UNIVERSIDAD SAN FRANCISCO DE QUITO USFQ

Colegio de Ciencias e Ingenierías

Gamificación de la experiencia universitaria en un ambiente virtual por medio del uso de tokens no fungibles.

Raymond André Maugé Jordán

Ingeniería en Ciencias de la Computación

Trabajo de fin de carrera presentado como requisito
para la obtención del título de
Ingeniería en Ciencias de la Computación

Quito, 18 de enero de 2023

UNIVERSIDAD SAN FRANCISCO DE QUITO USFQ

Colegio de Ciencias e Ingeniería

HOJA DE CALIFICACIÓN
DE TRABAJO DE FIN DE CARRERA

Gamificación de la experiencia universitaria en un ambiente virtual por medio del uso de tokens no fungibles.

Raymond André Maugé Jordán

Nombre del profesor, Título académico **Daniel Riofrío, PhD**

Quito, 18 de enero de 2023

© DERECHOS DE AUTOR

Nombres y apellidos: Raymond André Maugé Jordán

Código: 00116233

Cédula de identidad: 0918281213

Lugar y fecha: Quito, 17 de enero de 2023

ACLARACIÓN PARA PUBLICACIÓN

Nota: El presente trabajo, en su totalidad o cualquiera de sus partes, no debe ser considerado como una publicación, incluso a pesar de estar disponible sin restricciones a través de un repositorio institucional. Esta declaración se alinea con las prácticas y recomendaciones presentadas por el Committee on Publication Ethics COPE descritas por Barbour et al. (2017) Discussion document on best practice for issues around theses publishing, disponible en http://bit.ly/COPETheses.

UNPUBLISHED DOCUMENT

Note: The following capstone project is available through Universidad San Francisco de Quito USFQ institutional repository. Nonetheless, this project – in whole or in part – should not be considered a publication. This statement follows the recommendations presented by the Committee on Publication Ethics COPE described by Barbour et al. (2017) Discussion document on best practice for issues around theses publishing available on http://bit.ly/COPETheses.

RESUMEN

Este proyecto busca el aumento de la motivación de los estudiantes universitarios por medio de la gamificación o ludificación de la experiencia universitaria. Esto por medio de la implementación de un sistema de recompensas escalable utilizando tecnología de blockchain, específicamente el estándar ERC721 de Zeppelin, también llamado NFT. Para complementar a este sistema se desarrolló una aplicación móvil capaz de interactuar con la red, y una aplicación en realidad virtual para visualizar los tokens y usarlos.

Palabras clave: Blockchain, Realidad extendida, Desarrollo Hibrido, NFT, ERC721, IPFS

ABSTRACT

This project strives to increase college students' motivation and engagement through gamification of the college experience. This is tried through the implementation of a scalable reward system using blockchain technology, specifically OpenZeppellin's ERC721 standard, also known as NFT. To complement this system a mobile application able to interact with the blockchain, and a virtual reality experience to visualize the tokens and use them was implemented.

Key words: Blockchain, Virtual Reality, Hybrid Development, NFT, ERC721, IPFS

Tabla de contenidos

Índice de Figuras

Introducción

El siguiente trabajo explora la combinación de varias tecnologías con el fin de lograr la gamificación del entorno académico. La gamificación en sí es un anglicismo proveniente de la palabra game o juego, y hace referencia a la transformación de una o varias actividades no lúdicas, como el entorno laboral o académico, en actividades con elementos de juego, como premios o tablas de puntaje. Siendo el objetivo volver a estas actividades más atractivas al usuario y mejorar su rendimiento en las mismas.

El método de gamificación explorado para este proyecto es el uso de recompensas por méritos académicos. Utilizando tecnologías novedosas tanto para su distribución como para la interacción con las mismas. En el caso de la distribución se explora la funcionalidad de blockchain para almacenar y acceder a información, y realidad virtual como medio de interacción. Para poder interactuar con los premios emitidos de una manera más fácil y poder administrar el sistema se utilizó una aplicación móvil desarrollada por medio de métodos híbridos de desarrollo.

La particularidad del uso de estas tecnologías yace en que la implementación de todas en un mismo proyecto es algo que no se ve a menudo, por lo que para poder llevar a cabo el mismo es necesario tener en mente que no existe amplia documentación sobre el tema. Por lo que se abarcó el proyecto con la idea de probar múltiples soluciones para cada implementación ya que no todas serían compatibles para todas las partes del proyecto.

Motivación

Problemática

La educación de nivel superior puede resultar un reto importante para una parte significativa de la población, alrededor de un 31% de personas que comienzan una educación de tercer nivel no la completan según estudios realizados por la organización por cooperación y desarrollo internacional (OECD)[1] sea por la dificultad de la materia impartida o por su nivel de interés del estudiante. Esta dificultad puede llevar a estudiantes desinteresados que realizan el mínimo, lo cual lleva a profesionales mediocres [2] o a un abandono completo de la carrera en favor de opciones más sencillas. Ambos casos resultan en una pérdida para la sociedad en general.

Para poder incrementar el interés del estudiante promedio se pueden implementar diferentes estrategias de aprendizaje, entre estas la gamificación o ludificación de la materia impartida. Este método incrementa el interés del estudiante promedio, siempre y cuando este tenga ciertas características de personalidad introvertidas [2]. En el caso de los estudiantes más extrovertidos, este no es el caso. Por lo cual una solución a este problema debe tener un componente intrínsicamente social.

Para poder captar a la población estudiantil más social, es necesario incluir un sistema de gamificación que implemente recompensas que se puedan exhibir. Es decir, estas recompensas tienen que poder ser vistas por el resto de la comunidad. Un claro ejemplo de esto son las medallas. Una medalla no solo funciona como una recompensa por un logro, sino también puede ser usado como un símbolo de estatus social. Tomando en cuenta esta problemática se puede plantar la siguiente solución.

Solución

Para motivar al cuerpo estudiantil se plantea la inclusión de elementos de juegos en todas las actividades académicas y deportivas que se realicen dentro de la universidad. Esto por medio de un sistema de recompensas en base a méritos. Este sistema de recompensas debe ser público, tener diferentes tipos de recompensas y cada estudiante debe poder mostrar sus logros tal que estos puedan ser utilizados como un símbolo de estatus dentro de la universidad.

Con el fin de lograr esto se propone el uso de un sistema de distribución de recompensas digital. Esto debido a que es más fácil mostrar un logro si este es siempre accesible, y al ser el celular inteligente uno de los dispositivos más abundantes entre la población universitaria, este resulta idóneo como principal vehículo de exposición de los premios. Para la distribución en sí, se tomó en cuenta la tecnología blockchain, explicada en detalle más adelante, la cual permite mantener un registro de toda emisión de recompensas, esto de una forma segura, escalable e inmutable. Además, el uso de blockchain permite utilizar un derivado de esta tecnología, el contrato inteligente, una especie de contrato que se ejecuta dentro de una red blockchain y comparte la característica de ser altamente confiable. El uso de contratos inteligentes garantiza que cada recompensa que sea otorgada es real y atribuida a la persona correcta. Finalmente, la recompensa en sí tomará la forma de un token no-fungible o NFT, un tipo de token inmutable que puede contener información única.

Las recompensas contendrán información referente a la emisión de esta, pero además tendrán información extra para su uso en otras plataformas. Esto produce que el token sea pesado para ser almacenado en una blockchain, por lo que parte de su información debe ser almacenada externamente. Para mantener el enfoque en la seguridad de los datos se utilizará un sistema de base de datos muy parecido al blockchain, que se llama IPFS, o sistema de archivos

interplanetario. Este sistema de base de datos guarda archivos de forma que no pueden ser alterados, y ofrece robustez en su almacenamiento.

Ya tomado en cuenta la creación de los premios, se debe plantear su distribución y exhibición. El sistema de distribución principal será una aplicación, la cual tendrá también la función de mostrar los tokens y los datos adjuntos a estos. Esta aplicación será móvil y se desarrollará bajo el framework hibrido Flutter, el cual permite un veloz desarrollo para múltiples plataformas. En tanto a la opción de exhibición más interesante para los tokens, se eligió el medio de realidad virtual, el cual permite la inmersión total del usuario, y se espera sea el mayor incentivo para obtener estos tokens.

Objetivos

Generales:

Unir las tecnologías de Blockchain y realidad extendida para gamificar la experiencia universitaria.

