a poder seleccionar el activo con el que desea comprar únicamente entre los activos que el exchange permite operar con el activo deseado a comprar. Luego hay tres campos de los cuales el usuario debe rellenar dos y el tercero se calcula automáticamente. Estos son la cantidad por comprar, la cantidad usada para comprar y el precio unitario de la operación en el activo que se utiliza para comprar. Es básicamente una regla de tres simple de la cual solo se necesitan dos datos de entrada ya que el precio vine dado por el exchange.

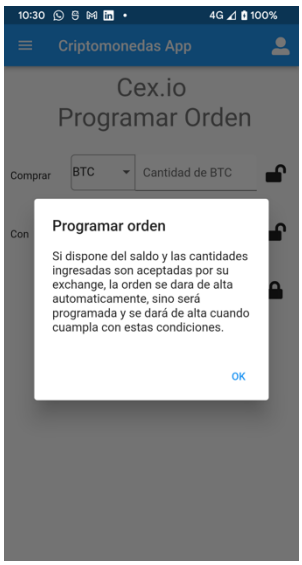


Figura 73: Info programar ordenes

El usuario podrá decidir cuales son los dos campos que desea rellenar, si la cantidad a comprar, la cantidad con la cual comprar o el precio unitario. Se comunica al usuario de esta situación mediante los iconos de candado y el color de los TextField. Los TextField habilitados son los blancos y el bloqueado gris. El usuario puede pulsar los diferentes candados para cambiar los campos que desea introducir y el que se calculará solo. Este funcionamiento siempre prioriza la entrada de la cantidad a comprar por sobre la de cantidad con la cual

comprar, si se aprieta el candado de precio unitario cuando esta cerrado se abre y se cierra el de cantidad con la cual comprar. Una vez los dos datos a completar están completos, además de calcularse el tercer elemento de la ecuación, se obtiene el valor total de la operación equivalente en la moneda Fiat que el usuario tenga configurada como principal en el menú de navegación. Si quiere ver el total en otra moneda solo debe cambiar la moneda principal en el menú de navegación y el cambie se realiza automáticamente. Si el campo de precio unitario se encuentra habilitado, el usuario puede en cualquier momento seleccionar automáticamente como precio unitario el precio actual de la moneda a comprar en la moneda con la cual comprar.

Por ultimo el usuario puede reiniciar el formulario y dejar todos los campos en blanco y puede pulsar en la i de información para leer que la orden se dará de alta automáticamente si dispone de saldo necesario o si no lo dispone la orden quedará programada.

Siempre de cualquier pantalla, si el usuario se encuentra autentificado puede volver a la pantalla principal de usuarios autentificados pulsando el ícono de persona que hay en la parte superior derecha.

Alertas:

Desde el home principal de usuarios autentificados se puede acceder a la pantalla alerta, al igual que del menú principal de navegación como se explico anteriormente en la sección correspondiente al menú de navegación.

En esta pantalla se pueden listar las alertas que se hayan programado, ver si están activas o desactivadas, se pueden activar o desactivar y hasta eliminarlas. Esta pantalla junto a la de alta de alertas corresponde a los requerimientos funcionales deseables RFd_002 y RFd_003.

Figura 74: Alertas

Figura 75: Alertas activas

Figura 76: Alertas desactivadas

En la pantalla alertas se puede elegir entre ver todas las alertas, solo las activas o solo las desactivadas. Todas se actualizan en tiempo real sin necesidad de la interacción del usuario. Si el usuario se encuentra en alertas activas y desactiva una, la misma desaparece de dicho listado automáticamente. Si el usuario se encuentra en viendo todas las alertas y una alerta activa es ejecutada por el *back-end* va a ver como en su dispositivo la alerta pasa de activa a desactivada, si el usuario se autentifica en dos dispositivos diferentes y en los dos dispositivos se encuentra viendo todas las alertas y elimina una alerta en un dispositivo, va a ver como se elimina la misma alerta en el otro dispositivo en tiempo real. Incluso si el usuario en un dispositivo se encuentra visualizando las alertas y en otro da de alta una nueva, va a ver como la nueva alerta le aparece automáticamente en el listado de todas las alertas o en el de alertas activas.

