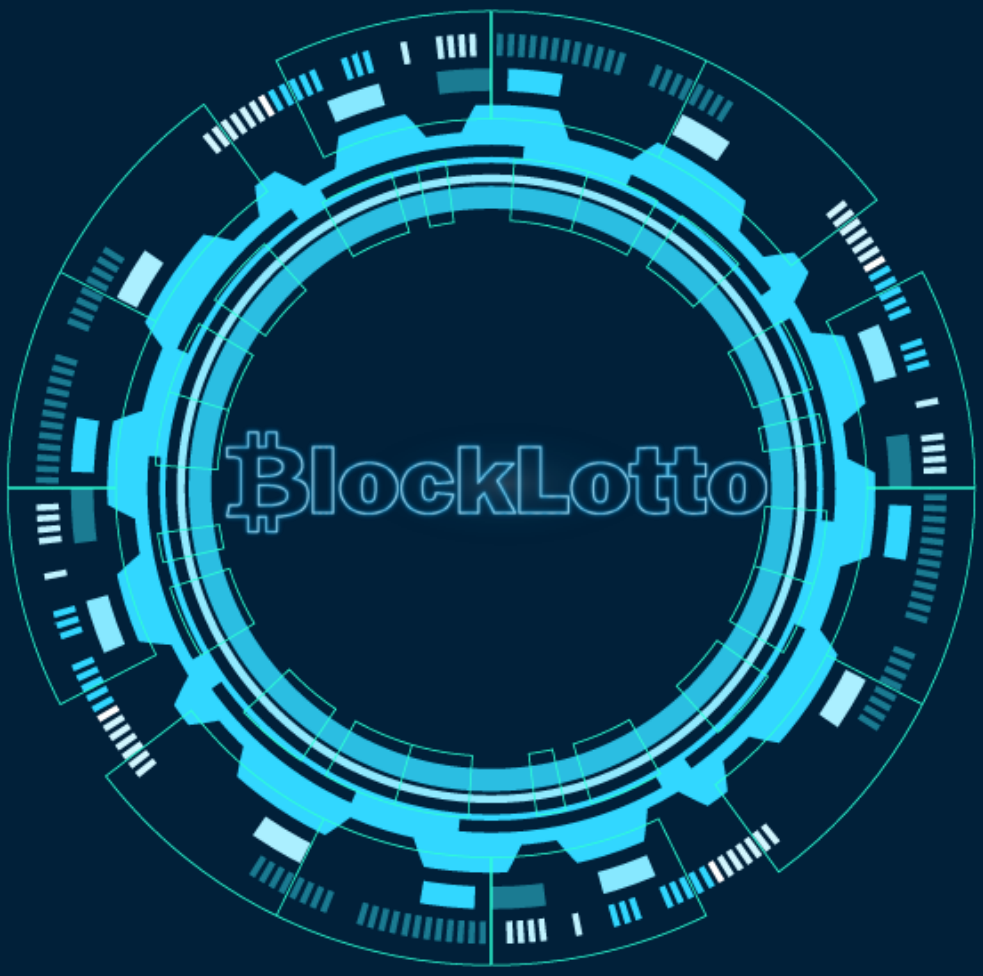
UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA

Escuela de Ingeniería Informática

Grado en Ingeniería Informática

Trabajo Fin de Grado

Curso 2018-2019



Autor: Cynthia Judith Afonso García

Tutor: José Juan Hernández Cabrera

Departamento de Informática y Sistemas

Contenido

1. Introducción

En los últimos años las criptomonedas han tenido un crecimiento exponencial, en especial Bitcoin, ya que facilitan la realización de pagos reduciendo riesgos, permiten realizar transacciones prácticamente de forma instantánea y de manera segura, facilitando así las transacciones internacionales. Haciendo todo esto, que aumenten el número de inversiones en esta criptomoneda.

Debido a este auge de Bitcoin, muchas personas se han visto interesadas en el tema, pero en un principio la manera de gestionar tus Bitcoins y transacciones ha sido mediante un wallet en el navegador del ordenador. Por este motivo, algunas empresas han visto .la oportunidad y han desarrollado aplicaciones para smartphones facilitando dichas funcionalidades a los usuarios.

Sin embargo, la mayoría de estas aplicaciones carecen de pagos instantáneos, ya que, aunque prometen realizar transacciones instantáneas siempre hay que esperar por los tiempos de confirmación (10 minutos de media). Además, la posibilidad de realizar micropagos sin costes adicionales es casi impensable.

Asimismo, nos planteamos: ¿cuántas aplicaciones móviles te permiten no solo gestionar tu wallet sino, además, invertir en sorteos para aumentar tus fondos de Bitcoins, permitiendo los micropagos sin costes adicionales y transacciones sin tiempo de confirmación? pues prácticamente ninguna.

En este TFT hemos propuesto una solución a dicho problema, desarrollando una aplicación para Android que permite gestionar el wallet (ver el saldo actual, movimientos, envío o recepción de Bitcoins, etc.), además de permitir a los usuarios participar en loterías realizando micropagos instantáneos en tiempo real, así como visualizar y llevar un control de los sorteos participados. Esto es posible, porque un servicio desarrollado exclusivamente para esta aplicación (otro TFT) lleva a cabo el proceso de realización y celebración de loterías, así como todas las funcionalidades referentes al wallet, mediante el protocolo Lightning Network (más adelante veremos más sobre este protocolo).

1.1 ¿Qué es Bitcoin?

Bitcoin es una moneda virtual. Fue concebida en 2009 por Satoshi Nakamoto. Está basada en la tecnología blockchain y, actualmente es la criptomoneda más utilizada del mundo. Bitcoin tiene diversas características:

- Descentralizada, sin una entidad intermediaria que las controle.
- Se puede cambiar bitcoins a otras monedas (euros, dólares, …) y viceversa.

- No hay intermediarios. Las transacciones se realizan directamente de persona a persona.
- Es seguro.
- Posibilidad de realizar transacciones rápidas a cualquier parte del mundo. [1]

1.2 ¿Qué es Blockchain?

Blockchain es la tecnología que permite realizar transacciones entre dos o más personas de manera segura sin la necesidad de utilizar intermediarios. Esta tecnología fue aplicada en un comienzo por Bitcoin (por lo cual es la primera plataforma blockchain).

Concretamente, blockchain es una estructura de datos en la que la información es agrupada en conjuntos (lo que se conoce como bloques) a los que se les añade meta informaciones correspondientes a un bloque de la cadena anterior de manera que, la información contenida en un bloque solo puede ser rechazada o editada modificando todos los bloques posteriores. Esto es una cualidad que facilita su utilización en entornos distribuidos de forma que la blockchain funcione como una base de datos pública no relacional que contenga un histórico inequívoco de información. [2] [3]

Por todo ello, se puede decir que blockchain es un libro de registro inmutable que contiene la historia completa de todas las transacciones que se han ejecutado en la red.

Algunas aplicaciones de la blockchain han sido:

- En el sistema financiero. Mediante carteras digitales (wallets) se permite a los usuarios realizar transacciones, agilizando los pagos y transferencias, y reduciendo notablemente su coste.
- Gestión de identidades. Permite a los usuarios crear su identidad digital y asegurarla a prueba de manipulaciones.
- Seguimiento de la cadena de suministros y prueba de procedencia. Es habitual que los productos que compramos y consumimos procedan de lugares diferentes, así se establece una cadena de suministros hasta llegar a la compañía que nos vende el producto final. Gracias a esta tecnología podemos garantizar la procedencia de los mismos.
- Como almacenamiento en la nube distribuido. En lugar de depender de servicios centralizados (Dropbox, Amazon, etc.) esta tecnología ofrece la oportunidad de almacenar datos o archivos en una red P2P, es decir, quedan guardados por muchos miembros de la red. Al no quedar la información guardada en un único espacio centralizado, es mucho más

difícil perder la información al ser atacada por hackers o al producirse un problema técnico. [4]

- Servicio de votación por internet. Esta tecnología permite que las personas no realicen más de una votación en la misma elección y favorece el anonimato del voto. [4}
- Ejecución autónoma y automática de contratos. Algunas blockchain permiten la posibilidad de crear contratos inteligentes (smart contracts). Los contratos inteligentes son programas que guardan la información de un contrato habitual, siendo capaces de ejecutarse y hacerse cumplir por sí mismos, sin necesidad de requerir de intermediarios o mediadores (como los notarios).[5]

1.3 ¿Qué es Lightning Network?

Lightning Network es un protocolo de pago que se ejecuta en la parte superior de un blockchain (más comúnmente Bitcoin) [6]. Asimismo, en la cadena de bloques de Bitcoin cada nodo almacena las transacciones que se producen en la red, pero esto tiene algunas limitaciones técnicas.

Lightning Network resuelve estas limitaciones debido a que permite realizar transacciones almacenando solamente lo necesario para realizar la misma. Por ello, destacamos tres características de Lightning:

- **Pagos instantáneos:** posibilidad de realizar pagos de manera inmediata sin preocuparse por los tiempos de confirmación (10 minutos de media en Bitcoin). Esto es posible gracias a smart contracts que no necesitan de la creación de una transacción por cada pago. Gracias a ello, podemos realizar los pagos seguros e instantáneos.
- **Escalabilidad:** capacidad de procesar millones de transacciones por segundo a través de la red a unos costes muy reducidos, debido a que se realiza fuera de la cadena de bloques reforzando todavía más la utilización de bitcoin. Esto en Bitcoin no es posible ya que su límite operativo es de siete transacciones por segundo.
- **Micropagos:** Lightning Network permite enviar hasta una cantidad tan pequeña como 0,05€ a otra persona/máquina sin coste adicional. Esto con Bitcoin fue posible en un comienzo, pero poco a poco ha ido desapareciendo esta capacidad, actualmente si envías una cantidad tan pequeña, puedes llegar a pagar más por las comisiones para garantizar que tu transacción sea confirmada que por la propia cantidad a enviar. [7] [8]

Por consiguiente, Lightning Network es un protocolo que permite escalar y acelerar las transacciones en las blockchains resolviendo diversas limitaciones técnicas que puedan presentarse en ellas, consintiendo la realización de transacciones en las que solo se almacenan los datos necesarios. Además, promueve la descentralización y soporta smart contracts, abriendo la posibilidad de realizar infinidad de aplicaciones que requieran transacciones económicas. [8]

1.3.1 ¿Cómo funciona Lightning Network?

Para comprender el funcionamiento es importante entender el siguiente término:

Canales de pago o "*payment channels*" son la base de este protocolo. En ellos se crea una transacción multifirma en la blockchain, con al menos una de ellas enviando fondos. De esta manera, cada futura transacción podrá realizarse si ambas partes han firmado con su clave privada. Los canales son lo que benefician a Lightning respecto a la red Bitcoin ya que el tiempo de apertura de estos canales es de unos diez minutos (de media) pero una vez abierto ambas partes pueden intercambiar activos de manera instantánea, utilizando los fondos almacenados en dicho canal. [7] [8]

Por lo tanto, el funcionamiento de Lightning Network consiste en abrir un canal de pago mediante la confirmación de una transacción en la blockchain correspondiente y, tras ello, ambas partes podrán realizar un número ilimitado de transacciones que actualizarán sus fondos en dicho canal sin transmitirlo a la blockchain. Sin embargo, cuando una de las partes desee cerrar el canal se transmitirá la transacción en su última versión para distribuir los fondos del canal.

Asimismo, es cierto que las transacciones de la red Bitcoin son baratas respecto a otros sistemas de pago más tradicionales, pero como ya se ha comentado anteriormente, en algunas ocasiones los usuarios necesitamos enviar pequeñas cantidades de dinero sin acogernos al coste de transmitir una transacción, pues estas deben minarse y almacenarse en los miles de nodos cada vez que se emiten. Y, este problema lo soluciona Lightning Network gracias a sus canales de pago.

2. Motivación y objetivos

Una de las motivaciones para realizar este TFT ha sido aprender a crear aplicaciones móviles desde cero. En el Grado nos enseñan a crear páginas webs o pequeñas aplicaciones de escritorio, pero desarrollar aplicaciones móviles es algo totalmente nuevo. Sin embargo, yo ya había visto (por mi cuenta) un poco sobre el desarrollo de

aplicaciones móviles en Android. Y, por ello, me llamó bastante la atención y me pareció interesante comenzar a programar apps móviles.

Además, las aplicaciones móviles cada vez tienen más importancia, los teléfonos móviles ya no los utilizamos exclusivamente para su funcionamiento original (llamar), incluso podemos llegar a pensar que es la funcionalidad menos usada hoy en día. Esto es debido, a que los usuarios nos hemos acomodado a tener algunas utilidades a mano que nos facilitan el día a día, aunque esto hace unos años era impensable, actualmente deseamos más y más comodidad. Por ello mismo, cuando mi tutor me comentó la propuesta de TFT me pareció una idea brillante y con mucha visión de futuro.

Otra de las motivaciones para llevar a cabo este TFT ha sido aprender más y de forma más cercana sobre Bitcoin, ya que es un tema bastante actual y, que cada año consigue más importancia en la economía mundial.

En resumen, la motivación para elegir y desarrollar este TFT ha sido aprender a desarrollar de forma completa en una tecnología prácticamente desconocida para mí y, de forma autodidacta, así como, con un tema muy actual en la sociedad, como es Bitcoin.

Los objetivos que se desean conseguir con este TFT son:

- Desarrollar la habilidad para poder crear un convenio entre ambas partes del proyecto (BlockLotto se divide en dos TFT, posteriormente se explicará con más detalle).
- Diseñar e implementar una aplicación Android que cumpla con los requisitos previos establecidos y que sea fácil de usar.
- Integrar las funcionalidades de la API (el otro TFT que conforma BlockLotto) en la aplicación utilizando el protocolo JSON.

3. Aportación Social

Actualmente, términos como: blockchain, smart contracts, criptomonedas, minado o bitcoin son muy utilizados diariamente. Esto comenzó como una novedad, pero poco a poco se ha convertido en un tema importante en nuestras vidas, ya no sólo como una tecnología nueva de la que podamos hablar, sino algo tan grandioso como para usarlo diariamente. Tal y como expone Alex Rayón Jerez en su artículo *Blockchain, Bitcoin y su (supuesta) aportación social* [9] muchas grandes empresas ya han comenzado a explotar estas tecnologías. Asimismo, explica algunos ejemplos:

- *Carrefour* en Francia ya está utilizando estas cadenas de bloques para trazar la procedencia de productos como la miel, queso, leche, salmón o hamburguesas. Y, en España ha sido pionera introduciendo este sistema de trazabilidad alimentaria blockchain en uno de sus productos: el pollo campero. [10]

- *El gigante del sector de los diamantes De Beer*, ha implantado una solución basada en blockchain para controlar cómo van cambiando de manos sus diamantes.

- *Telefónica, Seat* y el fabricante de componentes *Ficosa,* están trazando la cadena de suministro a través de una solución de contratos inteligentes.

- *Banco Santander* ha puesto en marcha One Pay FX, un nuevo servicio basado en blockchain para realizar transferencias de manera rápida, sin esperar a autorizaciones posteriores, para verificar las mismas de manera instantánea.

- *AXA* ha lanzado un seguro contra retrasos de salida de vuelos: compramos un billete de avión por 150€ y un seguro de 25€ por si el avión se retrasa 2 horas. Si pasadas esas horas de la hora de salida no ha despegado aún el avión, automáticamente (sin intervención de terceros), recibes 175 € en tu cuenta bancaria.

Por todo ello, vemos cómo esta tecnología está influyendo y cada vez con más importancia en diferentes sectores sociales.

De manera análoga, los dispositivos móviles son primordiales en la vida cotidiana de las personas. En la Ilustración 1 se puede observar los datos obtenidos por Consumer Barometer with Google en España [11]. En la misma, se observa como aumenta el uso de los dispositivos móviles con el paso de los años. Este uso en crecimiento se debe al desarrollo continuo de novedosas aplicaciones móviles que nos aportan funcionalidades complejas mediante un manejo mucho más sencillo, facilitando la vida cotidiana y creando un uso frecuente de los smartphones.

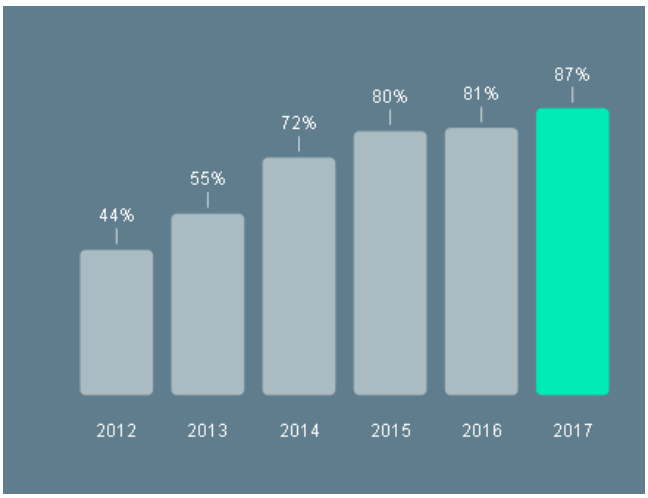


Ilustración 1 – Uso de los smartphones en los últimos 6 años

Además, en los últimos años se ha fomentado el uso de aplicaciones relacionadas con las criptomonedas, ofreciendo a los usuarios capacidades que varían desde conocer la cotización actual de las criptomonedas hasta almacenarlas (wallets) o enviarlas a otras personas.

Todo ello, ha hecho que la relación aplicación móvil- criptomonedas se haya convertido en algo tan popular. Viéndose beneficiados sectores como:

- Sector alimentario. Como ya se ha explicado anteriormente Carrefour ha sido pionero en España, pero probablemente muchas más empresas de este sector se sumen.
- Sector comercial. Anteriormente se han explicado algunas empresas que ya han empezado a incorporar blockchain y contratos inteligentes en la cadena de suministro.
- Sector económico. La banca se está viendo y se verá más afectada. Esto no es una sorpresa, ya que numerosos bancos están invirtiendo en criptomonedas. Un claro ejemplo es el banco BBVA que ya ha invertido en Coinbase (un wallet que permite comprar, vender y almacenar bitcoins y otras criptomonedas como Litecoin o Ethereum.
- Además, el ámbito de la ciberseguridad se está viendo beneficiado por la blockchain, ya que al eliminar intermediarios en transacciones financieras se reducen las posibilidades de verse afectado por hackers.
- Sector educativo. Algunas universidades ya han comenzado a utilizar estas tecnologías para verificar las credenciales de sus estudiantes.

De la misma forma, se verá afectado el sector social, creando un fuerte movimiento social en la utilización de estas tecnologías.

Además, como los seres humanos somos ambiciosos por naturaleza no solo nos basta con tener y usar criptomonedas. Ya se ha comentado como algunos bancos ya han

comenzado a invertir en este territorio, y poco a poco las personas tendrán el deseo de poder invertir también en él. Y, ¿qué hay más atractivo para agradar tus beneficios en criptomonedas que un sorteo? Actualmente, son muchas las personas que invierten su dinero físico en este tipo de juegos de azar, así que, dar la posibilidad a los usuarios de poder invertir con su dinero digital fomentaría la utilización de estas tecnologías, ofreciendo la posibilidad de tener una recompensa satisfactoria.

Con este proyecto se pretende fomentar el uso de estas tecnologías, así como contribuir al desarrollo de las mismas. Asimismo, se ha pensado en lo esencial que están siendo las aplicaciones móviles para nuestra sociedad en los últimos años, así como el auge de las criptomonedas y el deseo cada vez más común de tener más. Por tanto, en este proyecto se pretende aportar a los usuarios:

La posibilidad de tener a mano, de manera sencilla y en todo momento:

- Sus bitcoins, ver la cantidad de su saldo y llevar el control de los mismos.
- La gestión de su dinero, enviando o recibiendo bitcoins desde otro wallet, o simplemente llevando un control de los movimientos de su cartera.
- La oportunidad de aumentar sus fondos participando en sorteos, de forma bastante rápida y sin costes adicionales.
- Visión sencilla de los sorteos en los que está participando, así como en los que ya ha participado.

Con todo esto, la aplicación no sólo proporciona a los usuarios la posibilidad de gestionar su economía digital, sino que, además, da la oportunidad de obtener una recompensa, favoreciendo la economía de los mismos o simplemente aportando un entretenimiento. Así que, todo esto trae beneficios tanto al sector económico como al sector social.