Específicos:

- Generar una red de blockchain privada para el uso del proyecto.
- Diseñar un contrato inteligente para la emisión de las recompensas.
- Desarrollar una experiencia en realidad extendida que implemente los tokens emitidos.
- Desarrollar una billetera para uso del proyecto.
- Diseñar un esquema de distribución de recompensas.

Una vez tomado en cuenta estos objetivos es posible proceder a la siguiente parte del proyecto. La investigación de las tecnologías a usar, con el fin de entender a fondo cada pieza que se utilizará en el proyecto.

Investigación

Blockchain

Una red blockchain se puede describir como un sistema de base de datos especializado en transacciones, el cual aspira a ser incorruptible y descentralizado. Esta tecnología surgió en 2008, desarrollada por el mítico Satoshi Nakamoto con su famoso Bitcoin [3], dejando los cimientos para todo el movimiento tecnológico. Tal fue su influencia que la estandarización de la estructura de sistemas blockchain, que está regida bajo la norma ISO 23257:2022 [4], lista como fuente principal a [3]. En su forma más simple, no es más que una cadena de bloques consecutivos, cuyo contenido es verificado dentro de una red a la cual están conectados una cantidad significativa de nodos, los cuales además mantienen un historial de los bloques creados y verificados. La verificación de bloques lleva consigo un incentivo, las criptomonedas, las cuales pueden llegar a ser muy preciadas, tanto por la dificultad de obtenerlas como por su uso en transacciones no regularizadas [5-6]. Existen multitud de opciones para formar una red privada, pero para este proyecto se decidió utilizar el protocolo Ethereum, por su fuerte presencia en el ámbito de desarrollo en blockchain.

Ethereum es una tecnología basada en blockchain, la cual tiene un enfoque en el trabajo sobre la red, es decir, dar la capacidad a los usuarios de usar todo el poder de procesamiento utilizado sobre la red para realizar operaciones [7]. Esto se ha manifestado en cosas como aplicaciones descentralizadas, contratos inteligentes, y una multitud de protocolos tanto de conexión como de seguridad, lo cual mantiene a esta tecnología en constante cambio.

En tanto a la configuración de esta red, es necesario elegir un mecanismo de funcionamiento, o mecanismo de consenso, el cual determina como se realiza la verificación, y a su vez el costo

energético de la red. Los mecanismos más conocidos, y elegidos como candidatos son: PoW (proof of work), PoS (proof of stake) y PoA (proof of authority).

Proof of work, es el mecanismo de consenso clásico, definido en 1999 por Markus Jakobsson como un protocolo de verificación criptográfica [8]. Es el mecanismo de consenso utilizado por la mayoría de las redes blockchain, y el culpable del impacto ambiental que tienen. Según reportes del gobierno americano, a mediados del 2022, el consumo energético de estas redes llegó a estar entre el 0.4% y 0.9% del consumo global [9]. Esto debido a que este mecanismo de consenso pone a cada nodo en la red en una competencia de poder computacional.

En una red con este mecanismo, cada bloque es verificado con operaciones matemáticas que aumentan en complejidad con cada bloque verificado, y solamente el nodo que resuelve más rápido una operación recibe una recompensa. Esto genera la necesidad de utilizar computadoras cada vez más potentes y especializadas.
Los nodos validadores en estas redes son llamados nodos mineros y son totalmente anónimos. La red además es extremadamente segura, siendo solo posible un ataque si un agente malicioso tiene control de más de la mitad del poder computacional de la red. Y en caso de ataques, la red puede excluir al agente malicioso y continuar como un derivado de la red original.

Proof of Stake [10-11-12], a diferencia de PoW se basa en el uso de validadores los cuales ponen como respaldo (stake) una cantidad significativa de criptomonedas. Los validadores no necesitan competir ya que la red les asigna una cantidad de bloques para validar a todos los participantes de la red, en base a su respaldo. Estas validaciones son a su vez validadas por otros nodos en la red, por lo que, si un agente malicioso actúa sobre la red, este recibirá una

penalización, perdiendo los recursos que puso como respaldo, y su confiabilidad dentro de la red.

En este esquema se mantiene el anonimato de los nodos validadores, y la verificación de bloques no se vuelve más compleja a medida que avanza la red, tal que no es necesario componentes particularmente potentes para ser un validador, con lo cual el uso energético de la red es extremadamente inferior al de PoW. La mayoría de las críticas a este mecanismo de consenso viene de que es menos segura que una red PoW, ya que, al necesitar tener un respaldo, una entidad grande puede utilizar sus recursos para invertir agresivamente en su respaldo, obteniendo control sobre una parte significativa de la red y afectando que la validación de ciertos bloques.

Proof of authority [11-12] es un mecanismo de consenso bastante parecido a PoS. Dentro de una red usando este mecanismo cualquier nodo que quiera ser validador debe ser primero aprobado por otros validadores, para lo cual debe verificar su identidad y volverse un agente público. Al ser un agente público, un nodo validador es sujeto al escrutinio de todos los nodos de la red y pone su reputación en juego al validar bloques.

En este mecanismo se asume que un nodo validador es más confiable que en otros mecanismos de consenso, por lo que la verificación de bloque es más rápida y menos costosa energéticamente. Un ataque en esta red significaría que una mayoría de validadores trabajan en conjunto, siendo la única solución generar una bifurcación de la red sin los agentes maliciosos.

Siendo el sistema de recompensas uno con el fin de utilizarse a nivel universitario, se puede elegir el mecanismo de consenso PoA, al no ser necesario mantener el anonimato la universidad puede designarse como verificador principal con nodos secundarios para mantener la red. Esto también ofrece la posibilidad de expandir la red seguramente, ya que, al rechazar el anonimato, se puede designar que solo otras universidades, de por sí entidades públicas, pueden ser agregadas a la red como nodos verificadores.

Para poder comunicarse con esta red es necesario crear un nodo servidor capaz de recibir llamados RPC [13]. Un llamado RPC, o llamado de procedimiento remoto por sus siglas en inglés, es un llamado por el cual un dispositivo ejecuta un procedimiento en un espacio remoto y recibe el resultado del procedimiento como si se hubiera ejecutado localmente.

Contratos inteligentes y NFT

Las recompensas por distribuir deben ser objetos únicos, los cuales contengan información del logro y su dueño. Para esto se puede utilizar un token no fungible, o NFT ERC 721 por sus siglas en inglés, este token, diseñado por Wiliam Entriken [14] e implementado OpenZeppeli [15], funciona de forma parecida a una criptomoneda, ya que es emitido dentro de la red de blockchain por un nodo autorizado, pero a diferencia de la criptomoneda el token lleva un identificador único y además puede almacenar información. Al llevar un identificador este token es único, y al estar almacenado dentro de la red blockchain su valor no puede cambiar, por lo que es inmutable.

Para utilizar este tipo de token es necesario realizar un contrato inteligente, el cual limite quien puede emitir un premio y a quien debe ser emitido.

Este contrato debe ser escrito en un lenguaje llamado Solidity [16], y para desplegarlo en una red privada se debe utilizar una herramienta como Truffle [17], la cual compilan el código del contrato, e interactúan con la red para ejecutarlo. Para este proyecto se decidió utilizar Truffle, el cual ofrece un alto nivel de configurabilidad, lo cual es útil con un mecanismo de consenso poco convencional.

Desarrollo de Aplicación

Para poder interactuar con la red privada es necesario tener una interfaz amigable con el usuario. Una forma de hacer esto es por medio de una “billetera”, con la cual cada estudiante pueda acceder a su colección de premios personal, y un miembro de la facultad académica pueda generar tokens y enviarlos.

Para realizar el desarrollo de esta aplicación existen dos opciones principales, desarrollarla en nativo o con un framework híbrido. Siendo el desarrollo en nativo más largo al ser necesario escribir código para cada plataforma, pero con mayor control sobre el hardware del móvil, mientras el desarrollo en híbrido más rápido, pero sin acceso a ciertos elementos exclusivos de cada plataforma. Para decidir entre estos es necesario reconocer las necesidades de la aplicación [18].

Para este proyecto la aplicación necesita poder acceder a una red privada Ethereum y debe poder ejecutar comandos dentro de la misma. Además, debe poder mantener una base de datos con usuarios y asociar a cada uno sus credenciales. Para poder mantener un diseño simple, sería preferible que exista una distinción entre tipos de usuario, tal que solo un miembro de la facultad académica autorizado sea presentado con la opción de crear un token.