Figura 77: Eliminar alerta

Presionando en el ícono de campanas el usuario puede activar o desactivar la alerta correspondiente. Deslizando la fila correspondiente para la izquierda o derecha en una alerta y presionando en el ícono de cesto de basura, elimina automáticamente dicha alerta.

El funcionamiento de las alertas es muy sencillo, el usuario las da de alta directamente en estado activa. El *back-end* cada cierto tiempo verifica si la condición de la alerta se cumple y si es así la desactiva y le envía la alerta al usuario. Luego el usuario si lo desea puede volver a activarla o eliminarla. El resto se explica mejor en la sección sobre las notificaciones.

Programar alerta:

Pantalla que se accede desde el home principal del usuario autentificado y desde la pantalla de alertas. En esta pantalla el usuario selecciona una moneda, un monto y si la alerta se debe ejecutar cuando el precio de la moneda seleccionada sea mayor o menor al monto configurado.

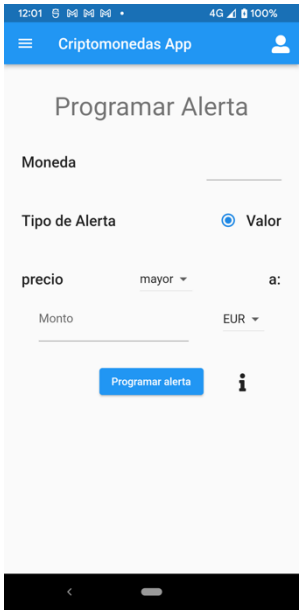


Figura 78: Programar alerta

Cumpliendo con el requerimiento no funcional RnF_002, al pulsar el ícono con la i de información se le mostrara un mensaje explicando el funcionamiento.

Si el usuario desea dar de alta una alerta que ya existe una idéntica, se dirá que la alerta fue dada de alta exitosamente, pero la alerta no será duplicada en la base de datos. He tomado esta decisión porque considero que dar mensajes de error puede ser frustrante y en este caso al usuario lo que le importa

es saber que la alerta exista y se encuentre activa.

3.3.2 Sobre la seguridad

Si bien Criptomonedas.app permite realizar operaciones sensibles para las cuales se necesitan credenciales de exchanges donde los usuarios poseen activos valiosos, la aplicación Criptomonedas.app es seguro desde todo punto de vista teniendo en cuenta que las credenciales no permiten trasferir dinero, sino solo dar de alta ordenes, comprar y vender y eliminar ordenes dadas de alta. Cada vez que se realizan estas operaciones el usuario recibe una alerta que le permitiría poder revertir la operación al poner una orden de compra inversa inmediatamente.

Las credenciales de los usuarios se resguardan en Firestore bajo la seguridad administrada por Google y el back end de Criptomonedas.app no hace un resguardo de dichas credenciales, solo las peticiona en tiempo real bajo las peticiones encriptadas https cada vez que necesita usarlas. Por todo lo dicho se puede considerar a Criptomonedas.app una aplicación segura.

3.3.3 Dar soporte a más exchanges

Se ha dejado preparado el código para soportar la interacción con múltiples exchanges fácilmente. Se da soporte. Un exchange como me había propuesto dado los tiempos de desarrollo, pero la implementación actual contemple el hecho de agregar más exchanges y facilita mucho dichas incorporaciones.