Por otra parte, desarrollar BlockLotto ha aportado nuevos conocimientos sobre el desarrollo de software (orientado a aplicaciones móviles Android) desde el análisis del producto hasta la implementación del mismo. Además, ha aportado conocimientos (teóricos) sobre blockchain y criptomonedas (en concreto Bitcoin, usando el protocolo Lightning Network). Del mismo modo, ha permitido afianzar conocimientos obtenidos durante la carrera, desde los más trabajados durante la misma (como establecer los requisitos necesarios de nuestro software, desarrollar código Java o utilizar código limpio) hasta los que no se pudieron poner tanto en práctica, como el uso avanzado de algunas tecnologías (conexiones a servicios webs, herramientas para desarrollar software, herramientas para organizar un equipo de desarrolladores, etc.)

4. Justificación de las Competencias

La realización de este TFT ha permitido cubrir las siguientes competencias:

CII01. Capacidad para diseñar, desarrollar, seleccionar y evaluar aplicaciones y sistemas informáticos, asegurando su fiabilidad, seguridad y calidad, conforme a principios éticos y a la legislación y normativa vigente.

Como se puede observar en este documento y en el proyecto en sí, este TFT ha abarcado todas las partes del desarrollo de un proyecto software, desde el análisis hasta su implementación y la correspondiente comprobación de su correcto funcionamiento. La creación de esta aplicación ha necesitado un análisis previo (explicado con más detenimiento y exactitud en el apartado *Análisis*) en el que se ha establecido las características, funcionalidades y/o requisitos necesarios para la aplicación. Además, se ha pensado en el posible futuro que pueda tener la misma, incluyendo al público al que va dirigida, así como, las capacidades y aportaciones que la app pueda dar a los usuarios.

Por otro lado, para comenzar con la implementación de la aplicación ha sido necesario crear un convenio entre ambas partes de BlockLotto (ya que es un proyecto conformado por dos TFT). Esto ha sido primordial, ya que en este proyecto se necesita gestionar la interacción usuario-aplicación y, ha sido imprescindible comunicar al desarrollo del servicio LottoService (el otro TFT), qué necesitamos mostrar al usuario, así como, las funcionalidades que necesitaría aportar a la app.

Todo esto, con su posterior implementación y pruebas correspondientes (explicadas en el apartado *Desarrollo*), han conformado las capacidades que abarca esta competencia.

CII02. Capacidad para planificar, concebir, desplegar y dirigir proyectos, servicios y sistemas informáticos en todos los ámbitos, liderando su puesta en marcha y su mejora continua y valorando su impacto económico y social.

Como anteriormente se ha comentado, al ser un proyecto formado por dos TFT ha sido necesaria la planificación y el acuerdo entre ambas partes para la correcta realización de los mismos. Asimismo, en este documento se puede observar en el apartado anterior *Aportación Social* el impacto y/o la contribución de este proyecto en la sociedad.

De igual modo, para el desarrollo adecuado de este proyecto ha sido esencial captar la forma apropiada para mostrar a los usuarios las funcionalidades establecidas. Esto desde un principio no ha sido sencillo, se ha necesitado ver diferentes maneras de reproducir la misma funcionalidad, pero mostrándolo de una forma diferente al usuario, hasta llegar a la interfaz más simple y fácil de usar de manera que no produzca incertidumbre y/o equivocación a la hora de usar la aplicación. Además, la implementación paralela de los dos TFT que conforman este proyecto ha posibilitado que cada parte comunique mejoras a añadir y/o problemas a solucionar tanto en el

propio TFT, como en el otro (información adicional, funcionalidades planteadas de una forma más sencillas, etc.). Asimismo, se puede concluir que el desarrollo de esta aplicación ha incluido una mejora continua, así como, se ha pensado en posibles extensiones para la misma (ver apartado *Posibles Extensiones*).

CII03. Capacidad para comprender la importancia de la negociación, los hábitos de trabajo efectivos, el liderazgo y las habilidades de comunicación en todos los entornos de desarrollo de software.

En la justificación de las dos competencias anteriores se ha comentado, que un paso indispensable en este proyecto ha sido la coordinación de ambos TFT. Estableciendo un pacto entre las mismas, y posteriormente trabajando ambas en paralelo, comunicando cualquier error, fallo o posible mejora de algún componente de una de las dos partes (o de ambas). Esto abarca desde el proceso de análisis hasta el proceso de implementación y pruebas. De modo que, se han cumplido y aplicado todas las capacidades comentadas en esta competencia.

CII08. Capacidad para analizar, diseñar, construir y mantener aplicaciones de forma robusta, segura y eficiente, eligiendo el paradigma y los lenguajes de programación más adecuados.

En la justificación de la competencia CII01 se explica, que este TFT abarca todas las partes de desarrollo de un proyecto (desde el análisis hasta su correspondiente implementación, pruebas y corrección de errores). Además, se ha elegido el lenguaje de programación pertinente dado que es una aplicación dirigida exclusivamente a dispositivos Android y se ha desarrollado en Android Studio.

CII011. Conocimiento y aplicación de las características, funcionalidades y estructura de los Sistemas Distribuidos, las Redes de Computadores e Internet y diseñar e implementar aplicaciones basadas en ellas.

La aplicación desarrollada necesita de funcionalidades implementadas en el LottoService para poder cumplir con los requisitos y características estipulados. De esta manera, se hace uso de Internet que, mediante una clase desarrollada con los métodos propiamente necesarios, la aplicación hace peticiones en JSON a un servicio web REST.

CII013. Conocimiento y aplicación de las herramientas necesarias para el almacenamiento, procesamiento y acceso a los Sistemas de información, incluidos los basados en web.

Como se ha explicado en el apartado anterior, el desarrollo de este TFT ha necesitado crear los utensilios necesarios para conectar la aplicación con el otro TFT que conforma este proyecto, y poder hacer uso de las funcionalidades que ofrece a la app.

Asimismo, para poder elaborar estos utensilios de manera correcta se ha empleado software profesional con el que se ha comprobado el funcionamiento adecuado y deseado de los mismos. Además, no solo ha sido consistido en la conexión y la utilización de las funcionalidades ofrecidas por el servicio web, sino que, además, como cada una de ellas necesita una información diferente, tienen por consiguiente una estructura de petición diferente. Por tal motivo, ha sido necesaria la transformación de las peticiones del usuario con la información captada desde la aplicación en la solicitud correspondiente al servicio web. De la misma manera, ha sido necesaria la captación de los resultados devueltos, y su posterior formateo en estructuras que fuesen útiles y visibles para los usuarios.

CII016. Conocimiento y aplicación de los principios, metodologías y ciclos de vida de la ingeniería de software.

Esta competencia está abarcada con el desarrollo en sí de este proyecto, ya que se usan metodologías iterativas, se hace uso de diagramas que facilitan el planteamiento del producto y sus características, del mismo modo que se utilizan conocimientos y herramientas propios de la ingeniería de software.

CII017. Capacidad para diseñar y evaluar interfaces persona computador que garanticen la accesibilidad y usabilidad a los sistemas, servicios y aplicaciones informáticas.

Las capacidades y habilidades descritas en esta competencia se han logrado, ya que, como se ha comentado en apartados anteriores, este TFT abarca todas las partes del desarrollo de un proyecto software (desde el análisis hasta las pruebas). De esta manera, se ha diseñado la interfaz de la aplicación ajustando las funcionalidades que esta ofrece a la pantalla de un smartphone. Esto es bastante complejo, ya que no se dispone de una pantalla excesivamente grande donde mostrar toda la información, y hace necesario la priorización de la información a mostrar a los usuarios (como nombre, fecha o precio de los sorteos), así como, la creación de funcionalidades que extiendan dicha información.

Asimismo, en el proceso de implementación de la aplicación se han hecho cambios en la interfaz, intentado buscar la interfaz más sencilla e intuitiva que permita a los usuarios utilizarla sin llevarlos a errores, evitando la frustración de los mismos y garantizando una aplicación manejable que aporte todas las funcionalidades que promete ofrecer de

manera bastante simple. Además, la gama cromática elegida para esta aplicación permite a todos los usuarios distinguir con claridad cada uno de los componentes que conforman la app.

Todo ello, permite a los usuarios utilizar la aplicación sin ningún tipo de problemas y garantizar que la interacción usuario-aplicación se lleve a cabo sin ningún inconveniente.

IS01. Capacidad para desarrollar, mantener y evaluar servicios y sistemas software que satisfagan todos los requisitos del usuario y se comporten de forma fiable y eficiente, sean asequibles de desarrollar y mantener y cumplan normas de calidad, aplicando las teorías, principios, métodos y prácticas de la ingeniería del software.

En la justificación de competencias anteriores ya se ha comentado que en este proyecto se han implementado las funcionalidades establecidas previamente en el proceso de análisis (en el establecimiento de características y requisitos). Además, se han evaluado posibles interfaces y se ha seleccionada la más adecuada (por claridad y facilidad en su correcta utilización). Asimismo, en el proceso de implementación se han utilizado técnicas como código limpio o refactoring que facilitan la realización de mejoras, eliminación de errores, así como el mantenimiento en sí de la aplicación. De esta manera, este proyecto se ha desarrollado cumpliendo las normas de calidad pertinentes, asegurando las utilidades ofrecidas a los usuarios y haciendo uso de teorías y prácticas propias de ingeniería del software.

IS02. Capacidad para valorar las necesidades del cliente y especificar los requisitos software para satisfacer estas necesidades, reconciliando objetivos en conflicto mediante la búsqueda de compromisos aceptables dentro de las limitaciones derivadas del coste, del tiempo, de la existencia de sistemas ya desarrollados y de las propias organizaciones.

Este proyecto ha permitido cubrir esta competencia, ya que como se ha nombrado en competencias anteriores, previo al proceso de implementación se ha hecho un procedimiento de análisis (ver *Análisis*) en el cual se ha planteado las características de la app, modelo de negocio, aplicaciones similares que puedan haber en el mercado, así como también los requisitos que esta debe poseer para satisfacer las necesidades de los clientes (ver *Requisitos*).

IS03. Capacidad de dar solución a problemas de integración en función de las estrategias, estándares y tecnologías disponibles.

Esta competencia ha sido abarcada ya que para poder integrar la otra parte de BlockLotto en la aplicación, ha sido necesario buscar estrategias que no bloqueen la aplicación mientras se realiza una petición a la API (puesto que se comunican mediante internet, y el LottoService debe procesar la petición y devolver un resultado). Sin el uso de las tácticas oportunas, al realizar el usuario una petición (como podría ser la compra de boletos de un sorteo) podría verse bloqueada la interfaz hasta recibir una respuesta por parte de la API (pudiéndose ralentizar internet y alargar todavía más la espera del usuario). Esto sería bastante ineficaz e incómodo de usar para los usuarios produciendo, además, una impresión bastante mala y desastrosa de la aplicación.

IS04. Capacidad de identificar y analizar problemas y diseñar, desarrollar, implementar, verificar y documentar soluciones software sobre la base de un conocimiento adecuado de las teorías, modelos y técnicas actuales.

El desarrollo de este proyecto no ha sido sencillo. Desde la búsqueda de una interfaz adecuada hasta la integración de la API se ha comprobado que todo funciona como se deseaba, y en caso de no ser así, se han buscado soluciones y alternativas en la implementación software para que todo funcione cubriendo las necesidades de los usuarios. Asimismo, el proyecto en sí agrupa temas actuales (aplicaciones móviles y gestión de criptomonedas). Y, se ha desarrollado con técnicas y herramientas actuales, teniendo estas últimas continuas actualizaciones que mejoran su utilización.

IS05. Capacidad de identificar, evaluar y gestionar los riesgos potenciales asociados que pudieran presentarse.

En el proceso de análisis de este TFT se ha identificado los posibles riesgos que podrían suceder, así como, el plan de acción pertinente. Por ello, se ha alcanzado las capacidades que abarcan esta competencia. (Ver *Riesgos*)

IS06. Capacidad para diseñar soluciones apropiadas en uno o más dominios de aplicación utilizando métodos de la ingeniería del software que integren aspectos éticos, sociales, legales y económicos.

Las habilidades que contempla esta competencia se han alcanzado por todo lo comentado en las competencias anteriores.

5. Análisis

5.1 Descripción y Características

BlockLotto es un proyecto conformado por dos trabajos de fin de título, los cuales son: el documentado en esta memoria (LottoWallet) y el TFT de Jorge Fernández Molines (LottoService), ambos tutorizados por José Juan Hernández Cabrera.

En concreto, este TFT ha consistido en el desarrollo de una aplicación móvil para dispositivos Android. Asimismo, la aplicación cuenta con características propias de una app wallet, es decir, la aplicación permite a los usuarios tener una cartera digital en la que pueden almacenar bitcoins, recibirlos o enviarlos a otro wallet, así como llevar un control de los movimientos del wallet. Además, permite a los usuarios invertir sus bitcoins en sorteos, y ver en cuales ha participado (tanto los finalizados como los sorteos pendientes de celebración) para llevar a cabo un seguimiento de los mismos.

Del mismo modo, esta aplicación tiene la peculiaridad que todas las transacciones que el usuario realice (tanto envío o recepción de bitcoins como compra de boletos) se efectuarán de manera instantánea y sin costes adicionales. Esto se garantiza gracias al protocolo Lightning Network del que hemos hablado anteriormente (ver *¿Qué es Lightning Network?*).

De esta manera, gran parte del desarrollo de este proyecto ha consistido en la elaboración de una aplicación Android que proporcione una interfaz adecuada para el usuario con todas las cualidades anteriormente nombradas. Hay que añadir que, dicho desarrollo no sólo ha consistido en implementar la interfaz apropiada que aporte las características comentadas, sino que además se ha hecho una integración del otro TFT que conforma BlockLotto. LottoService es un servicio web REST que gestiona la creación y celebración de sorteos, al igual que se encarga de realizar los pagos entre wallets y el ingreso de premios utilizando el protocolo Lightning Network.

A continuación, en la Ilustración 2 se expone un esquema de la estructura de BlockLotto:

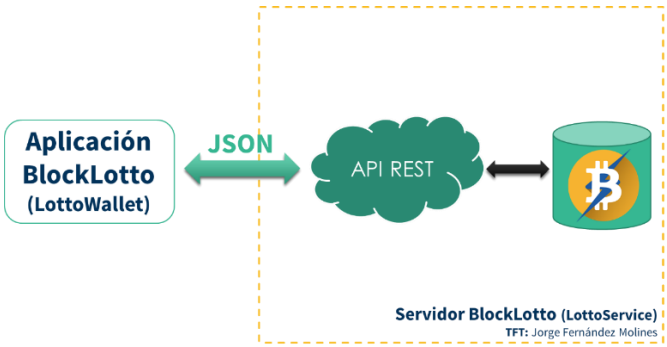


Ilustración 2 - Constitución de BlockLotto

Posteriormente, en el apartado *Desarrollo* se explicará más detalladamente las fases de implementación del proyecto.

En pocas palabras, este proyecto ha consistido en la implementación de una aplicación orientada a dispositivos Android, que comunicándose con un servicio web REST proporciona a los usuarios las siguientes funcionalidades:

- **Crear un wallet**. Permite a los usuarios crear una cartera digital en la que podrán almacenar sus bitcoins y con la que podrán realizar transacciones.

- **Restaurar su wallet**. Si un usuario ya ha usado anteriormente la aplicación y por algún motivo (pérdida del smartphone, formateo, etc.) no tiene acceso al wallet que se creó previamente (el cual posee sus bitcoins, compras, movimientos, premios, etc.), gracias al sistema de seguridad de veinticuatro palabras podrá recuperar su cartera con todos sus datos, dinero, etc.

- **Mostrar loterías.** Los usuarios verán tres listas de sorteos, en las que aparecerán los sorteos y su correspondiente información. Las tres categorías de sorteos son:
  - **Próximos sorteos.** El usuario podrá ver los sorteos que próximamente se celebrarán en BlockLotto y podrá comprar boletos para participar en el mismo.
  - **Sorteos pendientes.** El usuario ver todos los sorteos en los que ha participado y que aún no se han celebrado. En otras palabras, podrá llevar a cabo el control de sus compras.
  - **Sorteos celebrados.** El usuario verá todos los sorteos en los que ha participado y que ya han sido celebrados. Es decir, el usuario observará un historial de todas las compras que ha efectuado en la aplicación.

- **Comprar lotería.** Como se ha comentado en el punto anterior, cuando se inicia la aplicación el usuario ve los próximos sorteos de BlockLotto, y tiene la posibilidad de comprar boletos correspondientes a un sorteo para participar en el mismo.

- **Mostrar saldo del wallet.** El usuario podrá ver el saldo de su cartera mientras hace uso de la aplicación, de esta manera, podrá llevar el control de su dinero.

- **Mostrar el saldo de los canales**. Con esta funcionalidad el usuario podrá ver el saldo que posee en los canales que ha abierto previamente. Los canales se abren comprando boletos, enviando bitcoins, etc. (para entender este concepto ver el apartado *¿Cómo funciona Lightning Network?* explicado anteriormente).

- **Recuperar fondos de los canales.** El usuario al ver el saldo total en los canales abiertos puede recuperarlos para así almacenar esa cantidad de bitcoins en su wallet. Además, con esta funcionalidad el usuario cierra todos los canales de forma segura.

- **Enviar bitcoins.** Los usuarios pueden enviar sus bitcoins a otros wallets, esto lo podrán realizar de tres formas:
  - Introduciendo el importe y la clave pública del wallet al que se desea enviar los bitcoins.
  - Escaneando la clave pública del wallet al que desea enviar el dinero e introduciendo manualmente el importe. La aplicación muestra la clave pública de cada wallet, también, en código QR.
  - Escaneando la petición creada por el wallet receptor del dinero. Esta petición contará con la cantidad de bitcoins que usuario desea recibir y su clave pública, todo ello creando un código QR.

- **Recibir bitcoins.** Los usuarios pueden recibir dinero de otros wallets, como se ha comentado en el punto anterior. Además, si el wallet resulta ser ganador de un sorteo el sistema ingresará el premio en el mismo.

- **Mostrar movimientos.** Cada wallet realizará unas transacciones (ya sea compra de boletos, recepción de premios, envío de bitcoins, etc.), todas ellas serán visibles por el usuario dueño del wallet en cuestión.

- **Mostrar notificaciones.** El usuario al resultar ganador de un premio, no solo recibirá el ingreso del premio, sino que, además recibirá una notificación indicando que cantidad de dinero ha ganado y en qué sorteo ha sido el ganador. De esta manera, el usuario podrá llevar un control de los sorteos en los que ha sido premiado y los premios que ha recibido.

5.2 Estudio de Aplicaciones Similares

Como ya se ha comentado, actualmente hay infinidad de aplicaciones móviles y, por tal motivo hay muchas que ofrecen funcionalidades semejantes. Sin embargo, las empresas y desarrolladores se esfuerzan en ofrecer a los usuarios características propias que las diferencian unas de otras. Durante el proceso de análisis no se encontraron aplicaciones que ofrezcan todas las funcionalidades que proporciona nuestra aplicación, pero sí que es cierto que existen apps que proponen algunas de ellas.

A continuación, en la Tabla 1 se muestra una comparativa de varias aplicaciones que ofrecen alguna de las cualidades de nuestra app, así como las diferencias que existen entre ellas y BlockLotto.