Considerando esto, podemos decir que la aplicación a desarrollarse no necesita utilizar ningún elemento exclusivo para alguna plataforma por lo que no es necesario un desarrollo en código nativo. Tomando esto en cuenta se puede utilizar un framework como Flutter o React Native para realizar el desarrollo. En este caso se utilizará la que tenga mejor documentación, lo cual permitirá mejor resolución de problemas en caso de tener alguno, esta sería Flutter.

Realidad Extendida

La realidad virtual como concepto existe mucho antes que las computadoras, solo hace falta ver un cuadro panorámico, cuadros pintados con el objetivo de sumergir al observador en una escena congelada en el tiempo [19]. En los años 30 se comenzó a imaginar una tecnología que no por medio de gafas nos permita estar dentro de otro mundo, fue recién en los años 60 que esta tecnología toma su primer gran paso con un casco que cubre la visión del usuario sumergiéndolo en el mundo de una película [19]. En la actualidad la realidad virtual goza de experiencias interactivas que la lleva cada vez más cerca de la idea original de inmersión total, pero a su lado han surgido otras premisas, como la de la realidad aumentada y mixta, las cuales buscan integrar nuestro mundo físico a estos mundos virtuales [19]. Estas forman lo que se puede llamar una realidad extendida, en la cual el mundo virtual se junta al mundo real.

Para el desarrollo de una experiencia virtual es necesario la utilización de un motor gráfico. Hoy en día los motores más utilizados para este fin son Unity [20] y Unreal [21]. Siendo Unreal el más popular por su uso en juegos de video de alta gama y en efectos cinematográficos. Además de ser el software más abierto entre los dos, lo preferible para realizar una conexión entre este y la red privada.

Desarrollo

Planificación

Antes de comenzar con el desarrollo del proyecto se planteó un esquema de referencia de como interactuarían los diferentes componentes del proyecto. Este esquema se puede apreciar en la figura 1, esta muestra los componentes con sus respectivas tecnologías y como interactúan con los otros componentes.

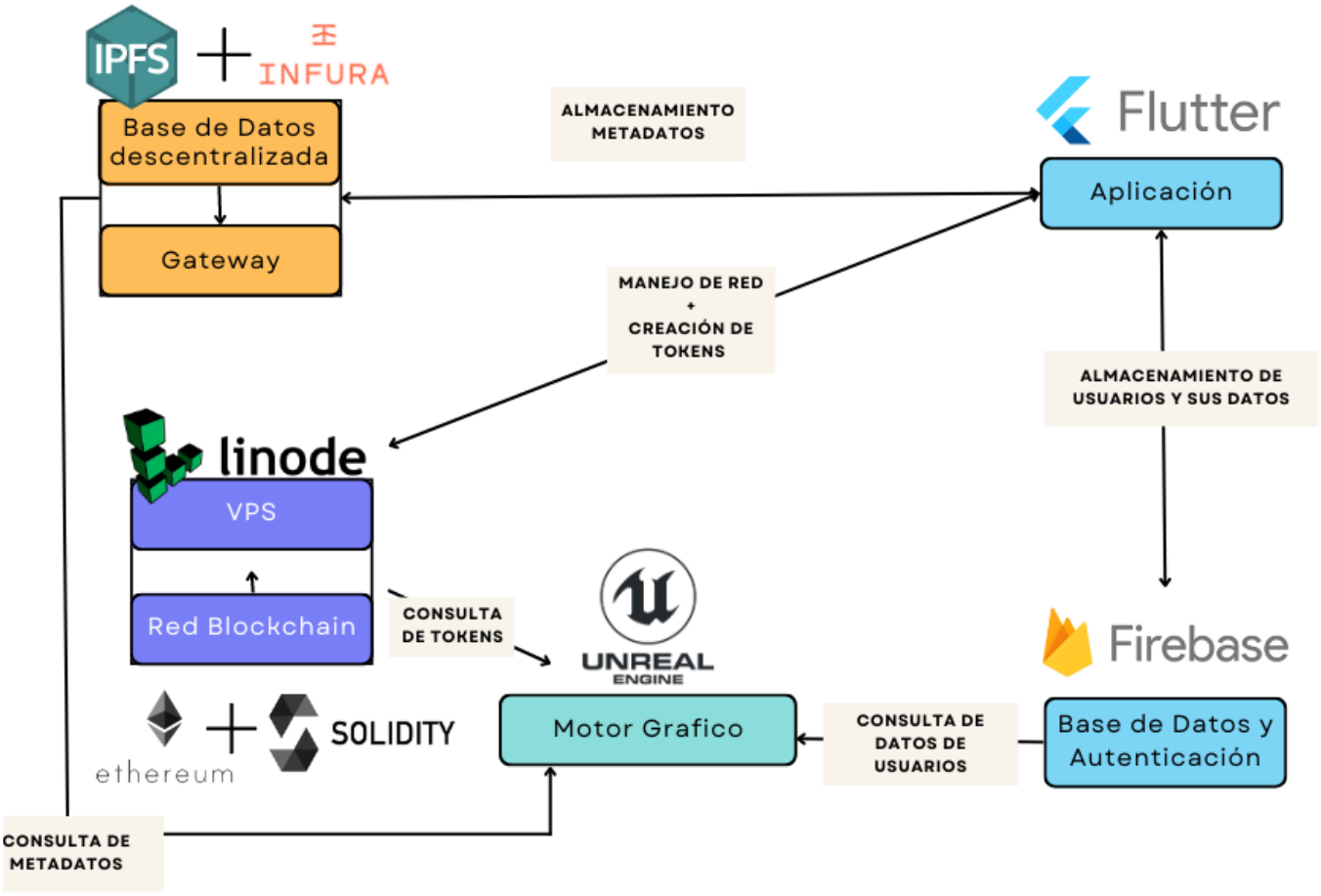

Figura 1: Esquema de conexión de tecnologías

En este esquema podemos ver los cinco componentes principales del proyecto:

- La red blockchain: Construida en base a Ethereum, sobre esta se desplegará un contrato escrito en Solidity. La red estará montada en linea sobre un servidor virtual privado (VPS) del proveedor Linode.
- Motor Gráfico: El motor Unreal es sobre el cual se construirá la experiencia virtual.
- Base de datos y autenticación: Base de datos en el servicio Firebase, la cual tendrá los datos de usuarios y proveerá un servicio de autenticación para los mismos.
- Base de datos descentralizada: Por medio de Infura, se formó un puerto de acceso a la base de datos descentralizada IPFS.
- Aplicación móvil: Construida sobre Flutter usando el lenguaje Dart.

De estos componentes la más compleja es la aplicación móvil, ya que por medio de esta se realizarán operaciones de consulta, creación y modificación tanto en la red blockchain como en las bases de datos. Contrario al motor gráfico, que solamente realizará operaciones de consulta sobre las mismas.

Red Privada

Para levantar la red privada se optó por utilizar la implementación del protocolo Ethereum en Golang, conocida como GETH [22]. Esta implementación ofrece una suite de herramientas que permiten el levantamiento de un servidor que puede funcionar como nodo principal o secundario de cualquier red Ethereum. Para este proyecto se consideró necesario solamente un nodo principal que reciba todos los llamados de dispositivos externos.

Al utilizar GETH la red Ethereum pudo ser configurada a partir de un archivo único inicial con los siguientes parámetros en formato JSON (Anexo A):

- ChainId: el identificador interno de la red debe ser único para cada red. En este caso se utilizó 420042.
- Actualizaciones por utilizar: El protocolo de la Ethereum ha tenido muchos cambios a lo largo de su vida útil, es posible definir si se utilizarán o no. Para esta red se tomaron todas las actualizaciones de seguridad hasta Berlín, la cual se incorporó a finales de 2021.
- Protocolo de consenso: ya designado como PoA, Ethereum lo llama clique, y se configuró con bloques de 2 segundos, agrupados en épocas de 30,000 bloques por lo que cada época de la red tendrá una duración aproximada de 17 horas.
- Dificultad: Determina la complejidad de la validación de cada bloque, debido a que no se realizan operaciones de minado se puede dejar en 1.
- Datos extra: En este campo se designa al validador inicial.
- Alloc: Designa la cantidad de recursos asignados a cuentas específicas en Wei (aproximadamente 10^-18 ETH).

El nodo se levantó en un ordenador corriendo MacOS Ventura con configuraciones mínimas de la red, lo cual inicialmente resultó suficiente para el proyecto. Pero a medida que el proyecto progresaba y se realizaban pruebas con dispositivos en redes diferentes, esta configuración mínima produjo conflictos al querer instanciar nodos secundarios en otros dispositivos. Por lo que se optó por montar el nodo principal en un VPS, o servidor virtual privado, corriendo Linux, específicamente la distribución Ubuntu, con el comando mostrado en la figura 2. Este comando abre el nodo a descubrimiento externo, con un puerto específico para llamados RPC por medio del protocolo HTTP, además de inicializarlo con una cuenta que servirá de administrador principal de la red.