Esto se puede observar en dos partes de la implementación. Primero en la parte de servicios: Existe el servicio del exchange Cex.io representado por la clase Cexlo que hereda de la clase abstracta ExchangeService. Luego en la parte del modelo existen las clases Exchange y ElExchange. La clase Exchange representa a un exchange que formará parte de una lista de exchanges que tendrá el usuario (por ahora representa solo al exchange Cex.io) y posee un atributo exchangeService de tipo ExchangeService (la clase abstracta) y que por ahora contendrá la clase Cexlo pero que luego podría tener cualquier servicio de cualquier otro exchange. La clase ElExchange por su parte, representa al exchange con el que el usuario podría encontrarse interactuando en un momento dado, nótese que esta clase posee un atributo elExchange de tipo Exchange y que es el que usan todas las pantallas para mostrar los datos.

De esta manera, implementar un nuevo exchange implicaría crear una clase que herede de ExchangeService y haga de intermediario entre la API de dicho exchange y Criptomonedas.app. y por el otro adaptar un poco el código para que al listar los exchanges disponibles y el resumen general del Usuario en sus exchange (que hay en la home del usuario registrado) soporte una lista de Exchanges en vez de uno solo. Luego cuando se quiera añadir un tercer Exchange solo hará falta crear la clase que herede de ExchangeService y haga de intermediario entre la API de dicho exchange y Criptomonedas.app.

3.3.4 Sobre las notificaciones

Los usuarios autentificados pueden recibir dos tipos diferentes de notificaciones, ambas de alta prioridad. Una relacionada a las alertas y otra a las ordenes programadas.

Cuando se da de alta una orden programada, dicha orden se elimina de la base de datos y se le notifica al usuario correspondiente en todos los dispositivos en los que pudo haber iniciado sesión. Si el usuario pulsa en la notificación se abre la aplicación en la sección de ordenes.

Cuando se dan las condiciones necesarias para ejecutar una alerta, la misma se marca como desactivada y se le envía una notificación al usuario correspondiente. Si el usuario pulsa en la notificación se abre la aplicación en la página de detallas de la criptomoneda que el usuario a elegido al dar de alta l alerta.

Estos procesos son efectuados por el back end como se ha explicado en la sección correspondiente. Actualmente se ha configurado un Crontab para que el programa Python del back end se ejecute cada dos minutos.

3.4 Pruebas

Se ha testeado y depurado la aplicación de manera iterativa durante el proceso de desarrollo. Luego, al finalizar lo he vuelto a testear manualmente. Entre otras pruebas he realizado las siguientes:

Tabla 29: Pruebas

Grupo	Descripción	Resultado
LogIn	Poder cerrar sesión y volver a autentificarse con otro usuario.	✅
Exchange	Poder eliminar credenciales del exchange	✅
Alertas	Dar de alta alertas y que, al darse la condición, se reciba la alerta en todos los dispositivos en que el usuario haya iniciado sesión y aún tenga la aplicación instalada. Tanto cuando la aplicación esta activa como terminada o dormida.	✅
Alertas	Poder activar y desactivar alertas	✅

Finalmente, también he podido automatizar pruebas unitarias de algunas de las funciones de la clase utils. Todas pasan de manera satisfactoria.

4. Conclusiones

Haciendo el TFM he aprendido a realizar una aplicación móvil completa y compleja con Flutter. Una aplicación con base de datos, *back-end* y envió de alertas. Además, pude usar Firebase e integrarlo tanto con la aplicación como con un *back-end* en Python. También pude aprender a enviar alertas a usuarios partículas según sus intereses particulares.

Queda para el futuro bastante trabajo. Queda convertir la aplicación en multilenguaje, poder reescribir el código realizado al comienzo con los conocimientos adquiridos en la etapa final de desarrollo. Y también, queda pendiente añadir más categorías de alertas y añadir soporte para más exchanges.

En cuanto a los logros de los objetivos planteados estoy más que conforme. Comencé el trabajo con mucho cuidado, pensando que lo que me había propuesto era todo un desafío. Tenía muchas cosas por aprender y poco tiempo. Sin embargo, era un proyecto que me entusiasmaba y me interesaba realizar y por eso valía la pena el riesgo. Poco a poco, dividiendo las tareas complejas en partes más simples considero que pude lograr mucho mas que un producto mínimo viable. He cumplido con los requerimientos no funcionales, los requerimientos funcionales importantes y algunos deseables como, entre otros, todo lo referente a las alertas (RFd_003). Quedo muy contento con el resultado obtenido.