Aplicación	Descripción	Similitudes con BlockLotto	Diferencias con BlockLotto
Pryze	Es una aplicación que permite a los usuarios participar en sorteos, así como crearlos. Pryze es el primer protocolo de sorteos automatizado basado en Ethereum, para resolver los problemas de confianza, custodia, administración y cumplimiento legal mediante la participación de sorteos y competiciones en el blockchain. A través de esta aplicación los usuarios pueden beneficiarse de la tecnología blockchain para transformar todos los procesos administrativos que precisan los sorteos en procedimientos fáciles y automatizados, creados y operados a través de un sistema descentralizado, de código abierto y transparente. [12] Como se ha comentado está basado en Ethereum y para utilizar dicha aplicación es necesario poseer un wallet de este entorno.	Ofrece a los usuarios la posibilidad de participar en sorteos cuyos procesos administrativos están automatizados. Notifica al usuario cuando ha sido ganador de un sorteo.	Está basado en Ethereum no en Lightning Network. Los premios no siempre son criptomonedas. No permite gestionar el wallet.
FireLotto	Es una plataforma que permite a los usuarios participar en sorteos descentralizados basados en la cadena de bloques de Ethereum, gestionándolos automáticamente mediante contratos inteligentes sin ninguna interacción humana. Ofrece dos tipos de sorteos: un sorteo de la lotería y un juego instantáneo. [13]	Ofrece a los usuarios la posibilidad de participar en sorteos cuyos procesos administrativos están automatizados. Los premios de los sorteos son criptomonedas. Permite gestionar el wallet.	No es una aplicación móvil es una web. Está basado en Ethereum no en Lightning Network.
Jaxx Classic	Aplicación móvil que permite a los usuarios tener una cartera digital con diferentes criptomonedas. De esta manera, la app ofrece la posibilidad de almacenar dinero, así como de recibirlo o enviarlo a otros wallets.	Permite administrar el wallet (desde observar los movimientos del mismo para llevar a cabo un control de	No ofrece a los usuarios la posibilidad de participar en sorteos para aumentar la

		las transacciones realizadas, hasta enviar/recibir criptomonedas a/desde otro wallet).	cantidad de dinero que poseen. No hace uso del protocolo Lightning Network, por tanto, las transacciones no se harán de manera instantánea y, sin costes adicionales.
Bitcoin Wallet	Aplicación móvil que permite a los usuarios tener un wallet de Bitcoins. Asimismo, facilita a los usuarios la gestión de una cartera digital, permitiendo ver las transacciones realizadas, enviar/ recibir bitcoins, realizar peticiones de dinero, etc.	Permite gestionar la cartera digital (incluyendo características propias de este tipo de apps).	No ofrece a los usuarios la posibilidad de participar en sorteos para aumentar la cantidad de dinero que poseen. No hace uso del protocolo Lightning Network, por tanto, las transacciones no se harán de manera instantánea y, sin costes adicionales.

Tabla 1 - Comparativa de otras aplicaciones con BlockLotto

Por esta comparativa, se puede concluir que, aunque hay infinidad de aplicaciones que ofrecen funcionalidades similares, no existe ninguna app con los objetivos, características y funcionalidades de BlockLotto.

5.3 Requisitos

En el proceso de análisis del proyecto se incluye, también, el establecimiento de los diferentes requisitos de la aplicación. De esta manera, se ha descrito previamente el comportamiento que se deseaba que tuviera la aplicación para su posterior implementación. A continuación, se explican los requisitos establecidos para este proyecto:

Requisitos funcionales:

- El sistema debe permitir crear wallets.
- El sistema debe devolver las veinticuatro palabras de seguridad cuando el wallet sea creado correctamente.
- El sistema debe permitir restaurar wallets creados previamente, utilizando las veinticuatro palabras de seguridad.
- El sistema debe guardar en el dispositivo la clave pública, la dirección y la id del wallet.
- El sistema debe mostrar al usuario todos los sorteos disponibles por el sistema.
- El sistema debe permitir al usuario comprar boletos de un sorteo.
- El sistema debe permitir cancelar la compra de boletos de un sorteo.
- El sistema debe permitir seleccionar el número de boletos a comprar por el usuario.
- El sistema debe mostrar los sorteos en los que participa el usuario.
- El sistema debe mostrar el historial de sorteos en los que el usuario a participado y ya han sido celebrados.
- El sistema debe mostrar toda la información de los diferentes tipos de sorteos.
- El sistema debe permitir ordenar los sorteos por: precio ascendente, precio descendente o por fecha (más recientes primero).
- El sistema debe mostrar el saldo disponible en el wallet.
- El sistema debe mostrar el saldo acumulado en todos los canales abiertos.
- El sistema debe permitir recuperar el saldo de los canales acumulados.
- El sistema debe permitir cerrar los canales abiertos.
- El sistema debe mostrar los movimientos del wallet.
- El sistema debe permitir mostrar los movimientos según sean: pagos, ingresos o ambos.
- El sistema debe mostrar la clave pública de la cartera digital del usuario.
- El sistema debe permitir copiar la clave pública del wallet del usuario.
- El sistema debe mostrar un código QR con la clave pública del wallet del usuario.
- El sistema debe generar y mostrar un código QR de petición de bitcoins.
- El sistema debe poder escanear un código QR para conocer la clave pública del wallet al que se desea enviar bitcoins.
- El sistema debe poder escanear un código QR de petición de bitcoins.
- El sistema debe permitir el envío de bitcoins a otro wallet, utilizando la clave pública del wallet del usuario, la clave pública del wallet de recepción y la cantidad a enviar.
- El sistema debe mostrar todas las notificaciones enviadas por el sistema de loterías.

- El sistema debe añadir una notificación a la lista de notificaciones cuando el usuario es ganador de un premio.
- El sistema debe mostrar toda la información relevante de la notificación enviada por el sistema.
- El sistema debe permitir marcar las notificaciones como leídas.
- El sistema debe mostrar información acerca de la aplicación.

Requisitos no funcionales:

- El sistema debe ser desarrollado para dispositivos Android.
- La aplicación debe ser compatible con todas las versiones de Android, desde la versión 6.0 (API 23, Marshmallow).
- La aplicación debe ser instalada en el dispositivo antes de usarla.
- El sistema necesita acceso a Internet para mostrar la información del usuario y realizar transacciones.
- El sistema necesita acceso a la cámara del dispositivo para escanear los códigos QR generados por la aplicación.
- El sistema no debe bloquearse mientras se está procesando la transacción.
- El sistema debe permitir una navegación fluida por todos los componentes de la aplicación.
- El tiempo de aprendizaje del sistema por un usuario debe ser menor de 30 minutos.
- El sistema debe poseer interfaces gráficas bien formadas e intuitivas para el usuario.
- El sistema debe proporcionar mensajes de error que sean informativos y orientados al usuario final.
- El promedio de duración de fallos no podrá ser mayor a 15 minutos.
- El sistema utilizará el protocolo JSON para comunicarse con el LottoService.

5.4 Modelo de Negocio

El modelo de negocio que seguirá la aplicación será similar al que siguen muchas apps existentes actualmente. De esta manera, la aplicación se publicaría cuando estén terminadas todas las funcionalidades principales (como pueden ser crear el wallet o mostrar los sorteos disponibles). Tras ser publicada, se realizarán mejoras y/o extensiones cada cierto tiempo (mejoras de rendimiento, corrección de errores si los hubiera o incorporación de nuevas funcionalidades, aumentando las capacidades y utilidades ofrecidas a nuestros clientes).

Por otro lado, la aplicación está dirigida a todos los usuarios interesados en criptomonedas, especialmente, aquellos que desean ganar y gestionar bitcoins. La app será gratuita, siendo esto un factor bastante atractivo para muchas personas.

Además, se hará uso tanto de canales inbound como de canales outbound, recogiendo de cada uno de ellos sus mejores ventajas:

- *En canales inbound:*
  - Conocer mejor al consumidor, así como su proceso de adquisición y compra.
  - Aumentar la confianza de los clientes hacia la aplicación.
  - Mejorar la reputación online del producto.
  - Aumentar los seguidores en redes sociales que hacen publicidad pasiva del producto.
  - No es intrusivo (no agobia al consumidor) y genera un mayor compromiso con el producto.
  - Además, los resultados se pueden medir a tiempo real.

- *En canales outbound:*
  - Es visto por un mayor tipo de público.
  - Es perfecto para quienes buscan que su producto sea conocido.
  - Se enfoca en el producto y los servicios que se ofrecen.
  - Directo: este tipo de canales va al grano, no se anda por las ramas, deja claro el mensaje que quiere exponer de manera inmediata.
  - Mensaje recibido: es cierto que es necesario mostrárselo a mucha más gente, pero, dado que el mensaje expuesto es claro y conciso, llega a un mayor número de personas.

Asimismo, la combinación de estos dos tipos de canales permite asegurar que el producto llegue a un mayor número de personas dándolo a conocer tanto online como offline, produciendo un incremento en el número de personas interesadas en esta aplicación, garantizando la confianza de los mismos hacia el producto, así como incrementando la utilización del mismo.

A continuación, en la Tabla 2 se muestra el Lean Canvas de BlockLotto. En él se puede ver de forma bastante concreta y explícita, la idea y concepto de nuestro modelo de negocio:

<table>
<tr>
<td rowspan="2">PROBLEMA
Para almacenar, gestionar y aumentar la cantidad de bitcoins que poseemos hacemos uso de diferentes plataformas o aplicaciones.

Actualmente, las criptomonedas están en auge y existe una gran demanda de aplicaciones que ofrecen estas funcionalidades. Sin embargo, la mayoría de aplicaciones no integran todos estos servicios en su conjunto.

Las aplicaciones que permiten realizar transacciones con criptomonedas requieren un tiempo de espera por parte del usuario, así como costes adicionales en la realización de micropagos.

Alternativas existentes:

Jaxx, Pryzes, etc. (ver Estudio de Aplicaciones Similares).</td>
<td>SOLUCIÓN
Concentrar en una única aplicación la posibilidad de almacenar y multiplicar bitcoins, así como gestionar un wallet. Pudiendo, además, realizar transacciones de manera instantánea sin tener ningún tipo de costes adicionales (ni monetarios, ni de tiempo).</td>
<td rowspan="2">PROPUESTA DE VALOR ÚNICA
La manera más fácil de ganar y gestionar bitcoins de forma rápida y segura.

High-level concept:

Bitcoins instantáneos en tu mano.</td>
<td>VENTAJA COMPETITIVA INJUSTA</td>
<td rowspan="2">SEGMENTO DE CLIENTES
Personas (adultos) interesados en el tema.
Inversores.

Early adopters:

Inversores.</td>
</tr>
<tr>
<td>MÉTRICAS CLAVE

Crear wallets
Restaurar wallets
Participar en sorteos
Ver los sorteos en los que se está participando, así como en los que ya se ha participado.
Realizar envío de bitcoins.
Gestionar movimientos del wallet.
Alertas por haber ganado un sorteo.
Ingreso de premios.</td>
<td>CANALES

Anuncios de vídeo en Youtube.
Banners.
Anuncios tradicionales (prensa y/o televisión).
Redes sociales.
Contacto directo (boca a boca).</td>
</tr>
<tr>
<td colspan="3">ESTRUCTURA DE COSTES

Gastos de desarrollo, mantenimiento y mejora de BlockLotto.
Publicidad.
Cuenta de Google Developer.</td>
<td colspan="3">FLUJO DE INGRESOS

Comisión del 0,2% del premio de cada sorteo celebrado.</td>
</tr>
</table>

Tabla 2 - Lean Canvas

5.5 Utilidad y Viabilidad

Anteriormente se ha nombrado el segmento de clientes al que va dirigida la aplicación. Este segmento es bastante amplio ya que, básicamente, es todo tipo de personas interesados en criptomonedas (desde la realización de transacciones sin ningún tipo de costes hasta la posibilidad de multiplicar sus fondos). Además, la aplicación cuenta con una interfaz bastante sencilla e intuitiva, garantizando la fluidez y el fácil aprendizaje de su uso por parte del usuario. Del mismo modo, la aplicación cuenta con mensajes de ayuda y/ o errores bastante explicativos para que los usuarios no duden en ningún momento de las acciones que realizan en la misma. Igualmente, la app no sólo cuenta con una estructura intuitiva, sino que, además, cuenta con una gama cromática que facilita la percepción de cada componente a todos los usuarios (incluso hasta personas con daltonismo).

Así, esta aplicación permite acercar estas tecnologías a un mayor número de personas facilitando su uso y reduciendo los posibles gastos. Todas las funciones implementadas actualmente en la aplicación permiten la puesta del producto en el mercado. Sin embargo, el servicio que gestiona las loterías y se encarga de realizar los pagos no está trabajando directamente sobre el protocolo Lightning Network ya que, este protocolo aún no se ha desarrollado completamente. No obstante, se ha hecho una simulación de este protocolo para el desarrollo del LottoService (ver TFT de Jorge Fernández Molines). A pesar de ello, la aplicación no depende directamente del protocolo Lightning Network, ya que es el servicio el que se encarga de las gestiones.

El desarrollo funcional de la aplicación se ha realizado mediante softwares gratuitos (Android Studio, SoapUI…), esto es una clara ventaja que favorece la viabilidad del producto, puesto que no es necesario pagar licencias software reduciendo así los gastos. Sin embargo, sí que se ha hecho uso de softwares de pago. La aplicación cuenta con un logo, una splash screen conformada por un gif y, además, cada uno de los sorteos cuenta con un icono distintivo. Todo ello se ha realizado mediante diferentes aplicaciones de Adobe (Photoshop, Illustrator, After Effects), esto contaría con un gasto por adquisición de software específico. Este gasto no sería muy elevado, ya que Adobe cuenta con planes orientados a empresas que incluyen todas sus aplicaciones por 69,99€ al mes y, su utilización no será necesaria durante un gran período de tiempo, ya que su uso es sólo para fines estéticos que no influyen en el funcionamiento de la aplicación.

Asimismo, para realizar las pruebas se puede usar el emulador de dispositivos móviles que ofrece Android Studio o, simplemente es necesario un smartphone (con la versión 6.0 Marshmallow o superior) y hoy en día todos tenemos uno y, en caso de no poseerlo, se pueden conseguir por un precio razonable en múltiples establecimientos.

Por todo ello, el mantenimiento de la misma, así como la ampliación de funcionalidades y posibles extensiones (ver *Posibles Extensiones*) no supondrán un gasto elevado, puesto que se haría uso de hardware ya adquirido y software gratuito.

De esta manera, este producto es viable para empresas que se dediquen al desarrollo de software, ya que los gastos no son elevados y las ganancias están aseguradas.

5.6 Riesgos

En todo proyecto existe la posibilidad de que sucedan eventos y/o amenazas que provoquen situaciones no deseadas. Asimismo, los riesgos siempre implican incertidumbre (ya que puede ocurrir o no) y pérdida potencial (puesto que si el riesgo se hace realidad se producirán acontecimientos no deseados).

Por tal motivo, tiene gran importancia la gestión de riesgos en proyectos software. Dicha gestión tiene como objetivos: identificar, controlar y eliminar las fuentes de riesgo antes de que impidan lograr los objetivos del proyecto. Una vez identificados los riesgos es imprescindible establecer un plan para evitarlos, así como implantar otro, para que una vez se haya producido el riesgo, el proyecto no sufra secuelas (o al menos sean mínimas).

De esta forma, en el proceso de análisis de este TFT se incluyó un procedimiento de detección y gestión de riesgos. Así que, se han identificado una serie de posibles amenazas que pueden afectar a nuestra app. En la tabla 3 se expone cada uno de ellos, así como la probabilidad de que suceda, el impacto que supondría y el plan de mitigación correspondiente a cada uno de ellos.

Taxonomía del riesgo	Riesgo	Descripción	Probabilidad (%)	Impacto (días)	Plan de contingencia
Requisitos (Técnico)	Falta de requisitos	Para la realización de este proyecto se han establecido una serie de requisitos anteriormente explicados. Sin embargo, es común que los usuarios al utilizar frecuentemente la aplicación demanden más funcionalidades produciendo así, un incremento de requisitos en la aplicación.	80	15	Para prevenir la falta de requisitos se debe hacer un análisis exhaustivo de las posibles funcionalidades y capacidades que el usuario pueda necesitar. Además, se hará uso de un ciclo de vida del software que permita el incremento de los requisitos establecidos previo a la implementación del producto final. Asimismo, en el proceso de implementación se utilizarán técnicas y metodologías que permitan añadir nuevas funcionalidades fácilmente.
Mercado (Externo)	Competencia	El mercado es muy amplio y, como se vio en el apartado *Estudio de Aplicaciones Similares*, existen varias aplicaciones que proporcionan algunas cualidades parecidas a la de BlockLotto. De esta manera, ninguna ofrece todas las de nuestra app en su conjunto. A pesar de ello, podría ocurrir que en el proceso de desarrollo o, incluso, una vez finalizada la aplicación, salga otra aplicación	50	5	Prevenir este riesgo es bastante complicado ya que es un riesgo externo. Sin embargo, durante la fase de análisis del proyecto se debe hacer un estudio muy completo de los posibles competidores estudiando así, las carencias de los mismos para solventarlas en nuestro proyecto. En caso de que, inesperadamente, una vez puesto al mercado nuestro producto, aparezcan competidores, se harán encuestas de satisfacción sobre la aplicación a nuestros clientes, pudiendo así

		con las mismas características al mercado.			mejorar nuestros servicios, así como satisfacer las necesidades de los clientes. Además, para atraer nuevos clientes se aumentará la publicidad de la app.
Recursos (Organizativo)	Enfermedad	Durante la elaboración del proyecto el desarrollador puede ponerse enfermo, imposibilitando así el progreso de las tareas y/o la finalización del proyecto.	50	20	Dado que esto es un proyecto de fin de grado y sólo existe un desarrollador de la aplicación, para prevenir este riesgo es importante que el desarrollador se sienta a gusto, motivado, siguiendo un horario de trabajo y descanso y, dividiendo el trabajo en tareas de forma que se desarrollen algunas de ellas cada día (dependiendo de la dificultad). Para mitigar este riesgo se puede reducir el número de funcionalidades ofrecidas desde un comienzo por la app, para no saturar al desarrollador cuando se incorpore o, en caso de ocurrir en la recta final del proyecto publicar la aplicación con las funcionalidades implementadas.
Calidad (Técnico)	Calidad del producto final	Al no tener experiencia en el desarrollo de aplicaciones móviles, puede ocurrir que el producto final no esté a la altura de las	30	10	Para prevenir este riesgo es importante que el desarrollador complete diversos tutoriales con las tecnologías que se utilizarán. Además, en estos tutoriales

		expectativas generadas en el comienzo del proyecto.			debe imitar algunas interfaces existentes en otras apps para obtener conocimientos suficientes en el proceso de implementación. Para mitigar este riesgo es importante que se utilicen metodologías y técnicas de ingeniería del software de manera que, si el resultado obtenido no es el deseado pueda fácilmente realizarse cambios en el código.
Regulatorio (Externo)	Cambio de legislación	Actualmente se está estudiando la incidencia fiscal de nuevas tecnologías, como blockchain, y, en especial, las criptomonedas. Si las leyes cambian podrían suponer un gasto monetario adicional.	70	7	Esto no es un riesgo que podamos evitar, sin embargo, en caso de que se produzca habría que realizar los pagos oportunos. Si no se dispone del dinero necesario para ello, se podría aumentar la comisión del premio en la celebración de cada sorteo.

Tabla 3 - Gestión de Riesgos

5.7 Herramientas Utilizadas y Licencias Softwares

Para el desarrollo de este proyecto se hace uso de software de terceros. A continuación, se exponen los programas y aplicaciones utilizados a lo largo del desarrollo de la aplicación, así como sus características más importantes y sus respectivas licencias:

- **Android Studio**

El entorno de desarrollo utilizado ha sido Android Studio. Este entorno es perfecto para el desarrollo de aplicaciones Android (ya que es el oficial). Está basado en IntelliJ IDEA, por tal motivo cuenta con todas las ventajas que nos proporcionan sus herramientas [14]. Este entorno facilita no sólo la implementación de funcionalidades para las aplicaciones, sino que, además, simplifica la elaboración de una buena e intuitiva interfaz. Igualmente, existen infinidad de librerías desarrolladas para Android. Esto permite que el desarrollo en este entorno sea mucho más sencillo gracias a la cantidad de utilidades disponibles para el mismo.

Además, cuenta con un entorno emulador que permite probar la aplicación en el proceso de desarrollo. Sin embargo, en este proyecto no se ha hecho uso de esta funcionalidad, es decir, en el proceso de implementación de la aplicación se ha utilizado un dispositivo móvil para realizar las pruebas pertinentes.

Asimismo, se ha utilizado Android SDK 23, es decir, Android 6.0 Marshmallow (API 23). Por ello, la aplicación sólo es compatible para dispositivos Android con esta versión y superiores.

Android Studio no genera ningún tipo de impedimento en la comercialización de productos desarrollados con dicho software.