Figura 2: Inicialización de red privada

Este nodo, como se puede apreciar en la figura 3, es una bifurcación de la red principal a partir de la actualización de seguridad Berlin, siendo una red independiente con su propio ChainID (el Chain ID de la red principal es 1), y con su propio consenso.

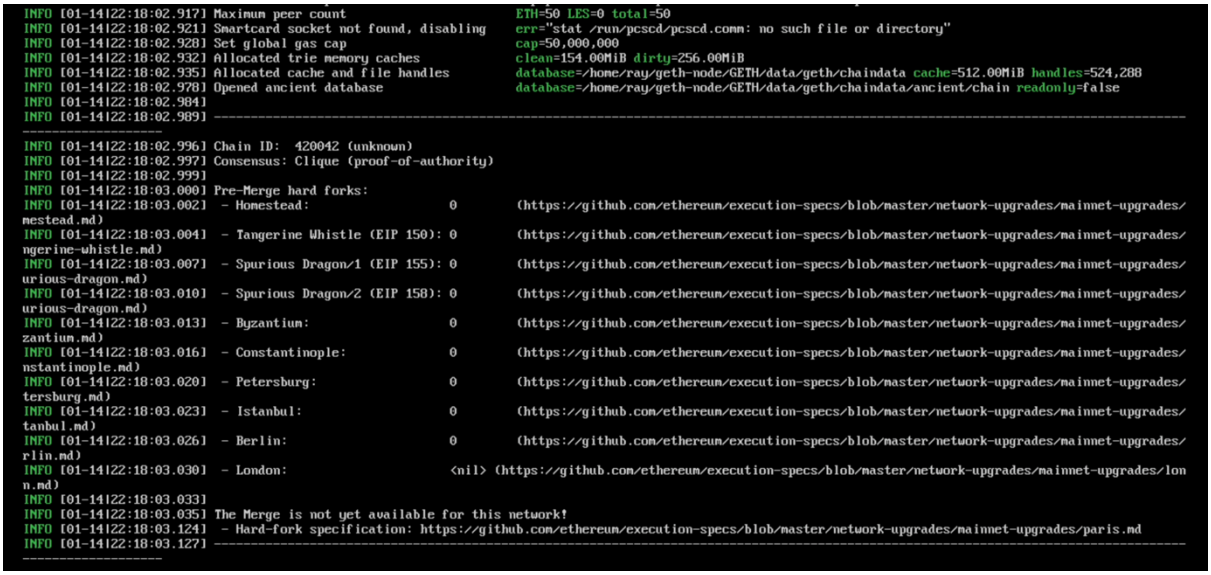

Figura 3: Estado inicial de la red privada

Para el nodo IPFS se comenzó por instanciar una copia de KUBO, la implementación Golang de IPFS [23] en línea de comando, en una VPS igual que el nodo Ethereum. Para comunicarse con este nodo era necesario utilizar llamados RPC por medio del protocolo HTTP al API generado por KUBO, para esto se tuvo que exponer un puerto al público, lo cual resulta problemático sin implementar algún protocolo de seguridad. Esto ya que el nodo puede ser usado por un agente externo para guardar contenido ilegal, o llevar al nodo a capacidad con documentos basura.

Para evitar este tipo de ataques existen dos opciones, implementar un protocolo de seguridad para realizar operaciones de guardado, o utilizar un servicio de hosting especializado en IPFS. Se comenzó con la primera opción, pero esto resulto en que el nodo se desconecte constantemente y sea poco confiable, por lo que se recurrió al servicio de IPFS brindado por Infura [24]. Este servicio monta un nodo IPFS con el esquema de autenticación HTTP Basic, generando una llave e identificador de proyecto los cuales, codificados en base64, dan permiso de guardado de datos.

Contrato Inteligente

Para realizar el contrato inteligente primero es necesario definir un sistema detallado de distribución de tokens. Como base de este sistema está la criptomoneda ether que a su vez esta subdividía en wei, donde 1 ether es igual a 10^18 wei. Esta moneda es asignada a ciertas cuentas administradoras al crear la red. Estas cuentas son asignadas a un cuerpo directivo de la universidad, quienes deciden un presupuesto de wei para el semestre académico. Este cuerpo directivo reparte este presupuesto a los directores cada departamento de la universidad, quienes a su vez lo distribuyen a cada profesor a discreción. Los profesores pueden usar este recurso

libremente para la emisión de premios a los estudiantes. Un gráfico mostrando esta jerarquía se puede ver a continuación en la figura 4.

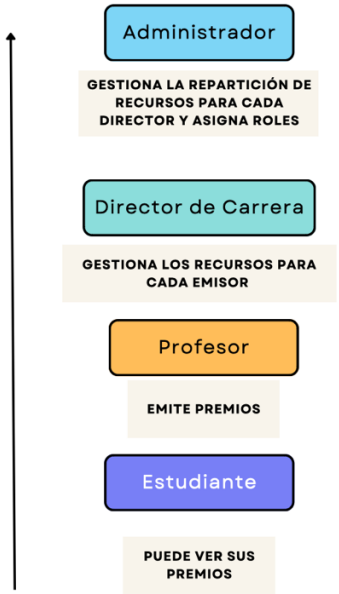

Figura 4: Jerarquía de roles

El contrato Inteligente que soporta esta estructura se realizó bajo el estándar ERC721 de OpenZeppelin, y se puede apreciar en el Anexo A. Bajo este estándar se utilizaron las siguientes interfaces: Access Control, ERC721Enumerable, y ERC721URIStorage. Estas permiten la implementación de métodos altamente optimizados para el manejo de variables dentro de los tokens, además de que están abalados por la comunidad como métodos altamente seguros para uso en redes públicas, y han sido probados extensamente por la misma.

Access Control nos permite implementar roles para los usuarios de la red, Estos roles son:

- Emisores: Usuarios con la capacidad de crear tokens, distribuirlos y transferir criptomonedas para su uso en creación de tokens.
- Administradores: Conceden el rol de emisores a usuarios y poseen grandes cantidades de fondos para manejar el límite de emisión de tokens.

Con el uso de ERC721Enumerable podemos consultar el registro de tokens fácilmente, aunque implica un costo de procesamiento mayor en la creación del token. Este costo inicial es compensado por las constantes consultas que se realizarán a la red para obtener la información de los tokens.

Mientras ERC721URIStorage nos permite almacenar los datos del token de forma flexible al poder pasar las variables únicas del token como argumento y guardarlas dentro de la red, sin esta extensión todos los datos referentes al token deben estar ligados a su ID. Ya que para el almacenamiento de la información extra se planteó el uso de IPFS como base de datos, el acceso a estos este dado por un URL único, y este debe forzosamente estar almacenado dentro de la red.

Una vez escrito el contrato se configura Truffle con el archivo de configuración disponible en el anexo C, en este archivo especificamos los parámetros de la red antes delimitados en la creación de esta. Una vez configurado Truffle nos da la opción de desplegar el contrato, como vemos en la figura 4, y obtenemos una dirección dentro de la red. Esta dirección nos permite ubicar el contrato y direccionar todo llamado RPC al mismo. También se genera un archivo JSON que contiene una versión del contrato en formato ABI. Este archivo detalla el funcionamiento del contrato en forma de instrucciones simples y es necesario incluirlo en cada llamado RPC hacia el contrato.

```
> Compiled successfully using:
   - solc: 0.8.17+commit.8df45f5f.Emscripten.clang


Starting migrations...
======================
> Network name:    'development'
> Network id:      42084218
> Block gas limit: 30000000 (0x1c9c380)


2_token_merito_migration.js
===========================

   Replacing 'TokenMerito'
   -----------------------
   > transaction hash:    0xd0de35c6ed8365ff6dcdf86c0f1ca9ec41de0a292cad20e6407283fb8312d48f
   > Blocks: 0            Seconds: 0
   > contract address:    0x7911DcfaD4b03657cA870328F53C883adaD6dd0B
   > block number:        198572
   > block timestamp:     1670265323
   > account:             0x877fd1934F16294097Eda2e94e5aC2EcDED0D03A
   > balance:             10384593717069655.260363890658439992
   > gas used:            3903723 (0x3b90eb)
   > gas price:           1 gwei
   > value sent:          0 ETH
   > total cost:          0.003903723 ETH

   > Saving artifacts
   -------------------------------------
   > Total cost:         0.003903723 ETH

Summary
=======
> Total deployments:   1
> Final cost:          0.003903723 ETH
```

Figura 5: Despliegue de contrato

Antes de proseguir con la siguiente etapa de desarrollo se utilizó una consola provista por Truffle para probar el contrato y asegurar su correcto funcionamiento. Un ejemplo de esto se observa en la figura 5, donde se puede apreciar la emisión de un token con la dirección de billetera del remitente y un URL con los metadatos del token como parámetros.