5. Glosario

Bitcoin: Es la primera criptomoneda creada en 2008 por una persona desconocida que e hizo llamar Satoshi Nakamoto. Su primer precio fue registrado en 2010, donde su máximo alcanzo los 0,39 dólares estadounidenses. Hoy (19/10/21) su precio es de 55 mil euros y posee una capitalización de mercado de mil millones de euros.

Blockchain: Tecnología distribuida y descentralizada que hace de libro contable de las criptomonedas. Se basa en la criptografía para asegurar integridad y seguridad tanto de las transacciones como de todo el sistema que constituyen.

Criptoactivo: Activo que tiene su origen en la criptografía y que posee un valor de mercado que permite comprarlos, venderlos o intercambiarlos por otras mercancías.

Criptocartera: Uso este termino para referirme al total de criptomonedas que el usuario posee.

Criptoconversor: Herramienta que permite saber el precio de una coriptomoneda en relación con otra

Criptodivisa: Ver criptomoneda.

Criptomercado: Mercado económico donde se opera con criptoactivos.

Criptomoneda: La criptomoneda o criptodivisa es un tipo de moneda digital basada en criptografía y la tecnología blockchain. Pueden ser descentralizadas y transparentes en cuanto a su emisión y transferencia de fondos.

Cryptotrading: La acción repetitiva de comprar y vender criptomonedas para hacer diferencia entre la compra y la venta, como si de acciones bursátiles se tratara.

Dinero Fiat: Del latín fiat, 'hágase', es una forma de dinero fiduciario cuya cualidad de dinero proviene de su declaración por parte del Estado como tal

Exchange: Sitio web o empresa donde se puede comprar y vender criptomonedas, ya sea con dinero real o con otras criptomonedas.

Etherium: En la actualidad (19/10/21) es la criptomoneda más importante después de bitcoin, en cuanto a capitalización de mercado, cursos y seminarios para iniciados. Es la primer criptomoneda que ha permitido contratos inteligentes.

Log out: Acción que hace que un usuario identificado en un sistema informático deje de estar identificado independientemente de si sigue en el sistema o no.

Trading: Acción de comprar y vender ciptoactivos de manera reiterada con la finalidad de ganar dinero o mas criptoactivos con la diferencia de precios de cada compra y venta.

Hardware wallets: Dispositivo electrónico que guarda las criptomonedas de manara segura y offline. Al usar este dispositivo las criptomonedas las tiene el propio usuario, sin intermediarios o exchanges.

6. Bibliografía y créditos

Bibliografía:

- [1] https://www.elconfidencialdigital.com/articulo/otros/38-jovenes-espanoles-plantea-invertir-criptomonedas-rebellion/20210927133215279419.html, 18/10/2021
- Alessandria, Simone; Kayfitz, Brian, *Flutter Cookbook*, Packt, Birmingham, 2021.
- [2] thispersondoesnotexist.com, 18/10/2021
- [3] fakenamegenerator.com, 18/10/2021
- [4] Van Der Heyde, Fien; Debrauwer, Laurent, UML 2.5, Ediciones Eni, Madrid, 2020
- [5] https://firebase.google.com/docs/admin/setup, 18/11/2021
- [6] https://docs.flutter.dev/get-started/editor 18/11/2021
- [7] https://developer.android.com/about/versions/12, 25/11/2021
- [8] https://firebase.flutter.dev/docs/auth/overview, 25/11/2021
- [9] https://firebase.flutter.dev/docs/firestore/overview, 25/11/2021
- [10] https://firebase.flutter.dev/docs/messaging/overview, 25/11/2021
- [11] https://man7.org/linux/man-pages/man8/cron.8.html, 25/11/2021
- [12] https://firebase.google.com/docs/admin/setup, 25/11/2021
- [13] https://firebase.flutter.dev/docs/functions/overview, 25/11/2021
- [14] https://firebase.google.com/pricing, 25/11/2021
- [15] https://es.wikipedia.org/wiki/ISO_4217, 25/11/2021
- [16] https://cex.io/rest-api, 25/11/2021
- [17] https://coinmarketcap.com/api/, 25/11/2021
- [18] https://es.wikipedia.org/wiki/Dinero_por_decreto/, 25/11/2021
- [19] https://meedu.app/cursos, 01/11/2021
- [20] https://www.youtube.com/c/flutterdev, 01/11/2021
- [21] https://www.youtube.com/c/FernandoHerreraCr, 01/11/2021
- [22] https://docs.flutter.dev/deployment/android, 01/12/2021
- [23] https://www.youtube.com/c/RicardoMarkiewicz, 03/12/2021