- **SoapUI**

SoapUI es una herramienta que permite realizar pruebas a aplicaciones con arquitectura orientada a servicio (SOA) y transferencia de estado representacional (REST). Además, soporta infinidad de protocolos entre ellos están SOAP, REST, HTTP, JMS, etc. [15]. SoapUI cuenta con dos versiones: SoapUI Open Source y SoapUI Pro [16]. En este proyecto se ha hecho uso de la versión de código abierto durante el proceso de integración del LottoService en la aplicación. Así, se ha usado para realizar peticiones al servicio y comprobar no sólo que las peticiones se realizan correctamente, sino que, además, devuelve los datos necesarios para la app. Además, ha sido de gran ayuda en la detección de errores, ya que con esta herramienta es bastante sencillo comprobar si el error se encuentra en la aplicación o en el servicio.

- **MySQLWorkBench 8.0**

MySQLWorkBench es una herramienta visual para diseñar bases de datos. Esta herramienta es de uso gratuito. Se ha utilizado para comprobar el correcto funcionamiento de la aplicación cuando utiliza las funcionalidades ofrecidas por el servicio.

- **Adobe:**

En este proyecto se ha hecho uso de algunos productos de Adobe:

- **Adobe Photoshop**

  Adobe Photoshop es un editor gráfico, usado principalmente para el retoque de fotografías.[17]

- **Adobe Illustrator**

  Adobe Illustrator es una herramienta que permite crear de forma sencilla e intuitiva ilustraciones y gráficos. [18]

- **Adobe After Effects**

  Adobe After Effects es una aplicación destinada a la creación de composiciones, así como la realización de gráficos profesionales en movimiento, incluyendo la posibilidad de incorporar efectos especiales. [19]

Así, en el desarrollo de la aplicación se ha hecho uso de Photoshop e Illustrator para la realización tanto del logo de la app, como de los iconos representativos de cada sorteo. Y, se ha utilizado After Effects para la realización de la animación inicial de la app. Adobe ofrece diferentes planes de contratación para cada una de estas herramientas. Existen diferentes planes: para individuos, para empresas, para estudiantes y profesores y, para colegios y universidades. Además, para cada uno de ellos existen diferentes opciones de adquisición variando las aplicaciones incluidas en la compra y, por consiguiente, su precio [20]. Sin embargo, para la realización del logo de la app, los emblemas de los sorteos y el gif inicial de la aplicación se ha utilizado el periodo de prueba gratuito que ofrecen cada uno de los programas descritos. Este periodo de prueba tiene una duración de siete días para cada una de las herramientas comentadas. Se consideró que con este tiempo era suficiente para realizar todos los componentes necesarios y, efectivamente fue suficiente.

- **COLOURlovers**

COLOURlovers es una comunidad creativa donde diferentes personas crean y comparten paletas y patrones, así como discuten sobre ellos. Además, los usuarios pueden indicar el código hexadecimal de cada color o, incluso, las imágenes que han utilizado para inspirarse en la creación de las mismas. Registrarse en esta web es totalmente gratuito, así como su utilización [21]. En este proyecto se ha hecho uso de esta web para la selección de los colores utilizados en la aplicación. Gracias a esta página web se ha encontrado una gama cromática con colores que permanecen en armonía y transmiten seguridad, confianza y equilibrio a los usuarios [22]. Estos son aspectos

bastante importantes a comunicar de manera indirecta por la app. De este modo, se ha hecho uso de la paleta Aurora creada y publicada por el usuario By Tabuu [23].

- **Colorblinding**

Colorblinding es una extensión para el navegador Google Chrome. Esta extensión simula sitios webs tal y como los vería una persona con discapacidad visual del color (Protanopia, Deuteranopia, Tritanopia, etc.) [24]. Asimismo, es un proyecto de código abierto y, por ello, esta extensión es totalmente gratuita. Colorblinding se ha utilizado como apoyo para seleccionar la gama cromática adecuada a utilizar en la aplicación. De manera que, todos los usuarios puedan distinguir correctamente cada uno de los componentes y elementos de la app sin dificultad.

- **Flaticon**

Flaticon es un sitio web que ofrece a los usuarios diseños gráficos de alta calidad: iconos vectoriales totalmente editables, seleccionados por un equipo de diseño con el fin de proporcionar a los usuarios un gran contenido utilizable tanto en proyectos personales como comerciales. En esta web se pueden diferenciar dos tipos de iconos: los premium y los gratuitos [25]. Durante el desarrollo de este proyecto se ha utilizado este sitio web para obtener iconos que no se encontraban en las librerías incluidas en Android Studio. De esta manera, se ha hecho uso de diferentes iconos (todos ellos gratuitos) proporcionados por la web.

- **QR Code Generator**

QR Code Generator es una página web que genera códigos QR con información establecida por el usuario (URLs, textos, e-mails, etc.). Además, QR Code Generator te ofrece ideas y posibles funcionalidades para tus webs, aplicaciones, etc. y, de esta forma, sacar mejor partido a los códigos QR. Este sitio web es de uso gratuito. Se ha utilizado para generar códigos QR, los cuales han permitido la realización de pruebas de la aplicación en el proceso de escaneo.

- **Trello**

Trello es una página web que permite organizar las tareas a realizar en un proyecto. Además, se puede utilizar en equipo. El uso de Trello es totalmente gratuito [26]. Se ha utilizado esta herramienta para la organización de las tareas y, además, para la comunicación entre ambos componentes de BlockLotto. Así, esta herramienta ha sido muy útil para comunicar los errores y fallos encontrados tanto en el servicio como en la app.

- **Sourcetree**

Sourcetree es un programa que simplifica la forma a interactuar con los repositorios Git. Gracias a su sencilla interfaz se hace mucho más fácil el uso de Git y el control de versiones de nuestro proyecto. Este software es totalmente gratuito [27]. Se ha hecho uso de este programa para llevar a cabo el control de versiones en el repositorio Github

de nuestro proyecto, realizando los correspondientes “commits” y “push”, así como creando la rama “slave” en la que se ha desarrollado todo el proyecto.

- **Github**

Github es una plataforma de desarrollo colaborativo que permite alojar proyectos utilizando el sistema de control de versiones Git [28]. La utilización de esta plataforma es totalmente gratuita, aunque existen algunas funcionalidades y planes orientados a empresas y/o al comercio que son de pago. Sin embargo, se ha utilizado de forma gratuita. En este proyecto se ha hecho uso de un repositorio en Github en el que se ha alojado el código fuente de la aplicación, encontrándose en él todas las versiones del proyecto.

Asimismo, se ha utilizado para la implementación del proyecto mi ordenador portátil:

Especificaciones	
Dispositivo	Portátil Lenovo G70-35
Procesador	AMD A8-6410 APU with AMD Radeon R5 Graphics 2.00 GHz
RAM	8,00 GB
Sistema Operativo	Windows 10 Home de 64 bits

Tabla 4 - Especificaciones del dispositivo utilizado para el desarrollo

Y, para la realización de pruebas del proyecto (tanto para comprobar que la interfaz era la deseada como para la realización de pruebas de funcionamiento) se ha utilizado un dispositivo móvil personal:

Especificaciones	
Dispositivo	Redmi Note 3 Pro
Procesador	Snapdragon 650
RAM	2,00 GB
Sistema Operativo	**Versión MIUI:** MIUI 10 Global 8.11.22 **Versión Android:** 6.0.1 MMB29M

Tabla 5 - Especificación del smartphone utilizado para el desarrollo y pruebas

6. Diseño

Una vez finalizada la fase de análisis dónde se establecieron las características, funcionalidades y requisitos software del proyecto, comenzó el proceso de diseño. En este apartado se describirán los distintos componentes utilizados en dicho proceso.

6.1 Diagrama de Casos de Uso

Basándonos en los requisitos establecidos en la fase anterior se elaboró un diagrama de casos de uso. En la Ilustración 3 se ven reflejadas las acciones que puede realizar un usuario en la aplicación.

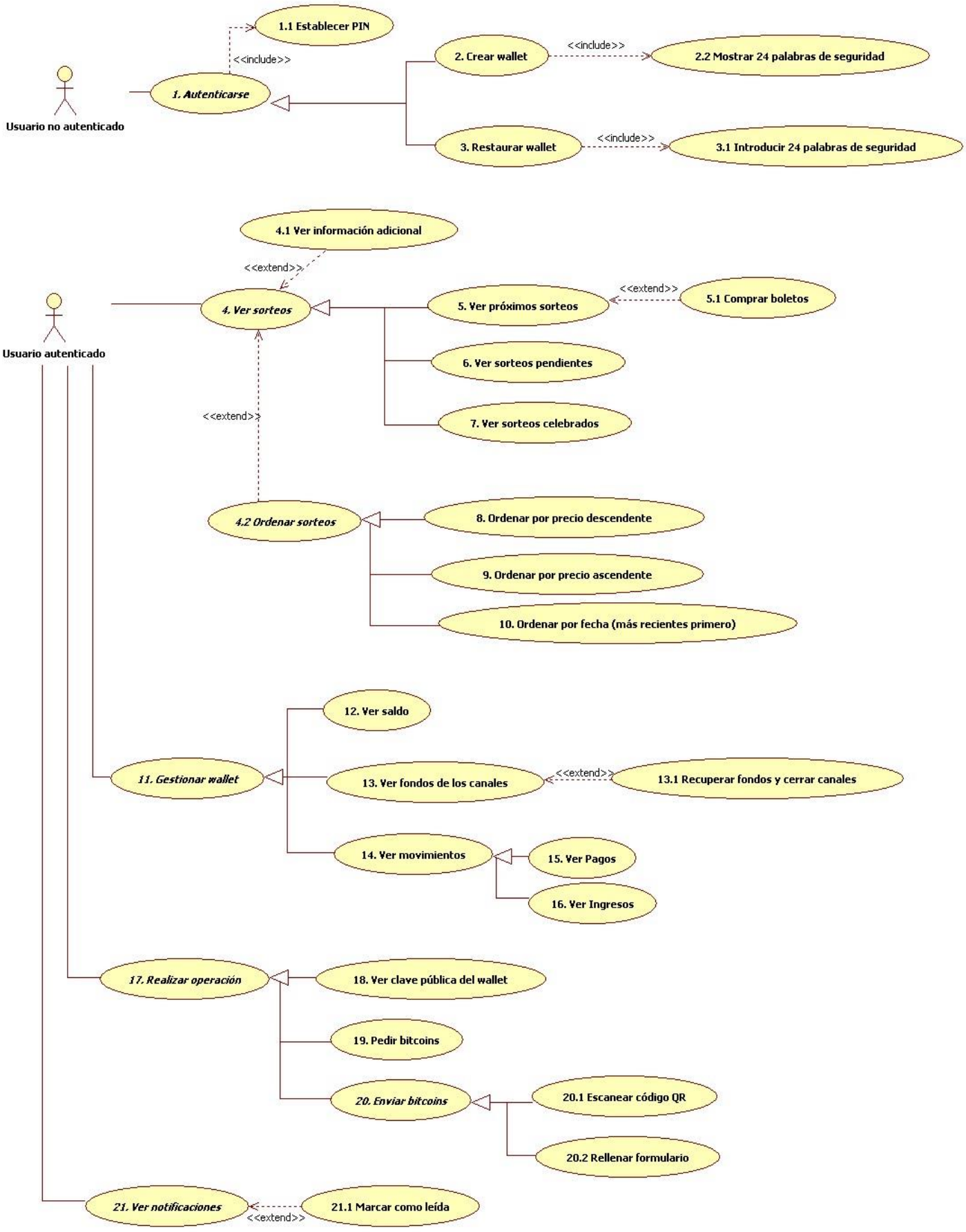


Ilustración 3 - Casos de Uso

6.2 Especificación de Casos de Uso

A continuación, en las siguientes tablas se explica la especificación de algunos casos de uso más relevantes:

CASO DE USO 2	**Crear wallet**		
Descripción	El usuario necesita poseer un wallet para utilizar la aplicación.		
Actor	Usuario.		
Precondiciones			
Flujo normal	**Paso**	**Acción**	
	1	El usuario pulsa sobre el botón para comenzar el proceso de creación del wallet.	
	2	El usuario establece el PIN del wallet.	
	3	El sistema crea el wallet.	
	4	El sistema muestra las 24 palabras de seguridad correspondientes al wallet.	
Postcondiciones	Se guardan las propiedades del wallet (dirección, id y clave pública) en el dispositivo, y el sistema muestra al usuario la pantalla de inicio de la aplicación dónde podrá empezar a utilizar nuestros servicios.		
Variaciones	**Paso**	**Acción**	
Extensiones	**Paso**	**Condición**	**Caso de Uso**
Excepciones	3	El dispositivo no tiene acceso a Internet.	
Observaciones	Para crear el wallet el sistema necesita tener acceso a Internet, en caso de no tener, el sistema informará al usuario para que lo intente posteriormente cuando disponga de Internet.		

Tabla 6 - Especificación de Crear wallet

CASO DE USO 3	**Restaurar wallet**	
Descripción	El usuario necesita un wallet para utilizar la aplicación. En este caso, el usuario accede a uno creado en un instante pasado al que actualmente no tiene acceso.	
Actor	Usuario.	
Precondiciones		
Flujo normal	**Paso**	**Acción**
	1	El usuario pulsa sobre el botón para comenzar el proceso de restauración del wallet.

	2	El usuario introduce las 24 palabras de seguridad correspondientes a su wallet.	
	3	El usuario establece el PIN de su wallet.	
	4	El sistema restaura el wallet.	
Postcondiciones	Se guardan las propiedades del wallet (dirección, id y clave pública) en el dispositivo, y el sistema muestra al usuario la pantalla de inicio de la aplicación dónde podrá empezar a utilizar nuestros servicios.		
Variaciones	**Paso**	**Acción**	
	4.1	El sistema no restaura el wallet.	
	4.1.1	El sistema redirige al usuario al paso 1.	
Extensiones	**Paso**	**Condición**	**Caso de Uso**
Excepciones	4	El dispositivo no tiene acceso a Internet.	
Observaciones	Para restaurar el wallet el sistema necesita que las 24 palabras de seguridad correspondan a algún wallet existente. Además, el sistema necesita tener acceso a Internet, en caso de no tener, el sistema informará al usuario para que lo intente posteriormente cuando disponga de Internet.		

Tabla 7 - Especificación de Restaurar wallet

CASO DE USO 5	**Ver próximos sorteos**		
Descripción	El usuario desea ver los próximos sorteos que se celebrarán en el sistema para participar en ellos.		
Actor	Usuario autenticado.		
Precondiciones	El usuario posee un wallet.		
Flujo normal	**Paso**	**Acción**	
	1	El usuario inicia la aplicación.	
	2	El sistema muestra los sorteos disponibles en el sistema.	
Postcondiciones			
Variaciones	**Paso**	**Acción**	
	1.1	El usuario ya ha iniciado la aplicación.	
	1.1.1	El usuario pulsa el botón del menú.	
	1.1.2	El usuario selecciona la opción “Sorteos”.	
	1.1.2	El sistema muestra los sorteos disponibles en el sistema	
Extensiones	**Paso**	**Condición**	**Caso de Uso**
	2	El usuario desea comprar boletos.	CU 5.1: Comprar boletos
Excepciones	2	El dispositivo no tiene acceso a Internet.	

Observaciones	Para mostrar los sorteos activos en el sistema el dispositivo necesita tener acceso a Internet, en caso de no tener, el sistema informará al usuario para que lo intente posteriormente cuando disponga de Internet.

Tabla 8 - Especificación de Ver próximos sorteos

CASO DE USO	**5.1**	**Comprar boletos**	
Descripción	El usuario desea participar en un sorteo.		
Actor	Usuario autenticado.		
Precondiciones	El usuario posee un wallet.		
Flujo normal	**Paso**	**Acción**	
	1	El usuario pulsa sobre el botón comprar.	
	2	El usuario selecciona la cantidad de boletos que desea comprar.	
	3	El usuario realiza la compra.	
	4	El sistema solicita confirmación de compra.	
	5	El usuario confirma la compra.	
Postcondiciones	La compra queda registrada en el sistema.		
Variaciones	**Paso**	**Acción**	
Extensiones	**Paso**	**Condición**	**Caso de Uso**
Excepciones	3	El dispositivo no tiene acceso a Internet.	
	3	El usuario no posee suficiente saldo en el wallet para realizar la compra.	
Observaciones	Para realizar la compra de boletos el sistema necesita tener acceso a Internet, en caso de no tener, el sistema informará al usuario para que lo intente posteriormente (cuando disponga de Internet).		

Tabla 9 - Especificación de Comprar boletos

CASO DE USO	**13**	**Ver fondos de los canales**
Descripción	El usuario desea ver los fondos almacenados en canales.	
Actor	Usuario autenticado.	
Precondiciones	El usuario posee un wallet. Y, el usuario debe tener canales abiertos.	
Flujo normal	**Paso**	**Acción**
	1	El usuario inicia la aplicación.
	2	El usuario pulsa el botón menú.
	3	El usuario selecciona la opción “Wallet”.

	4	El sistema muestra al usuario sus fondos almacenados en canales.	
Postcondiciones			
Variaciones	**Paso**	**Acción**	
	1.1	El usuario ya se encuentra dentro de la opción “Wallet”.	
	1.2	El sistema muestra al usuario sus fondos almacenados en canales.	
Extensiones	**Paso**	**Condición**	**Caso de Uso**
	4	El usuario desea recuperar sus fondos y cerrar todos los canales que tiene abiertos.	CU 13.1: Recuperar fondos y cerrar canales abiertos
Excepciones	2	El dispositivo no tiene acceso a Internet.	
Observaciones	Para mostrar los fondos del usuario almacenados en los canales el sistema necesita tener acceso a Internet, en caso de no tener, el sistema informará al usuario para que lo intente posteriormente cuando disponga de Internet.		

Tabla 10 - Especificación de Ver fondos de los canales

6.3 Diagrama de Clases

Además, en la etapa de diseño se ha realizado un diagrama de clases. Esto ha permitido describir de manera esquemática la estructura del sistema a implementar, facilitando su posterior elaboración.

En dicho diagrama se muestran los atributos y métodos más importantes que, posteriormente, han sido implementados en la aplicación. Sin embargo, estos no son todos los que conforman las clases de la aplicación, esto se ha hecho así, ya que el diagrama quedaría caótico.

En la siguiente Ilustración se puede observar el diagrama de clases realizado para este proyecto:

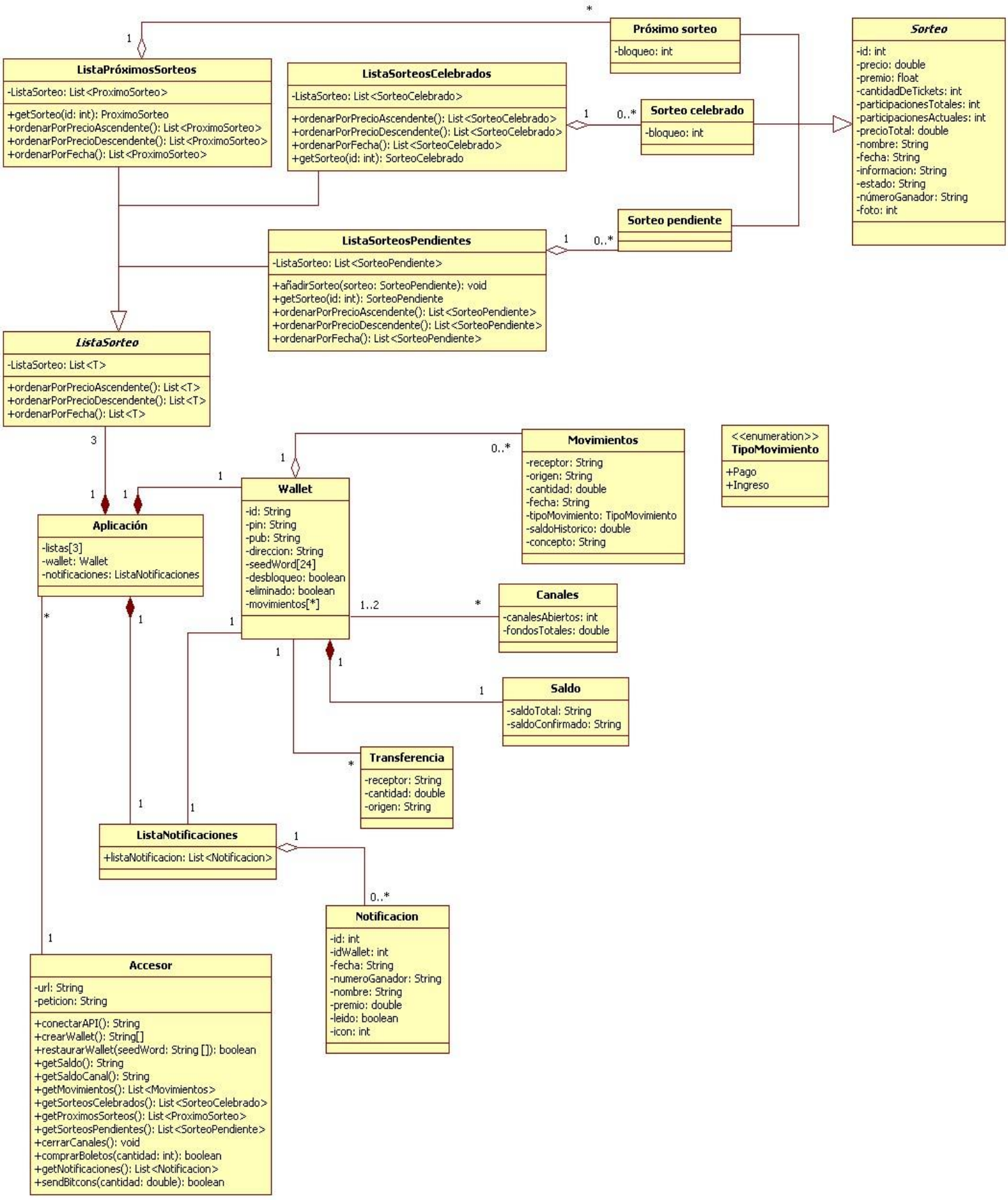


Ilustración 4 - Diagrama de Clases

6.4 Diseño del Convenio con LottoService

Un aspecto bastante importante de este proyecto es el acuerdo entre ambas partes de BlockLotto. Esto se realizó mediante el diseño de un contrato. Dicho contrato es necesario ya que, el servicio ofrece información que la aplicación debe mostrar al usuario a través de la interfaz.