Figura 6: Prueba de emisión de token

Diseño de Aplicación

El diseño de la aplicación comenzó con un prototipo simplificado en Figma como se muestra en la figura 7.

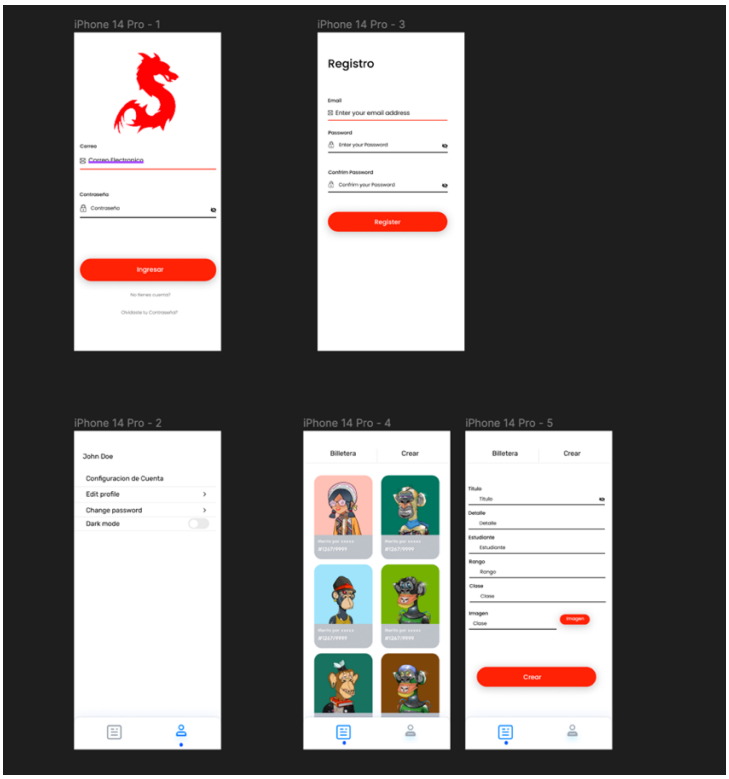

Figura 7: Diseño Aplicación

Este prototipo de diseño se utilizó como inspiración para el desarrollo de la app, además de definir que pantallas se buscaba tener en la app final. En este prototipo se contempló el uso de:

- Una pantalla de inicio de sesión con sus componentes de validación (olvido de contraseña y registro).
- Una pantalla de manejo de perfil.
- Una pantalla tabulada para ver los tokens y crearlos.
- Un navegador en pantalla baja entre perfil y la pantalla tabulada.

Desarrollo de Aplicación

Para el desarrollo de la aplicación primero se contempló que librerías y servicios utilizar. En el lado de servidor se necesitaba manejar una tabla de usuarios, un servicio de autenticación de usuarios, y una base de datos que contenga los datos de los usuarios. De lado gráfico de la aplicación era necesario crear un sistema de navegación y diferentes componentes gráficos.

Del lado servidor se optó por utilizar Firebase Auth para la tabla de usuarios y su autenticación para la base de datos se usó Firebase Firestore, al ser ambos servicios de Google, su integración con Flutter está bien documentada. Además, como se puede ver en las siguientes figuras (figura 8 y figura 9) su uso permitió la visualización y edición de datos por medio de la consola web de Firebase.

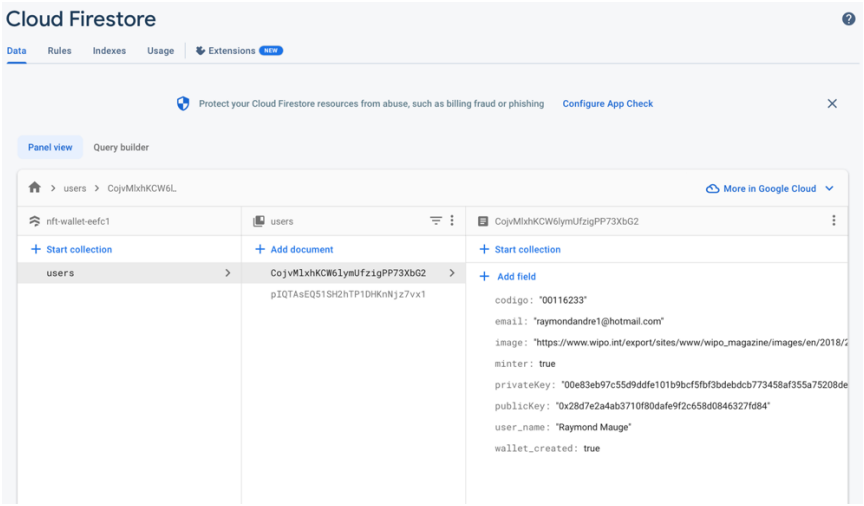

Figura 8: Base de datos en Firestore

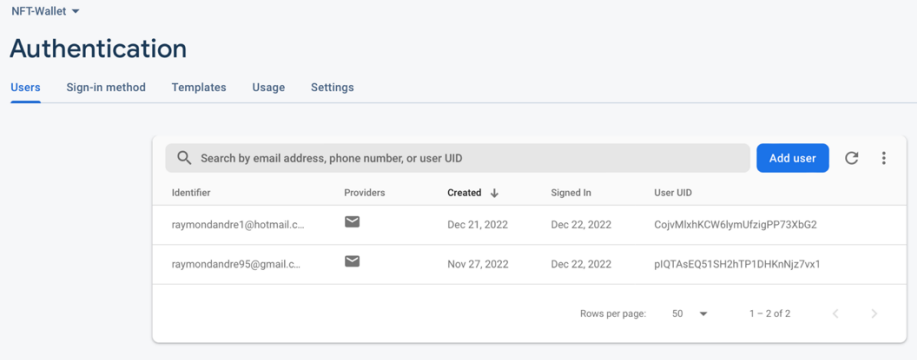

Figura 9: Base de Usuarios en Firebase Auth

Para la parte gráfica del flujo de autenticación del usuario se crearon widgets para interactuar con Firebase, estos se pueden apreciar en la figura 10 y 11. Para la pantalla de perfil se decidió utilizar la librería de Flutter SettingsScreen, esto simplificó el proceso de creación de la pantalla de perfil (Figura 11). Para los demás componentes no se utilizaron librerías fuera de las incluidas en Flutter.