Créditos:

Iconos de las figuras 58, 59 y 61 correspondientes a los botones “Alertas”, “Programar alertas”, “Criptocartera”, “Ver ordenes” y “programar ordenes” son de Flaticon.com

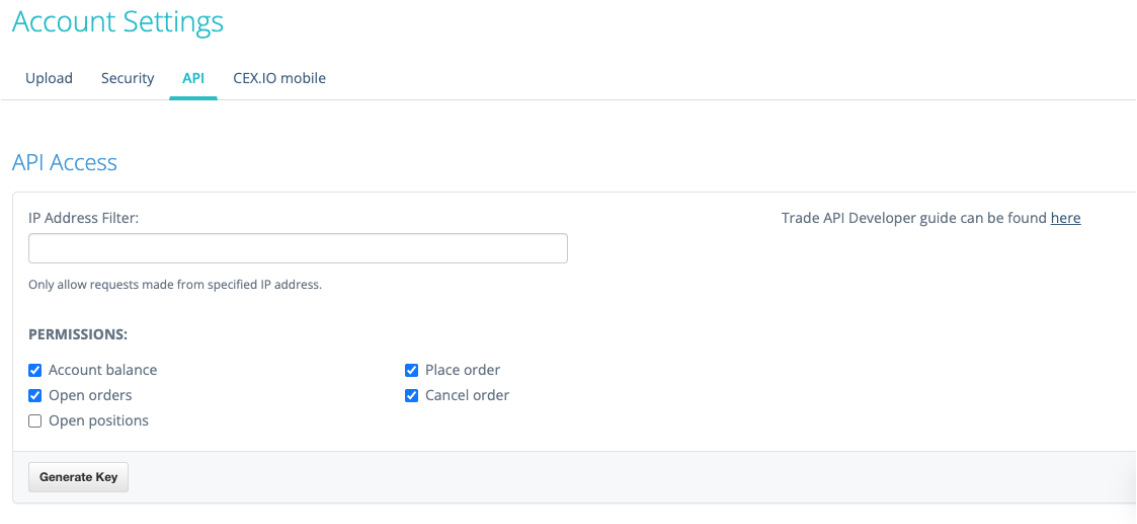
-

7. Anexos

7.1 Obtención de credenciales de la API de Cex.io

Para poder cargar las credenciales que permiten interactuar con las cuentas que los usuarios tienen en Cex.io primero el usuario debe obtener dichas credenciales en el sitio web del exchange.

Una vez que el usuario ya posee una cuenta en Cex.io y se encuentra logueado, debe ir a la sección de settings de su cuenta y de ahí a la subsección API. En el momento de escribir estas líneas la dirección url para llegar. a esta sección es https://cex.io/trade/profile#/api.



Una vez ahí debe seguir los pasos indicados con la salvedad de que para poder dar de lata ordenes en el exchange las credenciales deben tener el permiso “Open positions”, de lo contrario solo podrá consultar su información, pero no programar ordenes.