En la elaboración de este convenio se pensó en cada uno de los componentes que intervendrían en la aplicación, así como algunas funcionalidades que el servicio ofrecería.

Sin embargo, el pacto final difiere en algunos aspectos a lo que se había acordado en un comienzo. Esto se debe a que la elaboración de un convenio, sin haber realizado previamente el diseño de la interfaz, basándonos exclusivamente en los diagramas de clases y de casos de uso, es un proceso bastante complicado (sobre todo para estudiantes sin experiencia alguna en este tipo de proyectos). No obstante, se realizó un primer pacto, el cual estaba bien enfocado (salvo algunos aspectos). Seguidamente, se comenzó con el diseño de la interfaz y del servicio. Y, al mismo tiempo que se desarrollaba este diseño, se fue observando (desde ambas partes de BlockLotto) algunas características que la aplicación debía poseer y, que aún no estaba estipulado en el convenio. Entre ellos están: falta de información en los sorteos, imágenes no almacenadas en el servidor o mayor detalle y especificación en los movimientos del wallet.

Asimismo, se llegó a un convenio entre ambas partes. A continuación, se expone el convenio final:

Draws

Next Lotteries

Attributes:

```
int id;
double price;
float award;
int amountTicket;
int totalParticipations;
int currentParticipations;
double priceTotal;
int blocked;
String name;
String date;
String information;
String state;
String winningNumber;
```

Functions:

```
List<NextLotteries> getList();

boolean buyTickets(int idWallet,
int idRaffle, int
amountTickets);
```

Pending Lotteries

Attributes:

```
int id;
double price;
float award;
int amountTicket;
int totalParticipations;
int currentParticipations;
double priceTotal;
String name;
String date;
String information;
String state;
String winningNumber;
```

Functions:

```
List<PendingLotteries>   getList
(int idWallet);
```

Celebrated Lotteries

Attributes:

```
int id;
double price;
float award;
int amountTicket;
int totalParticipations;
int currentParticipations;
double priceTotal;
int blocked;
String name;
String date;
String information;
String state;
String winningNumber;
```

Functions:

```
List<CelebratedLotteries>getList
(int idWallet);
```

Wallet

Attributes:

```
String id;
String pass;
String pub;
String address;
int amount;
String[] seedWords;
boolean unlock;
boolean deleted;
```

Functions:

```
Wallet       createWallet(String
pass);

Wallet recuperationWallet(String
pass, String[] seedWords);
```

Balance

Attributes:

```
String total_balance;
String confirmed_balance;
```

Functions:

```
String getTotalBalance(int
idWallet);
```

Transactions

Attributes:

```
String to;
String from;
double amount;
String date;
String action;
double historicalAmount;
String concept;
```

Functions:

```
List<Transaction>
getTransactions(int idWallet);
```

Channels

Attributes:

```
double total_balance;
double confirmed_balance;
double unconfirmed_balance;
```

Functions:

```
Double getBalanceChannels(int
idWallet);
Balance getBTCFromChannels(int
idWallet);
```

SendBitcoins

Functions:

```
String sendBTC(int idWallet, int toWalletId, double amount);
```

Notifications

Attributes:

```
int id;
String codWallet;
String date;
String winningNumber;
String name;
double award;
int readed;
```

Functions:

```
List<Notification>
getNotifications(int idWallet);
String setReadNotification (int
idNotification);
```

Ilustración 5 - Convenio LottoApp-LottoService

Hay que añadir, que la comunicación entre aplicación y API se realizan mediante el protocolo JSON. Por tanto, aunque se hayan establecido funciones, no quiere decir que se siga esa técnica para utilizar las funcionalidades de la API, son meramente teóricas. De esta manera, la aplicación se ha desarrollado haciendo uso de este contrato.

6.5 Mockups

Una vez planteados los diagramas de clases y casos de uso y, diseñado un primer contrato con LottoService se prosiguió a diseñar un mockup de la aplicación. Esto se realizó para tener una idea visual y gráfica de cómo quedaría la aplicación. En las siguientes ilustraciones se expone el primer diseño:

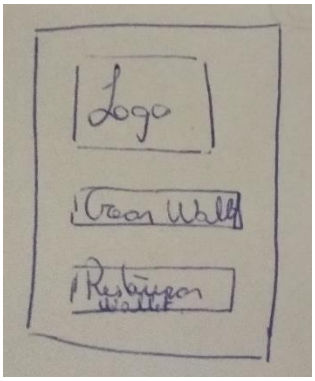


Ilustración 6 - Pantalla inicial de la app

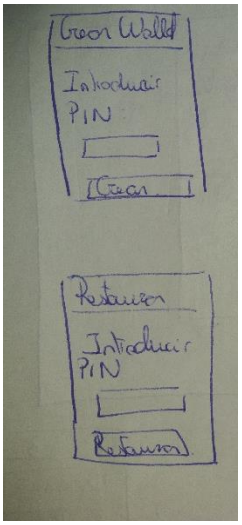


Ilustración 7 - Creación y restauración del wallet

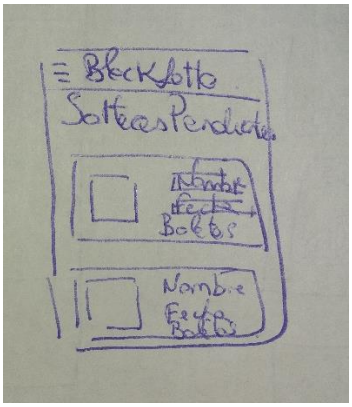


Ilustración 8 - Pantalla inicial tras haber obtenido un wallet

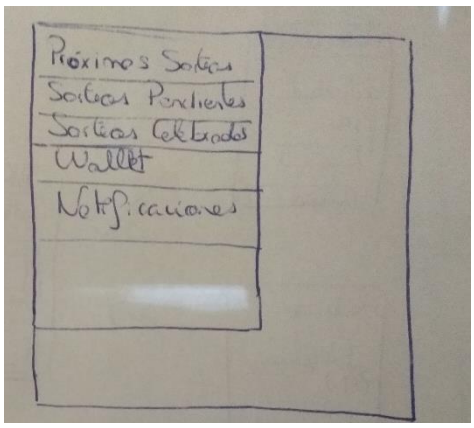


Ilustración 9 - Menú

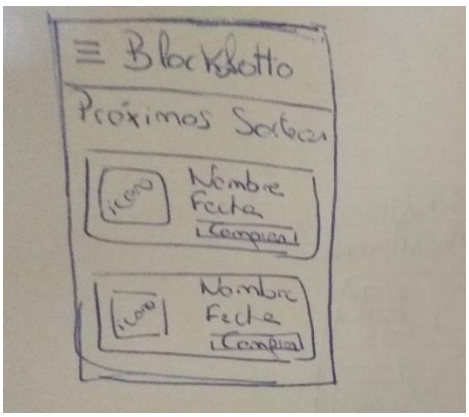


Ilustración 10 - Pantalla "Próximos Sorteos"

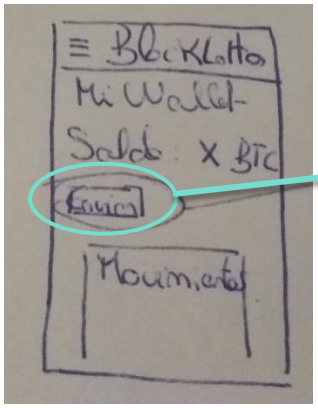


Ilustración 11 - Wallet, operaciones y movimientos

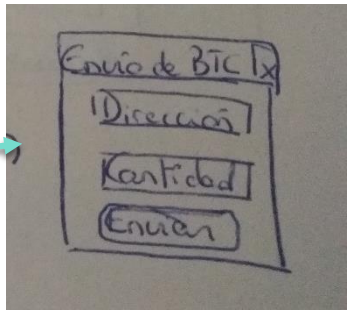


Ilustración 12 - Pantalla de envío de bitcoins

En este primer diseño la pantalla de inicio contaría exclusivamente con dos opciones “Crear” o “Restaurar” un wallet. Y, cuando se seleccionaba cualquiera de las dos opciones simplemente se requería un PIN. Posteriormente al primer diseño, analizando e investigando se descubrió que Lightning Network sigue un mecanismo de seguridad basado en 24 palabras. Estas 24 palabras son únicas, es decir, no existe más de un wallet con la misma secuencia de palabras. Además, en esta secuencia es muy importante el orden de las mismas, ya que, aunque la secuencia contenga las mismas palabras, dos ristras se diferenciarán en el orden de cada una de ellas. Por tal motivo, se tuvo que modificar esta pantalla.

Del mismo modo, en este diseño cuando el usuario crea/restaura un wallet la aplicación lo redirige a la opción “Sorteos Pendientes”. Tras la realización de este diseño se observó que esto no tendría sentido, puesto que en el caso de haber creado el wallet, el usuario no posee ninguna compra y, por ello, se redirigiría a una pantalla en blanco.

Además, el menú mostraba como opciones los diferentes tipos de sorteos. Esto sería bastante incómodo para el usuario final, ya que cuando realiza una compra y desea verla, tendría que ir al menú y buscar la opción deseada. Asimismo, se consideró añadir una nueva funcionalidad: mostrar el historial de todos los sorteos celebrados en el sistema. Sin embargo, no se añadió porque se llegó a la conclusión de que no era una necesidad que el usuario pudiera tener.

Por otro lado, se decidió que no era necesario mostrar al usuario en todo momento el nombre de la app.

De igual forma, el wallet muestra una estructura un poco indecente, dado que muestra excesivas funcionalidades en una misma pantalla y, la opción de "Enviar Bitcoins" se vería incompleta exponiéndose de esta manera.

Por todos estos motivos, se tuvo que rediseñar la aplicación. Y, de este modo, se realizó otro mockup de la misma, corrigiendo todos los errores anteriormente cometidos y aportando un aspecto más profesional a la aplicación. A continuación, se expone el mockup final:

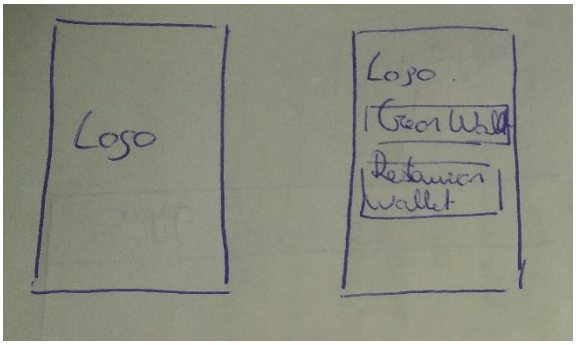


Ilustración 13 - Animación Inicial y Pantalla de inicio

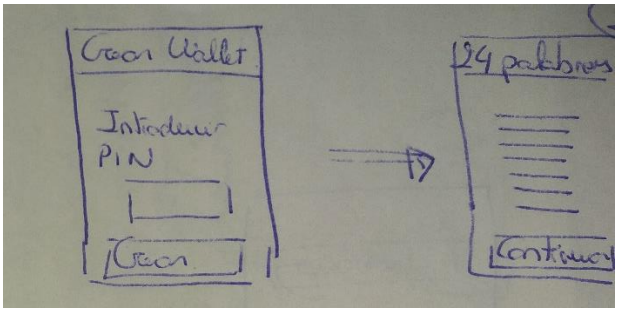


Ilustración 14 - Procedimiento para crear un wallet

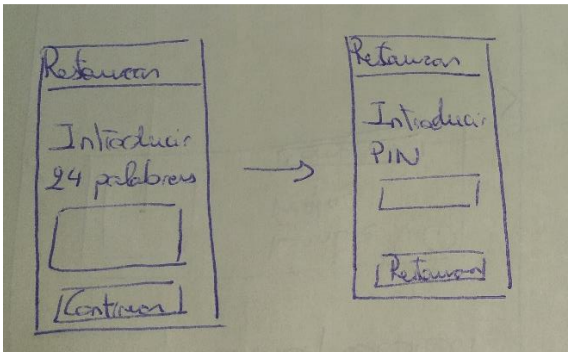


Ilustración 15 - Procedimiento para restaurar un wallet

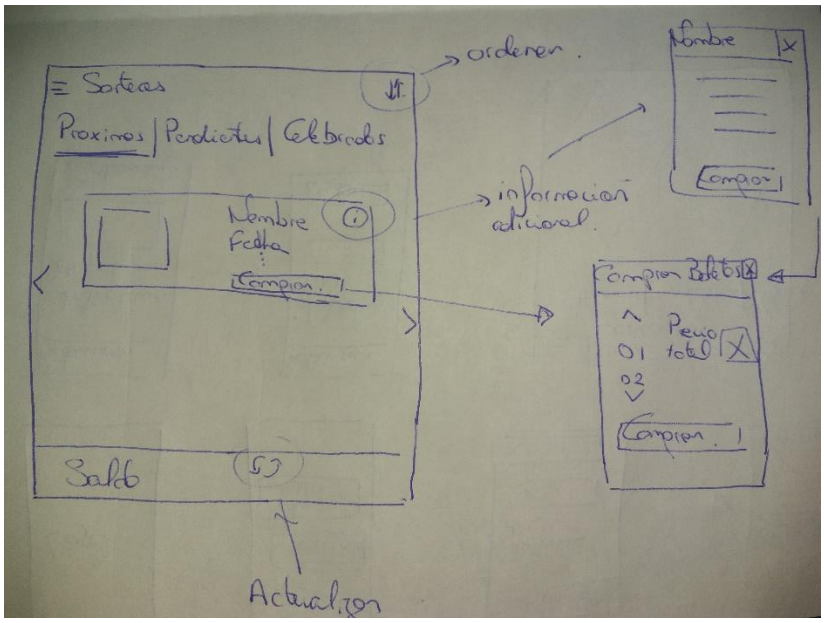


Ilustración 16 - Pantalla principal y proceso de compra

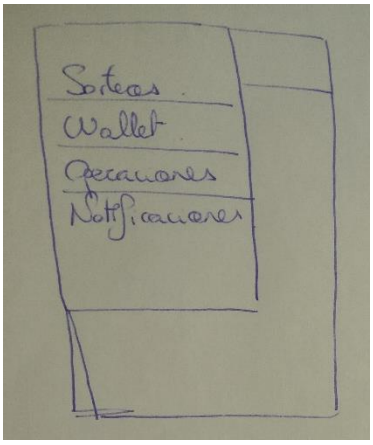


Ilustración 17 - Menú

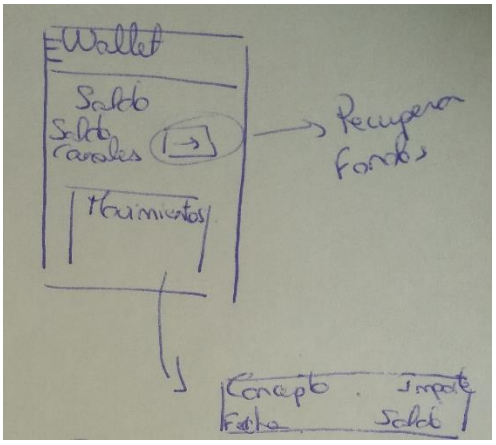


Ilustración 18 - Wallet

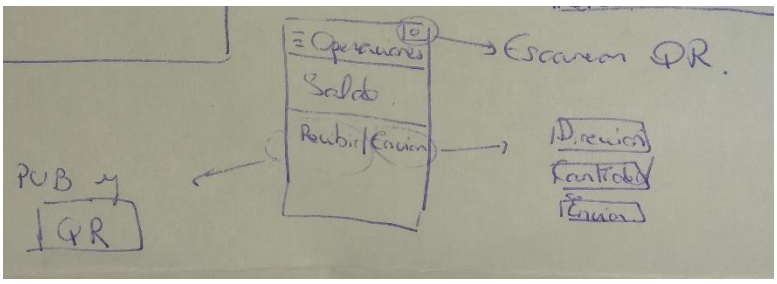


Ilustración 19 - Operaciones (Recibir y Enviar bitcoins)

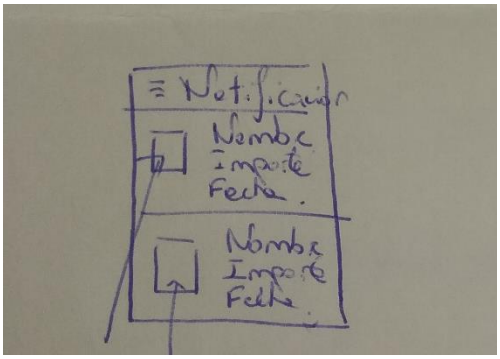


Ilustración 20 - Notificaciones

Con este diseño se corrigieron numerosos errores cometidos en el mockup anterior. Asimismo, se rediseñó la aplicación ofreciendo un aspecto visual más agradable e intuitivo. Además, se estableció como principal de la app la pantalla encargada de mostrar los próximos sorteos que se celebrarían en el sistema, junto con la opción de compra. También se reflexionó sobre la información a mostrar de cada sorteo y, de esta manera, se añadieron más campos explicativos de los mismos.

De la misma manera, se puede ver en la Ilustración 17 como se ha abstraído cada una de las funcionalidades de la aplicación en opciones mucho más simples, abarcando todas ellas, pero ofreciendo al usuario una interfaz más fácil de manejar. Un claro ejemplo:

Si estamos utilizando la aplicación y queremos comprar sorteos, simplemente vamos a la opción “Sorteos”, en ella observaremos los que se celebrarán próximamente en el sistema. Así, se puede realizar la compra de boletos y, con un simple movimiento hacia la izquierda se puede contemplar la compra realizada, así como, todos los sorteos pendientes de celebración en los que el usuario está participando.

Además, la aplicación muestra en todo momento el saldo que el usuario posee en su wallet.

Por otro lado, se añadieron funcionalidades no tenidas en cuenta en el primer mockup. Estas funcionalidades pueden ser: ver los fondos de los canales, así como recuperarlos y cerrar dichos canales.

También, se estableció el aspecto de la pantalla de notificaciones. En ella, se mostraría las notificaciones que el sistema envía al usuario al haber ganado un sorteo (el sorteo que ha ganado, el importe del premio, así como la fecha). Para facilitar el manejo de las mismas al usuario, se pensó en un icono que permita conocer al usuario si la notificación es nueva o, por el contrario, ya la ha leído. Por ello, se permite al usuario cambiar el estado de las mismas a "leída", esto se haría con un simple toque en la notificación correspondiente.