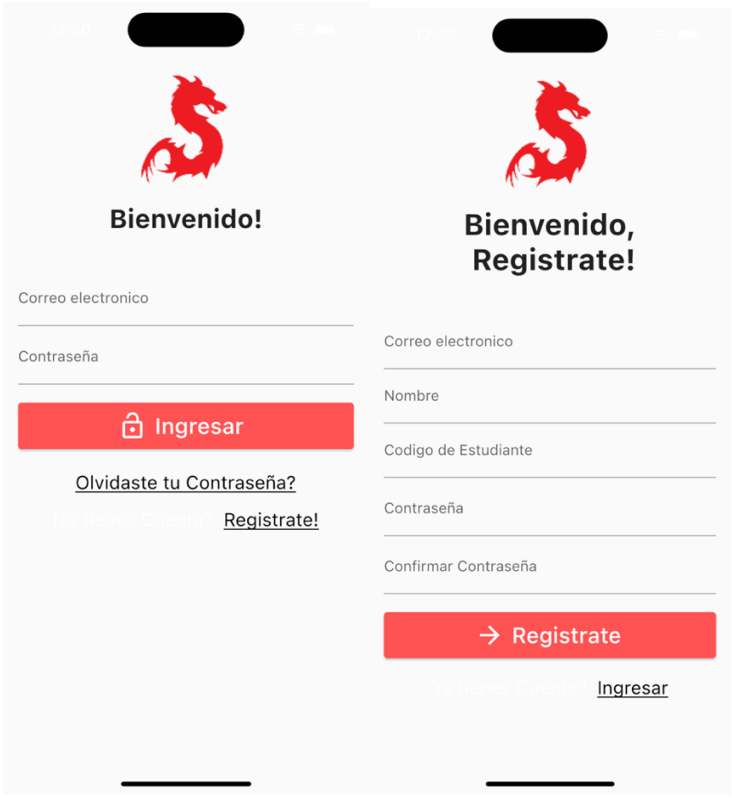

Figura 10: Flujo registro de usuario

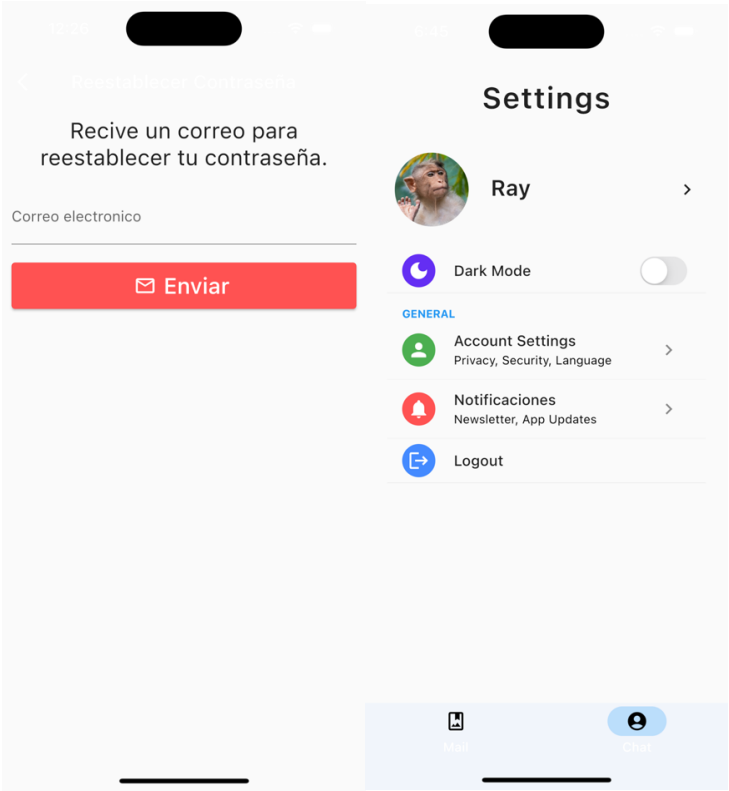

Figura 11: Recuperación de contraseña y ajustes de usuario

Conexión a red privada.

Una vez realizada la parte gráfica de la aplicación y conectarla a la base de datos, se prosiguió a conectarla a la red privada de Ethereum. Para esto se utilizó el paquete WEB3Dart, que simplifica la composición de llamados RPC y conexión con el nodo designado de la red. Para lograr esta conexión también fue necesaria la correcta configuración del rúter conectado a la red Ethereum, ya que estos podrían no llegar desde una red externa.

Para simplificar el proceso de inscripción, se crea una billetera en la propia aplicación, siendo una billetera un par de llaves privada y pública. La llave privada es generada por medio de un algoritmo EdDSA, y la llave pública es derivada de la privada. Estas llaves son guardadas en la base de datos y el usuario puede acceder a ellas por medio de la pantalla de perfil (Figura 12).

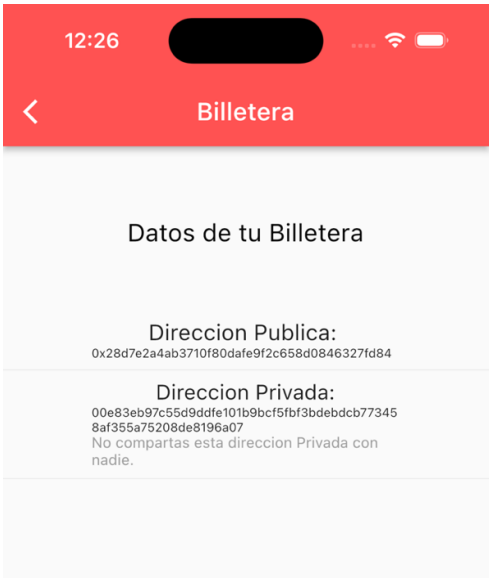

Figura 12: Billetera de usuario

Teniendo una dirección de billetera, esta puede recibir tokens. Para poder ver los tokens que tiene se toma el contrato escrito en Solidity y se extrae su ABI (Interfaz binaria de la aplicación) en formato JSON, el cual nos permite generar llamados RPC al contener el detalle del contrato y su codificación. Estos llamados son generados igual con funciones de la librería WEB3Dart. En el pedazo de código a continuación en la figura 13 se puede ver su uso, tal que el contrato es definido con el archivo ABI en la función privada _getContract, y se le adjunta la función deseada con la función _Query.

```dart
Future<List<dynamic>> _query(String functionName, List<dynamic> args) async {
  DeployedContract contract = await _getContract();
  ContractFunction function = contract.function(functionName);
  List<dynamic> result = await ethereumClient.call(
      contract: contract, function: function, params: args);

  return result;
}

Future<DeployedContract> _getContract() async {
  String abi = await rootBundle.loadString("assets/abi.json");

  DeployedContract contract = DeployedContract(
    ContractAbi.fromJson(abi, contractName),
    EthereumAddress.fromHex(contractAddress),
  ); // DeployedContract

  return contract;
}
```

Figura 13:Consulta al contrato

Esto se usa como en el siguiente ejemplo ilustrado en la figura 14 en el cual la función _getBalance devuelve una lista de todos los tokens asociados a la cuenta.

```dart
Future<void> _getBalance() async {
  loading = true;
  setState(() {});
  List<dynamic> result =
      await _query('tokensOfOwner', [EthereumAddress.fromHex(address)]);
  balance =
      (await ethereumClient.getBalance(EthereumAddress.fromHex(address)))
          .getValueInUnit(EtherUnit.ether)
          .toString();

  _getMetaData(result.toString());
  loading = false;
  setState(() {});
}
```

Figura 14: Consulta a la red ethereum

Una vez obtenidos los tokens objetivo, se extrae su meta data la cual, como se puede ver en el avance anterior, contiene un URL que apunta a un archivo JSON. Los datos contenidos en el

archivo JSON se extraen con facilidad y se los adjunta a un objeto que muestre sus detalles de forma legible al usuario. Con estas mismas funciones es posible realizar el llamado RPC correspondiente a la creación de tokens, una vez probada esta conexión se realizaron las pantallas en la figura 15.