Finalmente, se pensó que enviar bitcoins sería mucho más cómodo para el usuario si se ofrecía la posibilidad de escanear códigos en los que estuviera la clave pública del wallet receptor.

6.6 Elección de gama cromática

Para la implementación de la aplicación fue necesario elegir los colores característicos de la app. De esta manera, se elaboró un proceso de investigación en el que cual, se encontró que impresión daba al usuario cada color [22]. Asimismo, como se pretende que el usuario esté a gusto con la app, aportando toda confianza en ella y sintiendo que su dinero está alojado en un sitio seguro, se eligió una gama cromática en la que predominan el color azul y tonos verdes azulados. La paleta utilizada se ha encontrado en COLOURlovers (Aurora creada y publicada por el usuario By Tabuu [23]).

En la siguiente figura se muestra la gama cromática utilizada:

Ilustración 21 - Gama cromática

Además, se ha hecho uso de Colorblinding para asegurar que todos los usuarios pudiesen utilizar la app sin problemas (incluso los usuarios con dificultad visual del color). En las siguientes ilustraciones se observa como usuarios con esta dificultad visual pueden ver cada uno de los colores sin ningún problema (en las ilustraciones se muestran las dificultades visuales del color más comunes).

Ilustración 22 - Red-Blind / Protanopia

Ilustración 23 - Green-Blind / Deuteranopia

Ilustración 24 - Blue-Blind / Tritanopia

Es de esperar que estos usuarios no vean los colores tal y como lo haría una persona sin esta dificultad. Sin embargo, cada uno de ellos se distingue perfectamente, siendo esto el objetivo que se perseguía.

Además, el fondo de la aplicación se ha establecido en blanco. Esto ha facilitado, aún más, que estas personas puedan distinguir sin ningún problema cada uno de los componentes de la app, a la vez que se transmite a todos los usuarios claridad y transparencia en nuestro negocio.

7. Desarrollo

7.1 Metodología

La etapa de desarrollo de este TFT ha sido la más larga y la más compleja. Asimismo, se ha hecho uso de metodologías ágiles. Las metodologías ágiles son técnicas que agilizan el desarrollo de software, favoreciendo la toma de decisiones, donde los requisitos y soluciones evolucionan con el tiempo según las necesidades de los clientes.

Más concretamente, se ha hecho uso de la metodología iterativa. De esta forma, se han seguido las siguientes iteraciones:

- Implementación de la interfaz de la app.
  - Crear wallet.
  - Restaurar wallet.
  - Mostrar sorteos disponibles en el sistema.
  - Mostrar sorteos pendientes.
  - Mostrar sorteos celebrados.
  - Comprar boletos de lotería
  - Ver saldo del wallet.
  - Ver fondos de los canales.
  - Recuperar fondos acumulados en los canales.
  - Ver movimientos
  - Enviar bitcoins a otro wallet.
  - Notificar si el usuario es ganador.
- Integración API LottoService en la App.

Así, se han ido realizando cada una de ellas, con su posterior prueba y evaluación. Esto ha servido para concretar si la misma estaba correcta o, por el contrario, seguía sin finalizar.

Posteriormente se explicará cada una de ellas con más detalle.

7.2 Planificación Inicial y Final

Antes del comienzo del proyecto se realizó una planificación inicial dónde se estimó la duración de cada una de las fases del mismo. En esta planificación se pasaron muchos aspectos por alto y, por tal motivo, la planificación inicial carece de numerosas tareas que se realizaron, así como, el número de horas que se invirtieron en este proyecto. En la siguiente tabla se puede ver la planificación inicial del proyecto.

Fases	Duración Estimada (horas)	Tareas (nombre y descripción, obligatorio al menos una fase)
Estudio previo/ Análisis	125	Tarea 1.1: Investigación de Block Chain, Lightning y Smart Contract. Qué es, usos y posibles aplicaciones.
		Tarea 1.2: Investigación de desarrollo de Apps
		Tarea 1.3: Determinación de los posibles requisitos (funcionales y no funcionales) de la aplicación.
Diseño/Desarrollo/Implementación	100	Tarea 2.1: Diseño de la arquitectura necesaria para el desarrollo e implementación del proyecto.
		Tarea 2.2: Integración del módulo LottoServices
Evaluación/Validación/Pruebas	38	Tarea 3.1: Realización de pruebas de integración, de sistema, de rendimiento, de carga, de stress, etc.
Documentación/Presentación	37	Tarea 4.1: Realización de un documento explicativo de cada una de las funciones de la aplicación, así como los posibles problemas encontrados.
		Tarea 4.2: Preparación de la presentación ante el tribunal.

Tabla 11 - Planificación inicial

A continuación, se muestra la planificación que se ha llevado a cabo. En ella, se puede observar las tareas realizadas (a groso modo), así como la duración de cada fase. La duración (en horas) de la fase “estudio previo/ análisis” sí que estuvo bien estimada, a pesar de incluir más tareas de las previstas. Sin embargo, en la fase “Diseño/Desarrollo/Implementación” se dedicaron muchas más horas de las previstas, ya que no se poseía experiencia alguna en la creación de logos y/o animaciones. Y, tampoco, se tenía experiencia en el desarrollo de aplicaciones para dispositivos Android.

Todo ello, produjo un aumento en el número de horas invertidas en esta fase. Asimismo, la realización de pruebas se realizó paralelamente a la implementación de la app. Esto se debe a que se ha seguido una metodología iterativa, en la cual se prueba y evalúa cada iteración una vez completada para, así, poder marcarla como "finalizada".

Fases	Duración Estimada (horas)	Tareas (nombre y descripción, obligatorio al menos una fase)
Estudio previo/ Análisis	125	Tarea 1.1: Investigación de Blockchain, Lightning Network y smart contract. Qué es, usos y posibles aplicaciones.
		Tarea 1.2: Investigación sobre la estructura de aplicaciones wallets.
		Tarea 1.3: Determinación de los posibles requisitos (funcionales y no funcionales) de la aplicación.
		Tarea 1.4: Identificación de los posibles riesgos del proyecto y realización del plan de contingencia correspondiente a cada uno de ellos
Diseño/Desarrollo/Implementación	150	Tarea 2.1: Realización de los diagramas pertinentes (diagrama de casos de uso y diagrama de clases).
		Tarea 2.2: Diseño del convenio con LottoService.
		Tarea 2.3: Diseño de la interfaz de la aplicación.
		Tarea 2.4: Creación y edición de logo, iconos representativos de los sorteos y animación inicial de la app.
		Tarea 2.5: Implementación de la interfaz de la aplicación.
		Tarea 2.6: Integración del LottoServices
Evaluación/Validación/Pruebas	55	Tarea 3.1: Evaluación de la interfaz. Tarea 3.2: Evaluación del correcto funcionamiento de las capacidades ofrecidas.
Documentación/Presentación	37	Tarea 4.1: Realización de un documento explicativo de cada una de las funciones de la aplicación, así como los posibles problemas encontrados.
		Tarea 4.2: Preparación de la presentación ante el tribunal.

Tabla 12 - Planificación final

Finalmente, es importante comentar que para organizar las tareas a realizar durante el desarrollo de este proyecto se ha hecho uso de Trello. De esta manera, se ha podido dividir cada tarea en subtareas más sencillas, permitiendo establecer fácilmente que tareas permanecían incompletas, así como las ya finalizadas. Además, esta herramienta ha facilitado la comunicación de errores entre el servicio y la aplicación. En la Ilustración 25 se muestra el tablero utilizado para organizar la elaboración de este proyecto:

Ilustración 25 - Tablero organizativo durante el desarrollo

En la Ilustración 25 se puede observar el tablero durante la fase de desarrollo. En él se pueden ver la lista de tareas del proyecto, así como las subtareas en las que se dividió. Se utilizaron etiquetas de color para indicar el estado de cada una:

- Rojo-> Sin implementar.
- Verde -> Terminada.
- Naranja -> Incompleta.
- Amarilla -> Esperando cambios en el servicio para su finalización.

7.3 Fases del desarrollo del proyecto

Como se ha comentado anteriormente, el proceso de desarrollo de este proyecto ha sido el más complejo y el más largo. La implementación de la aplicación BlockLotto se ha realizado usando una metodología iterativa (anteriormente comentada), la cual ha dividido el proyecto en iteraciones.

Sin embargo, cada una de las iteraciones divide el proceso de implementación en diferentes fases. A continuación, se explica el proceso de desarrollo de cada una de las fases, así como las tareas incluidas en las mismas.

7.3.1 Implementación de la Interfaz

Para poder implementar la interfaz se tuvo que realizar previamente un diseño de la app (ver *Mock-Ups*). Una vez diseñado, se comenzó con el proceso de implementación.

El proceso de implementación de la aplicación comenzó con la creación del logo. Se diseñó y elaboró una imagen que representara la aplicación, así como todas las tecnologías que utiliza. Además, se decidió incluir el nombre de la app en el mismo. De esta manera, se elaboró el logo utilizando Adobe Illustrator. Esto fue algo complicado, ya que no se disponía de experiencia alguna con esta herramienta.

Posteriormente a la creación del logo, se determinó que quedaría más profesional incluir en la pantalla de inicio una animación, en lugar de una imagen estática. Asimismo, se procedió a crear una animación con el logo elaborado anteriormente. Esto se intentó llevar a cabo utilizando exclusivamente Adobe Photoshop, pero dada la inexperiencia se volvió bastante complicado y se tuvo que utilizar, en su lugar, Adobe After Effects. Con esta última, se compuso la animación. No obstante, hubo un problema: ¿Cómo reproducir una animación cuyo formato es “.mp4” o “.avi” en una aplicación Android? Las soluciones encontradas eran bastante engorrosas, pero se encontró una librería que permite reproducir gifs en Android [29]. De esta manera, se prosiguió a convertir la animación creada en un gif. Para ello, se utilizó Adobe Photoshop.

Seguidamente a la realización del logo y de la animación, se pensó en que los sorteos tienen iconos representativos. Así que, se confeccionaron con Adobe Illustrator.

Una vez se finalizó el proceso de elaboración de logos e iconos distintivos, comenzó el proceso de implementación de la aplicación en su conjunto. Para ello, se ha hecho uso de Android Studio. Asimismo, como Android tiene muchas versiones y variantes, fue necesario decidir en qué versión de Android se desarrollaría la misma. Para elegir qué versión utilizar se realizó una pequeña labor de investigación en la cual se encontró un artículo que expone un estudio de las versiones de Android más utilizadas en 2018 [30]. Esto se puede observar a continuación:

Version	Codename	API	Distribution
2.3.3 - 2.3.7	Gingerbread	10	0.4%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	0.5%
4.1.x	Jelly Bean	16	1.9%
4.2.x		17	2.9%
4.3		18	0.8%
4.4	KitKat	19	12.8%
5.0	Lollipop	21	5.7%
5.1		22	19.4%
6.0	Marshmallow	23	28.6%
7.0	Nougat	24	21.1%
7.1		25	5.2%
8.0	Oreo	26	0.5%
8.1		27	0.2%

Tabla 13 - Utilización de las versiones de Android en 2018

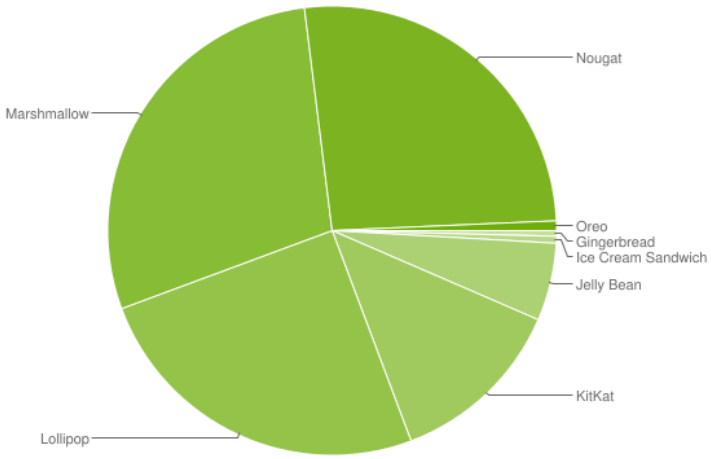


Ilustración 26 - Gráfico comparativo del uso de versiones

Tanto en la tabla como en la ilustración se observa que la versión más utilizada es Marshmallow. Además, la segunda más utilizada es Nougat, la cual es una versión posterior a Marshmallow. Por ello, se decidió que la mejor opción era desarrollar la app con Marshmallow, ya que sería un buen punto de partida, puesto que es la más utilizada y sería compatible con versiones posteriores, siendo la versión directamente superior la segunda más utilizada. Estos datos son los que se utilizaron para decidir en qué versión desarrollar la app (comienzo del proyecto). Actualmente, estos datos han variado un poco, pero Marshmallow sigue siendo la versión más utilizada en 2018.

Una vez decidido todo esto, comenzó el proceso implementación de la interfaz. Android Studio hace uso de layouts (xml) para establecer las interfaces de las aplicaciones y, además, para determinar el correcto funcionamiento de la misma ofrece diferentes lenguajes de programación. En este proyecto se ha utilizado Java.

Así, podemos dividir la implementación de la misma en diferentes iteraciones y tareas:

Pantalla de inicio. Se ha hecho uso de un "Splash Screen" el cual se muestra cada vez que se abre la app. Se reproduce el gif anteriormente comentado durante cuatro segundos. Al mismo tiempo, se comprueba si el usuario ha creado previamente un wallet o no (explicado con más detalle en el punto siguiente). De esta manera, tras reproducir la animación la aplicación redirige al usuario a la pantalla correcta (pantalla de creación o restauración de wallet o pantalla principal de la app)

```
public class SplashScreen extends AppCompatActivity {


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_PORTRAIT);
        this.getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN, WindowManager.LayoutParams.FLAG_FULLSCREEN);

        setContentView(R.layout.activity_splash_screen);

        new Handler().postDelayed(() → { checkDataFile(); }, delayMillis: 3970);


    }

    private void checkDataFile(){
        SharedPreferences preference = getSharedPreferences( name: "credenciales", Context.MODE_PRIVATE);
        String address = preference.getString( s: "address", s1: "-1");
        String pub = preference.getString( s: "pub", s1: "-1");
        String id = preference.getString( s: "id", s1: "-1");
        if((!address.equals("-1")) && (!pub.equals("-1")) && (!id.equals("-1"))){
            Intent intent = new Intent(getApplicationContext(), MainActivity.class);
            startActivity(intent);
            overridePendingTransition(R.anim.zoom_forward_in, R.anim.zoom_forward_out);
        }else{
            Intent intent = new Intent(getApplicationContext(), Start.class);
            startActivity(intent);
            overridePendingTransition(R.anim.zoom_forward_in, R.anim.zoom_forward_out);
        }
    }
}
```

Ilustración 27 - Splash Screen

Creación del wallet. Se ha creado una actividad en la que se informa al usuario que necesita introducir un PIN para poder realizar el procedimiento. Asimismo, se crea el wallet y, posteriormente, se muestran las 24 palabras de seguridad de dicho wallet al usuario. Estas 24 palabras no son almacenadas en ningún sitio (ni servidor, ni dispositivo móvil). Por ello, queda bajo responsabilidad del usuario guardarlas en un lugar seguro. La interfaz informa detalladamente de esto al usuario y le ofrece la posibilidad de copiarlas para enviarlas a un lugar seguro.

R*estauración del wallet.* De igual forma, se ha hecho uso actividades. Sin embargo, en este procedimiento, se solicita al usuario que introduzca tanto el PIN como las 24 palabras de seguridad.

Cuando el wallet es creado o restaurado, la aplicación guarda en el dispositivo un archivo con la clave pública, el id y la dirección del wallet (el guardado de estos datos no distorsiona la seguridad de la app). Este es el método que se encarga de ello:

```
private void createDataFile(String id, String pub, String address){
    SharedPreferences preference = getSharedPreferences( name: "credenciales", Context.MODE_PRIVATE);
    SharedPreferences.Editor editor = preference.edit();
    editor.putString( s: "address", address);
    editor.putString( s: "pub", pub);
    editor.putString( s: "id", id);
    editor.commit();
}
```

Ilustración 28- Credenciales

Así, es utilizado para redirigir al usuario a la pantalla correcta cuando se abre la aplicación. Y, de esta manera, no se hace uso de un login en nuestra app. Además, este archivo es utilizado posteriormente en la realización de peticiones a la API. Asimismo, en caso de desinstalar la app, el archivo también desaparecerá.

Así, una vez creado/restaurado el wallet la app redirige a la pantalla principal y, permite utilizar todas las funcionalidades de la misma. En la etapa de diseño se estableció que la app tendría que poseer un menú con todas las opciones pertinentes. Por ello, se ha hecho uso de un "Navigation drawer". Esto cambia un poco el mecanismo de desarrollo de la app, ya que no se hace uso de actividades, sino de fragmentos.

Se han creado los siguientes fragmentos:

- *Sorteos*. En este fragmento se muestran tres tipos de sorteo (próximos, pendientes y celebrados). Además, se expone el saldo actual del wallet, así como permite actualizar el mismo (por si ha habido algún ingreso). Como en este fragmento se muestran tres tipos de sorteos y, anteriormente se diseñó la manera en la que se establecerían los mismos, se ha hecho uso de un "TabLayout", el cual cumple con lo establecido en la etapa de diseño. Por tal motivo, podemos concluir que este fragmento es como una matrioshka, puesto que contiene tres fragmentos (correspondientes a cada sorteo) en su interior.

  Asimismo, los sorteos se muestran en una especie de lista. Además, para cada sorteo se ha creado una estructura descriptiva correspondiente a cada uno de ellos. Esto ha facilitado el almacenamiento de los mismos en las listas, mediante la utilización de adaptadores que han permitido establecer cada uno de los componentes en la interfaz para, posteriormente, ser visualizados por el usuario. Igualmente, se desarrolló la funcionalidad de poder ordenar los sorteos por: precio ascendente, precio descendente y fecha (más recientes primero). Esta última, no ha sido necesaria implementarla, ya que la API los devuelve así por defecto. A continuación, se muestran los métodos utilizados para implementar el ordenado de los sorteos por su precio:

```
private void sortPriceAsc(ArrayList<PendingLottery> aux){
    Collections.sort(aux, (Comparator) (pendingLottery, t1) → {
            return Double.compare(pendingLottery.getPrice(), t1.getPrice());
    });
    pendingLotteries.addAll(aux);
}

private void sortPriceDesc(ArrayList<PendingLottery> aux){
    Collections.sort(aux, (Comparator) (pendingLottery, t1) → {
            return Double.compare(t1.getPrice(), pendingLottery.getPrice());
    });
    pendingLotteries.addAll(aux);
}
```

Ilustración 29- Ordenar Listas

➢ *Wallet.* Este fragmento muestra el saldo del wallet, los fondos de los canales abiertos y los movimientos. Además, permite recuperar los fondos de los canales y ordenar los movimientos según sean "Pagos" o "Ingresos". Para mostrar los movimientos, de la misma manera que se hizo con los sorteos, se ha hecho uso de adaptadores.

➢ *Operaciones*. En este fragmento, igualmente, se muestra el saldo del wallet. En él se permite ver la clave pública del wallet, realizar peticiones para que otro wallet realice el envío de bitcoins, así como permite realizar transferencias a otros wallets.

- Al *ver la clave pública* se observa un código QR que facilita el envío de bitcoins desde otro wallet.
- La *realización de peticiones* consiste en introducir en un formulario la cantidad de bitcoins que se desea recibir y, el sistema genera un código que el usuario podrá enseñar a otro usuario y, este al escanearlo le realizará el envío de bitcoins que desea.
- *Enviar bitcoins*:
    - Existen varias opciones:
    - Rellenar el formulario introduciendo tanto la clave pública como la cantidad que se desea transferir.
    - Escanear la clave pública y rellenando manualmente la cantidad que se enviará al otro wallet.
    - Escanear una petición de otro usuario enviando la cantidad al wallet receptor la cantidad de bitcoins que dicho usuario a pedido que le transfieras.

A continuación, se muestran los métodos utilizados para realizar el escaneo de códigos QR:

```
private void scanQR(){
    IntentIntegrator integrator = new IntentIntegrator(this.getActivity()).forSupportFragment(this);
    integrator.setDesiredBarcodeFormats(IntentIntegrator.QR_CODE_TYPES);
    integrator.setCameraId(0);
    integrator.setOrientationLocked(true);
    integrator.setPrompt("Escanea el código");
    integrator.setBeepEnabled(false);
    integrator.setBarcodeImageEnabled(false);
    integrator.initiateScan();
}

@Override
public void onActivityResult(int requestCode, int resultCode, Intent data) {
    IntentResult scanningResult = IntentIntegrator.parseActivityResult(requestCode, resultCode, data);
    Bundle qr = new Bundle();//create bundle instance
    if(scanningResult!= null){
        if(scanningResult.getContents() == null){
        }else {
            try {
                qr.putString("QR_SCAN", scanningResult.getContents().toString());
                fragmentSending.setArguments(qr);
                viewPagerOperation.setCurrentItem(2);
                fragmentSending.controlScan();
            }catch(NullPointerException e){
                Toast toast = new Toast(getContext());
                toast.setView(customToast);
                toast.show();
            }
        }
    } else {
        super.onActivityResult(requestCode, resultCode, data);

    }
}
```

Ilustración 30 - Actividad de escaneo

Y, de esta manera, se gestionan los datos obtenidos al realizar el escaneo:

```
public void controlScan(){
    try {
        stringScan = getArguments().getString( key: "QR_SCAN");
        if(stringScan!=null && (!stringScan.equals("")) && (!stringScan.isEmpty())) {
          //  stringScan = getArguments().getString("QR_SCAN");
            for (int i = 0; i < stringScan.length(); i++) {
                if(i != stringScan.length()-1) {
                    if (stringScan.charAt(i) == ',') {
                        addressScan = stringScan.substring(0, i);
                        amount = stringScan.substring(i + 1);
                        break;
                    }
                }
                    addressScan = stringScan;
                    amount = "";
                }
                showAlertScanElement();
        }else{
            addressScan = "";
            amount = "";
        }
    } catch(NullPointerException e){
        addressScan = "";
        amount = "";
    }
}
```

```
private void showAlertScanElement(){
    if((!amount.isEmpty())){
        formatAmount();
        AlertDialog.Builder builder;
        builder = new AlertDialog.Builder(getContext(), R.style.MyDialogTheme);
        builder.setTitle("Enviar BTC")
                .setMessage("¿Estas seguro que quieres enviar " + amount + " BTC a la dirección: " + addressScan + "?")
                .setPositiveButton(android.R.string.yes, (dialog, which) → {
                      sendBTC();
                })
                .setNegativeButton(android.R.string.no, (dialog, which) → {
                        dialog.cancel();
                        addressReception.setError(null);
                        amountSend.setError(null);
                        addressReception.setText(addressScan);
                        amountSend.setText(amount);
                })
                .setIcon(R.drawable.ic_warning)
                .show();
    }else{
        amountSend.setError(null);
        amountSend.getText().clear();
        addressReception.setError(null);
        addressReception.setText(addressScan);
    }
}
```

Ilustración 31 - Control de datos escaneados

Notificaciones. En este fragmento se muestra una lista de notificaciones que el sistema envía al usuario para indicarle que ha sido el ganador de un premio. Esto se ha implementado mediante la elaboración de un adaptador que permita mostrar al usuario toda la información previamente establecida. Además, las notificaciones se clasifican en "leídas y no leídas". Por ello, es la aplicación la que se encarga de establecer el icono distintivo para cada tipo de notificación. Este es el método utilizado para ello:

```
10
11      public void setIcon() {
12          if(readed == 1){
13              icon= R.drawable.ic_notification_open;
14          }else{
15              icon = R.drawable.ic_notification_close;
16          }
17      }
18
```

Ilustración 32 - Icono de Notificación

Asimismo, al igual que se necesitó crear una estructura con los sorteos, ha pasado con los movimientos del wallet y con las notificaciones. Estas estructuras facilitan el manejo de estos objetos, ya que tienen numerosos atributos y métodos que permiten utilizar cada uno de ellos en los diferentes fragmentos y adaptadores.

Finalmente, hay que mencionar que el uso de un "Navigation drawer" ha causado que la actividad principal se dedique exclusivamente a controlar el movimiento por el menú y, la correcta visualización del fragmento correspondiente. A continuación se muestra la actividad principal, llamada MainActivity:

```
public class MainActivity extends AppCompatActivity
        implements NavigationView.OnNavigationItemSelectedListener {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        Toolbar toolbar = findViewById(R.id.toolbar);

        DrawerLayout drawer = findViewById(R.id.drawer_layout);
        setSupportActionBar(toolbar);
        ActionBarDrawerToggle toggle = new ActionBarDrawerToggle(
                activity: this, drawer, toolbar, "Open navigation drawer", "Close navigation drawer");
        drawer.addDrawerListener(toggle);
        toggle.syncState();

        NavigationView navigationView = findViewById(R.id.nav_view);
        navigationView.setNavigationItemSelectedListener(this);

        if(savedInstanceState==null){
            FragmentManager fragmentManager = getSupportFragmentManager();
            fragmentManager.beginTransaction().replace(R.id.container, new Fragment_lottery()).commit();
            getSupportActionBar().setTitle("Sorteos");
        }else{
            String p = savedInstanceState.getString( key: "TITLE");
            getSupportActionBar().setTitle(p);
        }

    }

    @Override
    public void onSaveInstanceState(Bundle savedInstanceState) {
        super.onSaveInstanceState(savedInstanceState);
        savedInstanceState.putString("TITLE", (String) getSupportActionBar().getTitle());
    }
```

```
    @Override
    public void onBackPressed() { moveTaskToBack( nonRoot: true); }


    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.main, menu);
        return false;
    }

    public void itemSelected(int id, FragmentManager fragmentManager, DrawerLayout drawer){
        switch(id){
            case R.id.my_lotteries:
                actionItemSelected( tittle: "Sorteos", new Fragment_lottery(), fragmentManager);
                break;
            case R.id.my_wallet:
                actionItemSelected( tittle: "Wallet", new Fragment_my_wallet(), fragmentManager);
                break;
            case R.id.operation_option:
                actionItemSelected( tittle: "Operaciones", new Fragment_operation(), fragmentManager);
               break;
            case R.id.my_notifications:
                actionItemSelected( tittle: "Notificaciones", new Fragment_notification(), fragmentManager);
                break;
            case R.id.about_aplication:
                actionItemSelected( tittle: "Acerca de", new Fragment_about(), fragmentManager);
                break;
        }
        drawer.closeDrawer(GravityCompat.START);
    }


    public void actionItemSelected(String tittle, Fragment fragment, FragmentManager fragmentManager){
        getSupportActionBar().setTitle(tittle);
        fragmentManager.beginTransaction().replace(R.id.container, fragment).commit();
    }
```

```
@SuppressWarnings("StatementWithEmptyBody")
@Override
public boolean onNavigationItemSelected(MenuItem item) {
    int id = item.getItemId();

    FragmentManager fragmentManager = getSupportFragmentManager();
    Toolbar toolbar = findViewById(R.id.toolbar);
    DrawerLayout drawer = findViewById(R.id.drawer_layout);
    setSupportActionBar(toolbar);
    ActionBarDrawerToggle toggle = new ActionBarDrawerToggle(
            activity: this, drawer, toolbar, "Open navigation drawer", "Close navigation drawer");
    drawer.addDrawerListener(toggle);
    toggle.syncState();

    NavigationView navigationView = findViewById(R.id.nav_view);
    navigationView.setNavigationItemSelectedListener(this);

    itemSelected(id, fragmentManager, drawer);

    return true;
}
```

Ilustración 33- MainActivity

7.3.2 Integración de la API

Una vez finalizada toda la implementación de la interfaz, se prosiguió a integrar la API en la aplicación. Dado que todos los componentes de la interfaz se crearon y organizaron tal y como se acordó en el convenio con LottoService (ver *Diseño del Convenio con LottoService*), esta fase del desarrollo no es muy extensa.

LottoService es un servicio web REST. Por tal motivo, la aplicación y el servicio se comunican mediante el protocolo JSON. Por ello, para integrar la API en la aplicación fue necesario conocer las direcciones URL correspondientes a cada funcionalidad.

Una vez se conocidas todas las direcciones, se precisa conocer cómo responde el servicio ante las peticiones. Para ello, se ha hecho uso de la herramienta SoapUI. Esta herramienta ha permitido conocer durante el proceso de desarrollo si la petición se estaba realizando correctamente, así como, la estructuración de las respuestas de la API a dichas peticiones. Además, se ha utilizado MySQLWorkbench para comprobar que realmente las funcionalidades utilizadas desde la interfaz de la app se estaban ejecutando correctamente, modificando la base de datos del servicio.

En la siguiente ilustración se puede observar un ejemplo de uso de SoapUI. En ella se está creando un wallet:

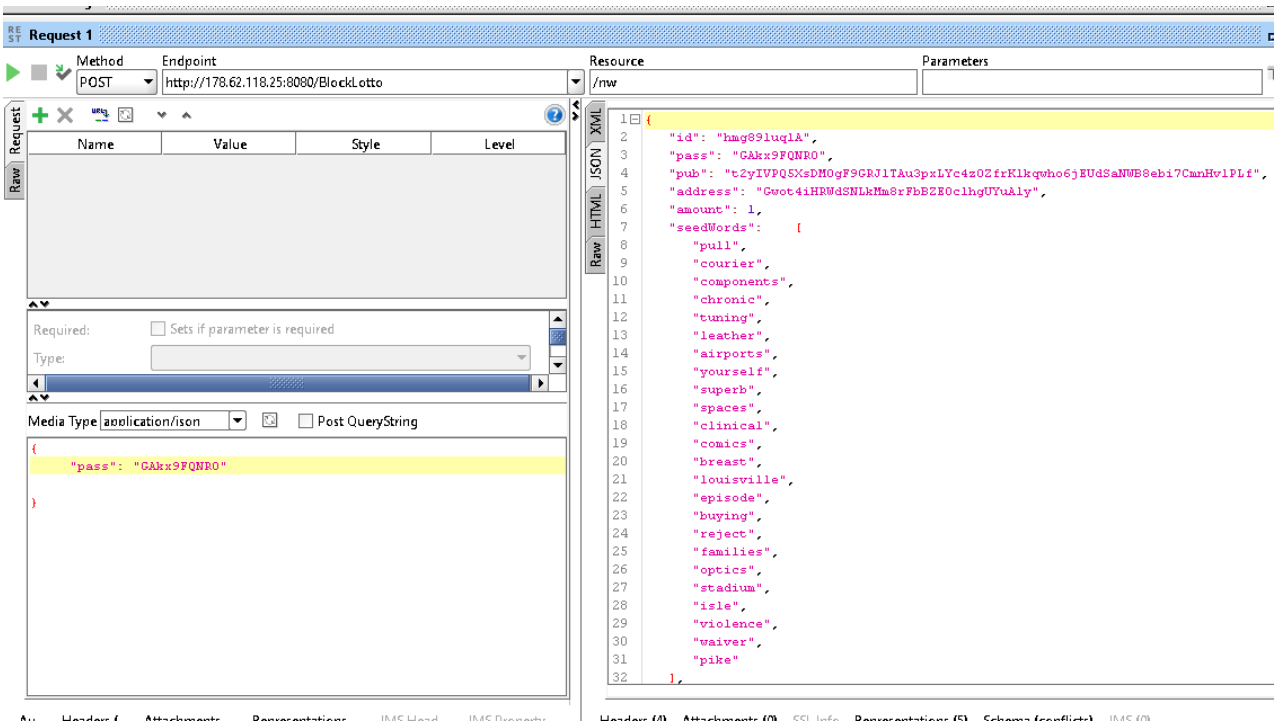


Ilustración 34 - SoapUI, Creación de wallet

Teniendo todo esto claro, se comenzó a desarrollar la comunicación LottoApp-LottoService. Para ello, se creó una clase y mediante un objeto "HttpURLConnection" se realiza la conexión al servicio. Además, se hace uso de objetos como JSONObject para realizar las peticiones al servicio. Y, de la librería Gson para transformar las respuestas del servicio en las estructuras creadas previamente en la app.

De esta manera, se realizaron los métodos pertinentes para realizar las solicitudes al LottoService. Estos métodos corresponden a las siguientes funcionalidades:

- Crear/Restaurar un wallet.
- Mostrar los sorteos.
- Comprar boletos.
- Mostrar el saldo del wallet y los fondos de los canales.
- Recuperar los fondos de los canales.
- Ver los movimientos del wallet.
- Enviar bitcoins.
- Ver las notificaciones enviadas por el sistema al wallet del usuario.
- Establecer la notificación correspondiente a "leída".

Después de realizar algunas pruebas, se visualizó cierto retardo en la navegación de la app. Esto fue debido a que la clase encargada de realizar las peticiones a la API (Accessor), se estaba ejecutando en primer plano. Entonces, la interfaz se pausaba hasta recibir la respuesta del servicio. Se solucionó fácilmente, convirtiendo la clase Accessor

en un “AsyncTask<String, Void, String>”. Sin embargo, funcionalidades como crear o restaurar el wallet no deben ejecutarse en segundo plano, ya que se necesita completar esta funcionalidad para utilizar la app. Todo ello es controlado en la clase Accessor. Por otro lado, se ha utilizado una interfaz AccessorResponse para que las actividades y fragmentos obtengan la respuesta del servicio cuando el Accessor realiza una solicitud.

Por otra parte, una vez finalizada la integración de las funcionalidades del LottoService se comenzó a realizar diferentes pruebas: funcionamiento de la interfaz, correcta visualización de la información, visualización de toda la información deseada, realización de compras adecuadamente, etc.

En estas pruebas, se observaron ciertos errores tanto de la aplicación como del servicio, comunicando los errores a la otra parte de BlockLotto para su corrección o, simplemente para su conocimiento.

No obstante, es necesario destacar un problema que surgió y llevó bastante tiempo resolverlo. Este problema es que, cuando el dispositivo no tiene Internet, la aplicación deja de funcionar inesperadamente. Esto, en un principio, fue algo alarmante, ya que no se conocía el origen del problema y, simplemente se observaba que la aplicación en algunos momentos dejaba de funcionar sin causa aparente. Sin embargo, se resolvió dicho problema asegurando antes de realizar las peticiones que el dispositivo tiene conexión a internet. En caso de no tener acceso a internet, la aplicación informa al usuario sin dejar de funcionar, ni mostrando errores. Los métodos utilizados para ello son los siguientes:

```
private boolean isNetEnabled(Context context) {
    ConnectivityManager connectivityManager = (ConnectivityManager) context.getSystemService(Context.CONNECTIVITY_SERVICE);
    NetworkInfo actNetInfo = connectivityManager.getActiveNetworkInfo();
    return (actNetInfo != null && actNetInfo.isConnected());
}

private Boolean isOnline() {
    try {
        Process p = java.lang.Runtime.getRuntime().exec( command: "ping -c 1 www.google.es");
        int val = p.waitFor();
        boolean reachable = (val == 0);
        return reachable;
    } catch (Exception e) {
        e.printStackTrace();
    }
    return false;
}
```

Ilustración 35 - Control de internet

A continuación, se muestra la clase Accessor, ya que tiene bastante importancia en esta fase, puesto que es la encargada de realizar las comunicaciones entre aplicación y servicio:

```
public class Accessor extends AsyncTask<String, Void, String> {
    public AccessorResponse accessorResponse = null;
    private String json_aux;
    private Context context;

    //ORDER to params --> url (api), request
    @Override
    protected String doInBackground(String... urls) {
        if((!isNetEnabled(context)) || (!isOnline())){
            json_aux = null;
            return null;
        }else {
            return connectAPI(urls[0], urls[1]);
        }

    }

    @Override
    protected void onPostExecute(String result) {
        try {
            accessorResponse.processFinish(result);
            this.finalize();
        } catch (Throwable throwable) {
            throwable.printStackTrace();
        }
    }

    private boolean isNetEnabled(Context context) {
        ConnectivityManager connectivityManager = (ConnectivityManager) context.getSystemService(Context.CONNECTIVITY_SERVICE);
        NetworkInfo actNetInfo = connectivityManager.getActiveNetworkInfo();
        return (actNetInfo != null && actNetInfo.isConnected());
    }
```

```
    private Boolean isOnline() {
        try {
            Process p = java.lang.Runtime.getRuntime().exec( command: "ping -c 1 www.google.es");
            int val = p.waitFor();
            boolean reachable = (val == 0);
            return reachable;
        } catch (Exception e) {
            e.printStackTrace();
        }
        return false;
    }


    public void createWallet(String pass, Context context){
        this.context = context;
        String url = "http://178.62.118.25:8080/BlockLotto/nw";
        String pin ="{" + "\"pass\""+ ":" + "\"" + pass + "\"" + "}";
        this.doInBackground(url, pin);
    }
    public String[] getResultCreateWallet(){
        String [] result = new String [4];

        if(json_aux == null){
            return null;
        }else {
            Gson gson = new Gson();
            Wallet element = gson.fromJson(json_aux, Wallet.class);

            String[] words = element.getSeedWords();
            String twentywords = "";
            for (String word : words) {
                twentywords = twentywords + " " + word;
            }

            result[0] = element.getId();
            result[1] = element.getPub();
            result[2] = element.getAddress();
            result[3] = twentywords.trim();

            return result;
```

```
public void recuperateWallet(String pass, String twentyFourWords, Context context){
    this.context = context;
    String url = "http://178.62.118.25:8080/BlockLotto/rw";
    String pin ="\"pass\""+ ":" + "\"" + pass + "\"";
    String words = "\"seedWords\"" + ":" + twentyFourWords;

    String request = "{" + pin + "," + " " + words + "}";
    this.doInBackground(url, request);
}

public String [] getResultRecuperateWallet(){
    String [] result = new String [3];
    if(json_aux == null){
        return null;
    }else {
        Gson gson = new Gson();
        Wallet element = gson.fromJson(json_aux, Wallet.class);

        if (element.isUnlock()) {
            result[0] = element.getId();
            result[1] = element.getPub();
            result[2] = element.getAddress();
            return result;
        } else {
            return null;
        }
    }
}

public void getBalance(Context context){
    this.context = context;
    String url = "http://178.62.118.25:8080/BlockLotto/wb";
    String request = checkDataFile(context);
    request = "{" + "\"" + "id" + "\"" + ":" + "\""+ request + "\"" + "}";
    this.execute(url, request);
}
```

```
public void getBalanceChannel(Context context){
    this.context = context;
    String url = "http://178.62.118.25:8080/BlockLotto/tbc";
    String request = checkDataFile(context);
    request = "{" + "\"" + "id" + "\"" + ":" + "\""+ request + "\"" + "}";
    this.execute(url, request);
}

public void getBTCFromChannels(Context context){
    this.context = context;
    String url = "http://178.62.118.25:8080/BlockLotto/wf";
    String request = checkDataFile(context);
    request = "{" + "\"" + "id" + "\"" + ":" + "\""+ request + "\"" + "}";
    this.execute(url, request);
}

public void getNextLotteries(Context context){
    this.context = context;
    String url = "http://178.62.118.25:8080/BlockLotto/listud";
    String request = checkDataFile(context);
    request = "{" + "\"" + "id" + "\"" + ":" + "\""+ request + "\"" + "}";
    this.execute(url, request);
}

public void getPendingLotteries(Context context){
    this.context = context;
    String url = "http://178.62.118.25:8080/BlockLotto/listdp";
    String request = checkDataFile(context);
    request = "{" + "\"" + "id" + "\"" + ":" + "\""+ request + "\"" + "}";
    this.execute(url, request);
}

public void getCelebratedLotteries(Context context){
    this.context = context;
    String url = "http://178.62.118.25:8080/BlockLotto/listpast";
    String request = checkDataFile(context);
    request = "{" + "\"" + "id" + "\"" + ":" + "\""+ request  + "\"" + "}";
    this.execute(url, request);
}
```

```
public void getNotifications(Context context){
    this.context = context;
    String url = "http://178.62.118.25:8080/BlockLotto/notifications";
    String request = checkDataFile(context);
    request = "{" + "\"" + "id" + "\"" + ":" + "\""+ request + "\"" + "}";
    this.execute(url, request);
}

public void setNotificationsRead(int id, Context context){
    this.context = context;
    String url = "http://178.62.118.25:8080/BlockLotto/nr";
    String request = "{" + "\"" + "id" + "\"" + ":" + id  + "}";
    this.execute(url, request);
}

public void buyLottery(int idLottery, int amountTicket, Context context){
    this.context = context;
    String url = "http://178.62.118.25:8080/BlockLotto/br";
    String idWallet = checkDataFile(context);
    String request = "{" + "\"" + "id" + "\"" + ":" + "\""+ idWallet + "\"" + ","
            + "\"" + "codRaffle" + "\"" + ":" + idLottery + ","
            + "\"" + "number" + "\"" + ":" + amountTicket + "}";
    this.execute(url, request);
}

public void getTransaction(Context context){
    this.context = context;
    String url = "http://178.62.118.25:8080/BlockLotto/pm";
    String request = checkDataFile(context);
    request = "{" + "\"" + "id" + "\"" + ":" + "\""+ request + "\"" + "}";
    this.execute(url, request);
}

public void sendBTC(Context context, String pub, Double amount){
    this.context = context;
    String url = "http://178.62.118.25:8080/BlockLotto/tc";
    String id = checkDataFile(context);
    String request = "{" + "\"" + "id" + "\"" + ":" + "\"" + id + "\"" + ","
            + "\"" + "toWallet" + "\"" + ":" + "\"" + pub + "\""  + ","
            + "\"" + "amount" + "\"" + ":" + amount + "}";
    this.execute(url, request);
```

```
private String connectAPI(String uri, String json) {
    StrictMode.ThreadPolicy policy = new StrictMode.ThreadPolicy.Builder().permitAll().build();
    StrictMode.setThreadPolicy(policy);

    URL url = null;
    HttpURLConnection connection = null;
    try {
        url = new URL(uri);
        connection = (HttpURLConnection) url.openConnection();
        connection.setRequestMethod("POST");
        connection.setRequestProperty("Content-type", "application/json");
        connection.setRequestProperty("Accept", "application/json");
        connection.setDoOutput(true);
        connection.setDoInput(true);
        connection.setConnectTimeout(300000); //set timeout to 5 min

        //request
        JSONObject cred = new JSONObject(json);
        OutputStreamWriter wr= new OutputStreamWriter(connection.getOutputStream());
        wr.write(cred.toString());
        OutputStream os = connection.getOutputStream();
        os.write(cred.toString().getBytes( charsetName: "UTF-8"));
        connection.connect();

        //response
        BufferedReader in = new BufferedReader(new InputStreamReader(connection.getInputStream()));
        String inputLine;
        StringBuffer response = new StringBuffer();
        json_aux = "";
        while ((inputLine = in.readLine()) != null) {
            response.append(inputLine);
        }
        json_aux = response.toString();
        connection.disconnect();
    } catch (MalformedURLException | JSONException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }
    return json_aux;
```