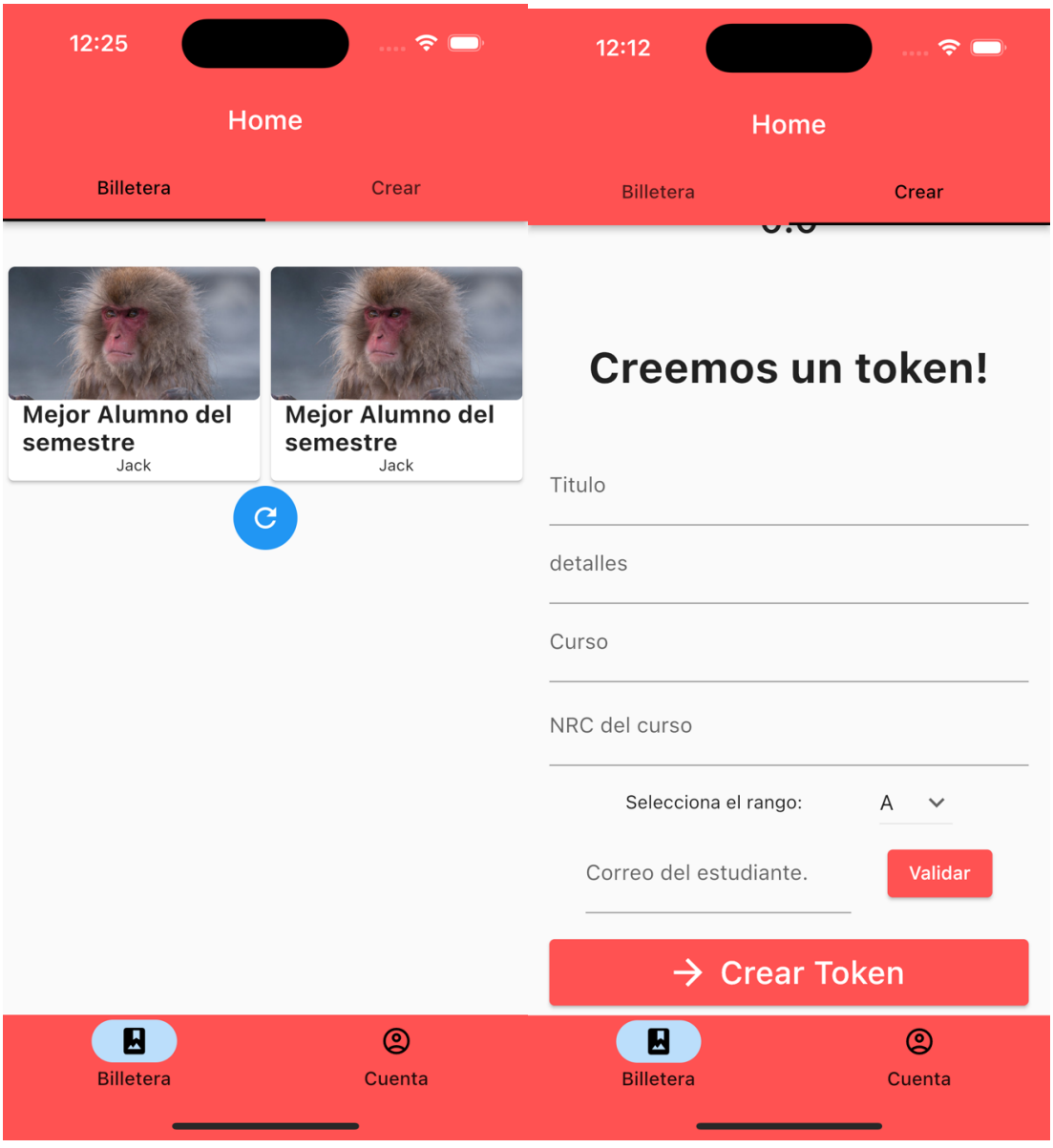

Figura 15: Creación y visualización de tokens

Para facilitar la interacción con la red se realizaron las siguientes pantallas (Figura 16), las cuales permiten, transferir recursos entre usuarios, y asignar un rol a otro usuario. Ambas funciones utilizan el mismo tipo de llamados RPC que la creación de tokens.

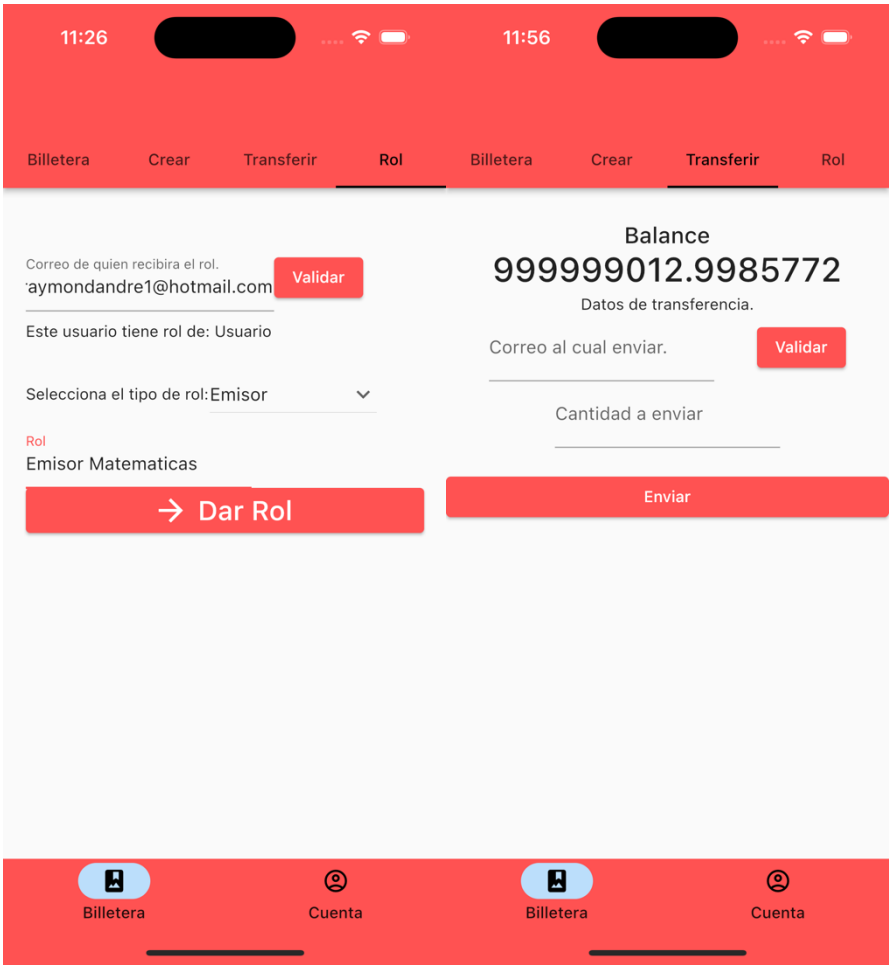

Figura 16 : Asignación de rol y transferencia de fondos dentro de la red

Almacenamiento de Tokens

Una vez solucionado el diseño y conexión de la aplicación se necesita una solución de almacenamiento para las imágenes y archivos JSON. Estos archivos son creados dentro de la aplicación y almacenados locamente antes de ser enviados por medio de una operación RPC con protocolo HTTP a la base de datos IPFS. Par armar esta operación se usa la librería Dio,

con la cual armamos una función postFile (Figura 17) donde especificamos el uso del protocolo de autorización BASIC de HTTP.

```
Future<dynamic> postFile(
    {required String url,
    var queryParameters,
    var formData,
    var authorizationToken}) async {
  try {
    Response response = await dio.post(
      url,
      data: formData,
      queryParameters: queryParameters,
      options: Options(
        headers: {
          "Authorization": "Basic $authorizationToken",
        },
      ),
      onSendProgress: (received, total) {
        if (total != -1) {
          print((received / total * 100).toStringAsFixed(0) + '%');
        }
      },
    );
    return response.data;
  } on DioError catch (e) {
    if (e.response != null) {
      return e.response?.data;
    }
  }
}
```

Figura 17: Operación POST

Desarrollo de experiencia Virtual

En la creación de la experiencia virtual se utilizó la última versión del motor gráfico Unreal, Unreal 5. Esta tiene amplia documentación y facilidades para la creación de experiencias virtuales, por lo que resulta perfecta para trabajar sin mayor experiencia. Unreal funciona en C++, pero permite el uso de una interfaz gráfica llamada Blueprints que está construido sobre C++. Blueprints facilita la creación de funciones para no programadores al dar una imagen clara de lo que hace un programa, pero permitiendo al programador modificarlo detalladamente por medio de código.

El primer paso en el desarrollo sobre el motor gráfico es lograr una conexión completa tanto con la red blockchain como con la base de datos IPFS y Firebase. Estas conexiones se pueden realizar solamente con el uso de blueprints, lo cual nos permite además usar los datos extraídos de forma eficiente al tener acceso a una gran cantidad de librerías abstraídas a este lenguaje visual.

Para poder ejecutar este código de manera global se crea un blueprint que servirá como la instancia de juego de todo el proyecto.

Para comenzar, con el fin de facilitar el acceso y almacenamiento de datos de los tokens que se consultarán, se creó un modelo de datos (Figura 18) para la billetera que contendrá estos datos, este contiene la dirección de la billetera del usuario, el identificador de los tokens que le pertenecen, y una referencia a la información que contienen.