Ilustración 36 - Accessor

8. Resultados

La finalización de este proyecto ha dado como resultado una aplicación móvil bastante funcional y atractiva para el usuario. Asimismo, se han completado todos los requisitos y funcionalidades previstos en el comienzo del mismo, con las características deseadas. A continuación, se muestran imágenes de la aplicación resultante del proyecto BlockLotto:

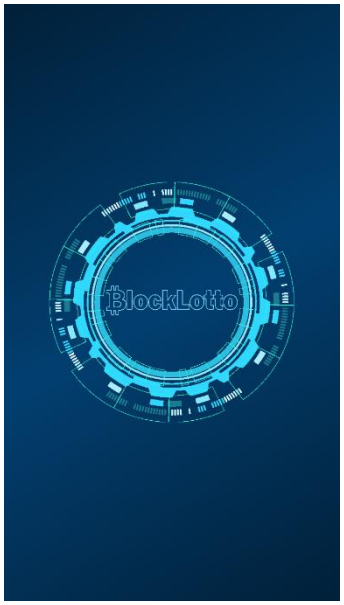


Ilustración 37 - Splash Screen

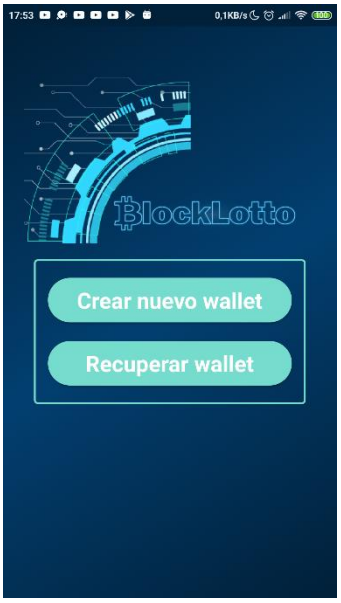
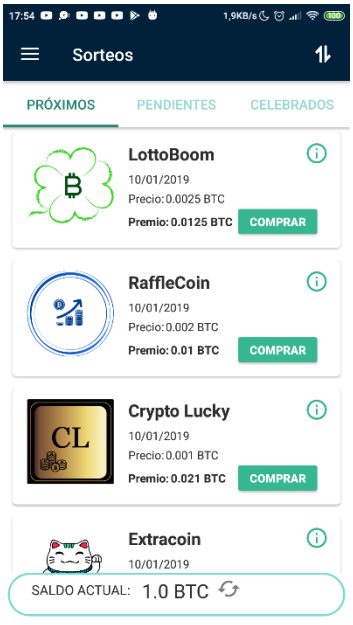
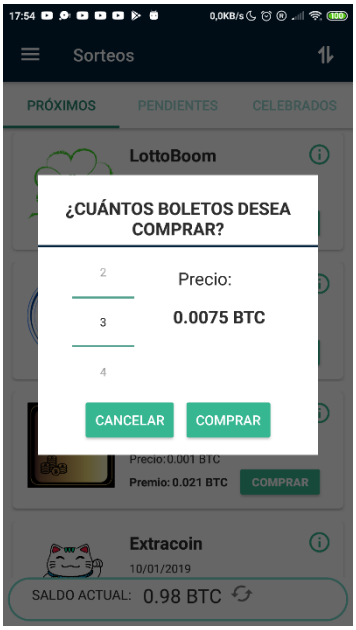
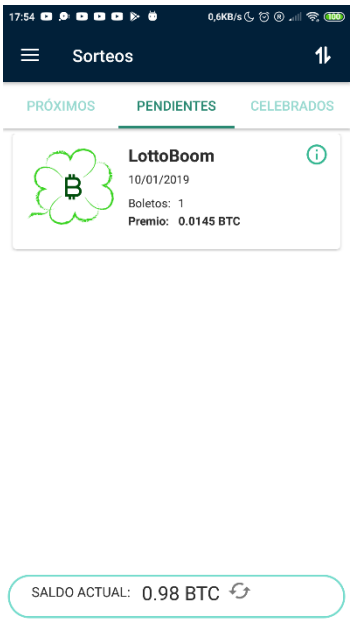


Ilustración 38- Crear/Restaurar Wallet

Ilustración 39 - 24 palabras de seguridad







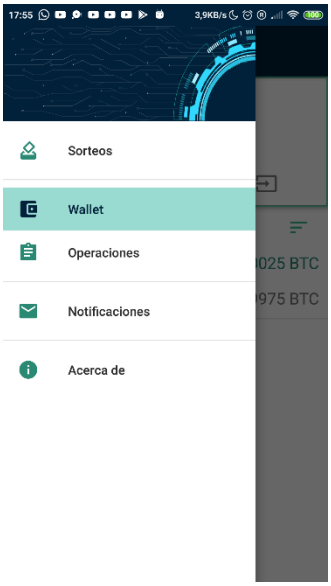


Ilustración 40 - Menú

Ilustración 41 - Wallet

Ilustración 42 - Operaciones

Asimismo, el proyecto completo se encuentra en el siguiente repositorio de GitHub:

https://github.com/CynthiaAG/BlockLotto/tree/master

9. Posibles Extensiones

En este TFT se han desarrollado todos los requisitos establecidos en el comienzo del mismo. Por tal motivo, se han implementado todas las funcionalidades pensadas para la app. Sin embargo, si el producto saliese al mercado, habría que estudiar si los clientes demandan funcionalidades adicionales y, en tal caso, cuales serían.

No obstante, puesto que un objetivo claro de esta app (y de cualquiera) es que la aplicación sea utilizada por el mayor número de personas, una posible extensión sería el cambio de idioma de la interfaz. De esta manera, la aplicación contaría con un público mucho mayor.

Además, para facilitar el uso de la misma sin necesidad de intermediarios, se podría desarrollar una plataforma (propia) encargada de cambiar dinero a bitcoins y viceversa.

10. Conclusiones

Este TFT me ha aportado bastantes conocimientos nuevos en desarrollo de software. Además, he aprendido a desarrollar aplicaciones orientadas a dispositivos Android. Asimismo, a lo largo de todo el Grado se inculca bastante el trabajo en equipo y, una vez más, en este proyecto se ha hecho uso del trabajo colaborativo. Este aspecto, en mi opinión, es muy importante no sólo para el desarrollo software sino, para cualquier entorno profesional.

En mi opinión, la aplicación a quedado bastante bien, ya no solo estéticamente sino funcionalmente. A pesar de los problemas ocurridos, cumple con todas las especificaciones establecidas previamente. Además, posee una interfaz increíblemente intuitiva.

Mi experiencia en este proyecto ha sido gratificante. El desarrollo de aplicaciones me parecía interesante y, tras realizar este proyecto espero realizar muchos más.

Además, el tema criptomonedas es realmente fascinante. Antes de comenzar con el proyecto conocía realmente poco sobre este tema y continuamente oía hablar de él.

Sinceramente, a valido la pena todo el esfuerzo y tiempo invertido en BlockLotto.

11. Referencias

[1] Definición y ventajas de bitcoin: https://www.btc-bitcoin.net/beneficios-ventajas.html

[3] Definición de blockchain: https://es.wikipedia.org/wiki/Cadena_de_bloques

[5]Definición de smart contracts: https://academy.bit2me.com/que-son-los-smart-contracts/#Que_es_un_smart_contract

[6] Definición de Lightning Network: https://en.wikipedia.org/wiki/Lightning_Network

[7] Características de Lightning Network: https://lightning.network

[8] Beneficios de Lightning Network: https://academy.bit2me.com/lightning-network/

[11]Consumer Barometer with Google: https://www.consumerbarometer.com/en/trending/?countryCode=ES&category=TRN-NOFILTER-ALL

[12] Pryze: https://blog.pryze.com/meet-pryze-7b0e51370164

[13] FireLotto: https://firelotto.io/

[14] Android Studio: https://developer.android.com/studio/intro/?hl=es-419

[15] Utilidad de SoapUI: https://es.wikipedia.org/wiki/SoapUI

[16] SoapUI: https://www.soapui.org/downloads/soapui.html

[17] Adobe Photoshop: https://www.adobe.com/es/products/photoshop.html

[18] Adobe Illustrator: https://www.adobe.com/es/products/illustrator/free-trial-download.html

[19] Adobe After Effects https://www.adobe.com/es/products/aftereffects.html

[20] Precios y planes de Adobe: https://www.adobe.com/es/creativecloud/plans.html?single_app=aftereffects&promoid=RPZBNCW8&mv=other

[21] COLOURLovers: https://www.colourlovers.com/

[22] La transmisión de los colores: https://desarrolloweb.com/articulos/color-web.html

[23] Paleta utilizada: https://www.colourlovers.com/palette/1876861/Aurora

[24]Colourblinding: https://chrome.google.com/webstore/detail/colorblinding/dgbgleaofjainknadoffbjkclicbbgaa

[25] Flatcoin: https://www.flaticon.com/

[26] Trello: https://trello.com/

[27] Sourcetree: https://www.sourcetreeapp.com/

[28] Utilidad de GitHub: https://es.wikipedia.org/wiki/GitHub

[29]Librería para reproducir gif: https://github.com/koral--/android-gif-drawable

[2] Artículo de Criptonoticias: https://www.criptonoticias.com/informacion/que-es-tecnologia-contabilidad-distribuida-blockchain/

[4] M.Rodriguez, *15 aplicaciones de la tecnología blockchain más allá de bitcoin*, artículo web. Disponible en: https://www.fin-tech.es/2016/10/aplicaciones-de-la-tecnologia-blockchain.html

[9] Alex Rayón, *Blockchain, Bitcoin y su (supuesta) aportación social,* artículo web. Disponible en: https://alexrayon.es/2018/06/04/blockchain-bitcoin-y-su-supuesta-aportacion-social/

[10]Artículo de Carrefour: https://www.carrefour.es/grupo-carrefour/sala-de-prensa/noticias2015.aspx?tcm=tcm:5-50248

[30] Vania Díaz, *Android Lollipop, Nougat, Oreo ¿Cuál es la Version Más Utilizada?,* artículo web. Disponible en: http://thegeekyshow.com/aplicaciones/android-lollipop-nougat-oreo-cual-es-la-version-mas-utilizada/

12. Anexo

Manual de usuario

Recursos Necesarios para su Instalación

Para instalar y utilizar la aplicación únicamente es necesario disponer de un dispositivo móvil Android, con la versión 6.0 Marshmallow (API 23) o superior. Y, además, acceso a Internet en el dispositivo.

Instalación

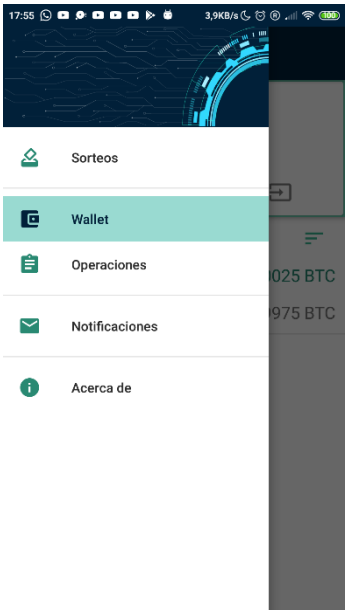
Descargar el APK del repositorio de Github (https://github.com/CynthiaAG/BlockLotto/tree/slave).

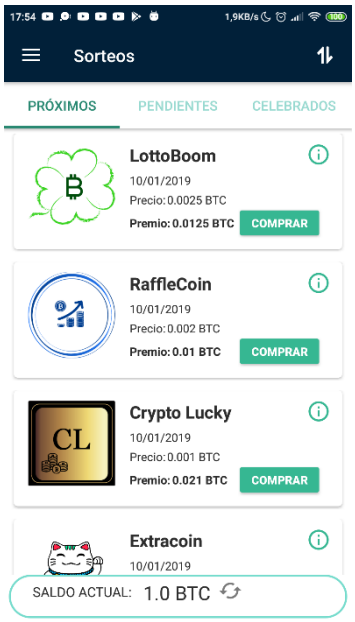
Utilización

Para comenzar a utilizar la app, una vez instalada, es preciso abrirla. Una vez abierta aparecerá una pantalla indicando si se desea crear o restaurar el wallet. Si no se ha usado la app antes, pulse en “Crear Wallet”. Si por el contrario ya se ha creado el wallet anteriormente y se ha desinstalado la aplicación debe pulsar “Recuperar Wallet”.

- **Crear Wallet**. Se pedirá que el usuario introduzca un PIN, esto es necesario para poder crear la cartera correctamente. Posteriormente, se mostrará una serie de palabras. Pulsar en el icono “?” para conocer más. Se debe guardar estas palabras en un lugar seguro, de no ser así, no se podrá recuperar el wallet nunca más y se podrá perder el dinero, así como las inversiones en sorteos.
- **Recuperar Wallet**. Se debe introducir las 24 palabras de seguridad correspondientes a l wallet y un PIN. Si no se posee las 24 palabras, no se podrá restaurar.

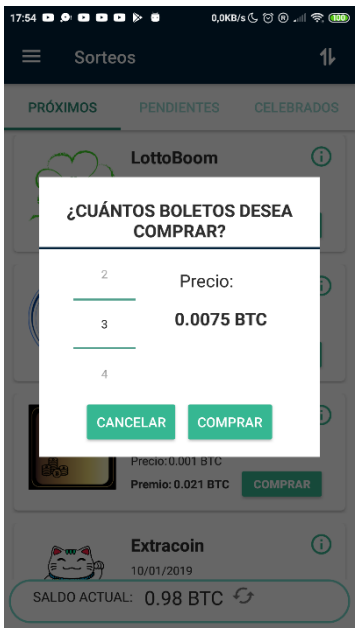
Una vez se posea un wallet se mostrará una pantalla con los sorteos disponibles en el sistema. Y, a continuación, se podrá abrir el menú de la aplicación. En dicho menú están todas las funcionalidades de la app:





Ahora, se expondrá cómo usar cada una de las opciones de la app:

- **Sorteos**. Al seleccionar esta opción vemos los sorteos disponibles en el sistema. Si con el dedo arrastramos hacia la izquierda, veremos los sorteos que tenemos pendientes. Los sorteos pendientes son los sorteos en los que se han comprado boletos y, que aún no se han celebrado. Si volvemos a arrastrar hacia la izquierda veremos los que ya se han celebrado en el sistema. Además, pulsando en el botón que se encuentra en la parte superior derecha, podemos ordenar todos ellos según el precio o la fecha.



  El botón que se encuentra en la parte superior izquierda de cada sorteo, sirve para obtener más información de cada uno de ellos. Así, el botón "COMPRAR" te permite comprar boletos de ese sorteo.

  Para **comprar boletos**, simplemente, hay que pulsar el botón comprar del sorteo deseado, se abrirá una ventana para que el usuario indique cuántos boletos desea y, el sistema le irá calculando la cuenta de su compra (por si desea modificarla). Una vez se ha indicado el número de

boletos que se desea, se pulsa en “COMPRAR” para finalizar el proceso. Finamente, el sistema pedirá una confirmación para llevarla a cabo.

Asimismo, también observamos los bitcoins que poseemos en el wallet y, un botón para actualizarlos (por si se ha recibido alguna transferencia desde otro wallet o, incluso, por haber ganado un premio).

- **Wallet**. Esta opción permite gestionar el dinero. Así, se puede ver el saldo del wallet, los fondos de los canales abiertos y los movimientos realizados en el wallet. Igualmente, se puede ordenar estos movimientos según sean pagos o ingresos, esto es bastante fácil, simplemente pulsa el botón (se encuentra en la esquina superior derecha de los movimientos). El botón se utiliza para recuperar los fondos que se encuentran en los canales y cerrar los mismos.
- **Operaciones**. Esto permite al usuario recibir bitcoins, así como enviarlos a otros wallets. Vamos por orden:

  - Recibir Bitcoins:

Cuando se abre esta opción se puede ver la clave pública del wallet y un código, este código, igualmente, es la clave pública. Sirve para que otros usuarios puedan escanearlo y enviarte bitcoins.

  - Pedir Bitcoins:

Si conoces a alguien que esté utilizando la app, puedes pedirle bitcoins. En esta sección se establece la cantidad que se desea solicitar y el sistema generará un código QR que podrá escanear otro usuario.

  - Enviar Bitcoins:

Se puede hacer de dos formas:

  - Pulsando el botón con aspecto de cámara y, escaneando la clave pública del usuario al que se desea enviar los bitcoins o bien, escaneando la petición que el usuario receptor ha solicitado. Para utilizar el escaneado es necesario permitir a la aplicación tener acceso a la cámara.
  - Rellenando el formulario. Se puede introducir manualmente tanto la clave pública del wallet receptor como la cantidad a enviarle.

- **Notificaciones**. En este apartado se ve una lista de las notificaciones que el sistema ha enviado al usuario. Si es la primera vez que se abre la aplicación, es normal que no se tenga ninguna. Las notificaciones se reciben por haber ganado algún sorteo. En caso de poseer alguna y haberla leído, pulse sobre ella para marcarla como "leída".
- **Acerca de**. Esta opción es plenamente informativa. Permite al usuario conocer la versión de la app, las funciones que ofrece, así como el código fuente y los servicios que utiliza.

Posibles problemas

Si no se instala la aplicación en un dispositivo con los requisitos anteriormente expuestos, es posible que no se instale y, en caso de instalarse, puede no mostrarse todos los componentes de la interfaz.

Si no se puede visualizar el contenido del wallet, así como los sorteos del sistema, es debido a que el dispositivo no tiene acceso a internet. Como se ha comentado anteriormente, es necesario el acceso a internet para la utilización de la app.