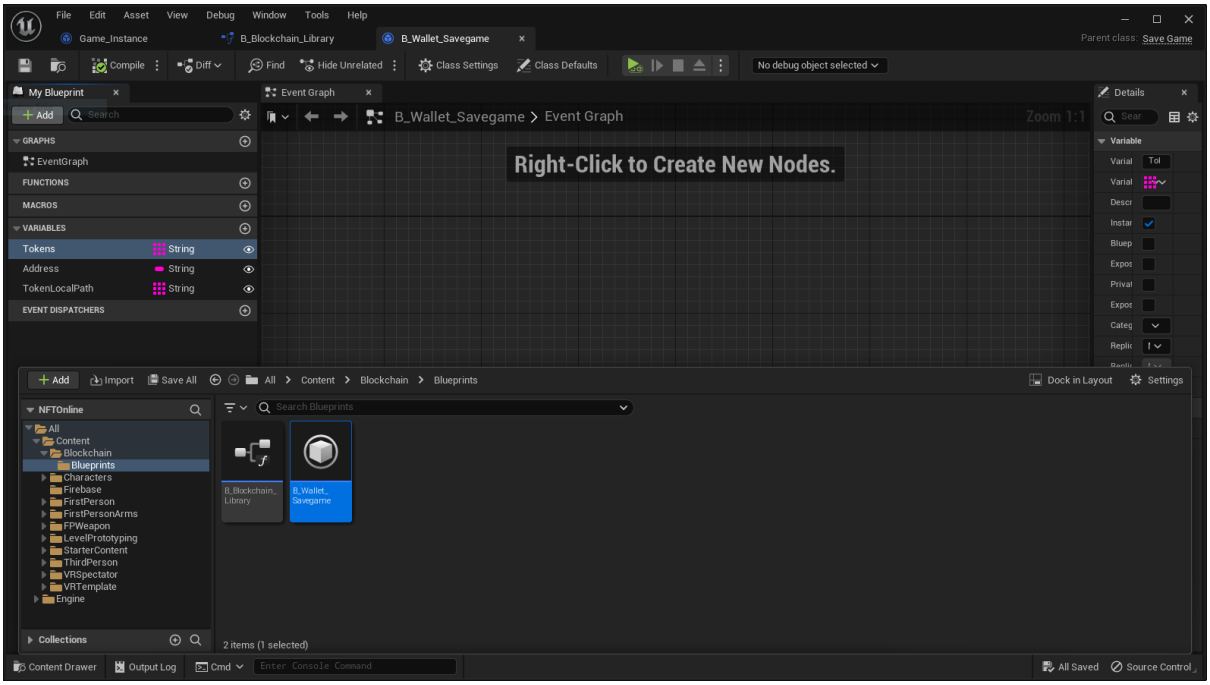

Figura 18: Modelo de datos

Seguido de esto se produce una librería de funciones, como WalletAdd (Figura 19) la cual se encarga de crear un archivo de partida el cual se guarda localmente y está asociada al usuario actual. Esta y otras funciones se encargan del manejo de variables dentro de esta partida guardada.

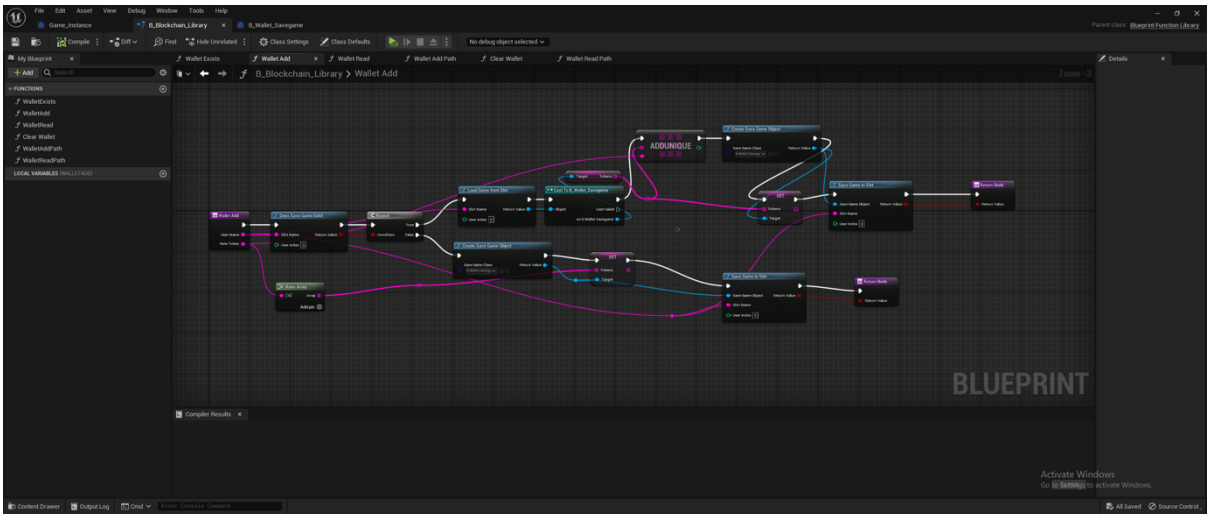

Figura 19: Función para añadir datos a billetera

Finalmente se crea un último blueprint que servirá como instancia del proyecto, es decir se ejecutará cada vez que se inicie el proyecto. Para esto primero se utiliza una librería de conexión a Firebase (Figura 20), provista por la misma entidad, con la cual pasamos como parámetros los datos de un usuario creado por medio de la aplicación, los cuales pueden ser ingresados por el usuario, o como en el caso a continuación, directamente ingresados para agilizar los procesos de prueba. Una vez ingresado al usuario se procede a acceder a la dirección de40illetera del usuario y su nombre de usuario en Firestore por medio de la misma librería.

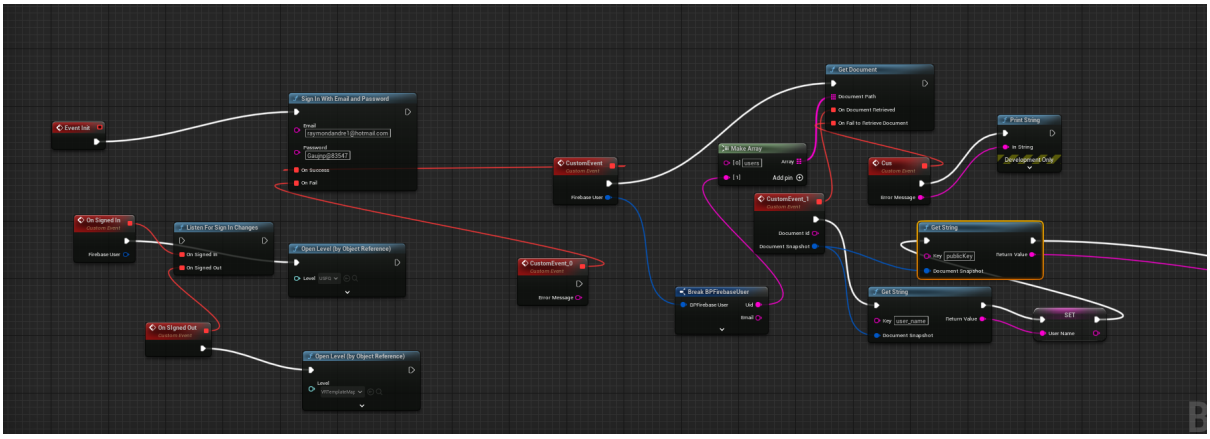

Figura 20: Conexión y autenticación de usuario

Una vez obtenidas estas variables se prosigue a utilizar la librería Unreal Web3 de 3S Game Studios, la cual permite la codificación de los llamados RPC a partir del ABI y dirección del contrato, con lo cual obtenemos los tokens del usuario y sus URL (Figura 21 y 22). Esta librería evita el uso de servidores externos, opción que se contempló en primera instancia, pero resultaba altamente inestable debido a la complejidad en la sincronización con Unreal 5.1.

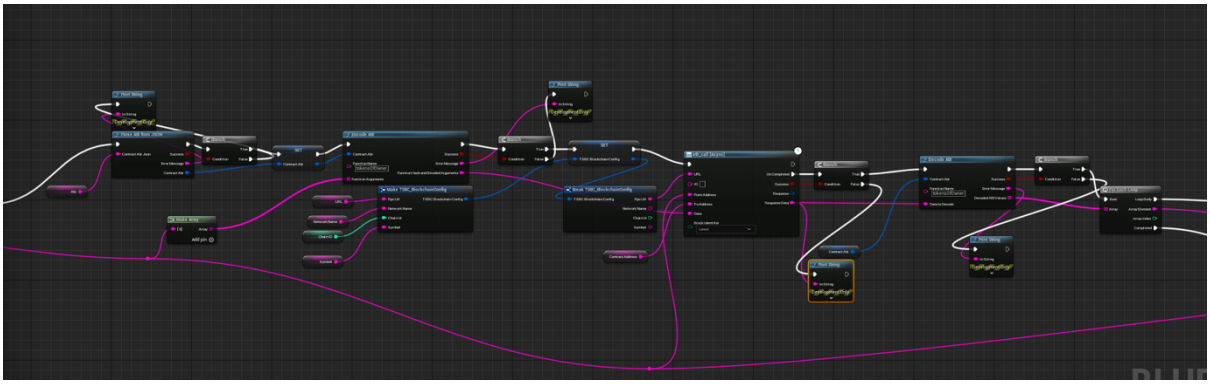

Figura 21: Obtención de tokens del usuario

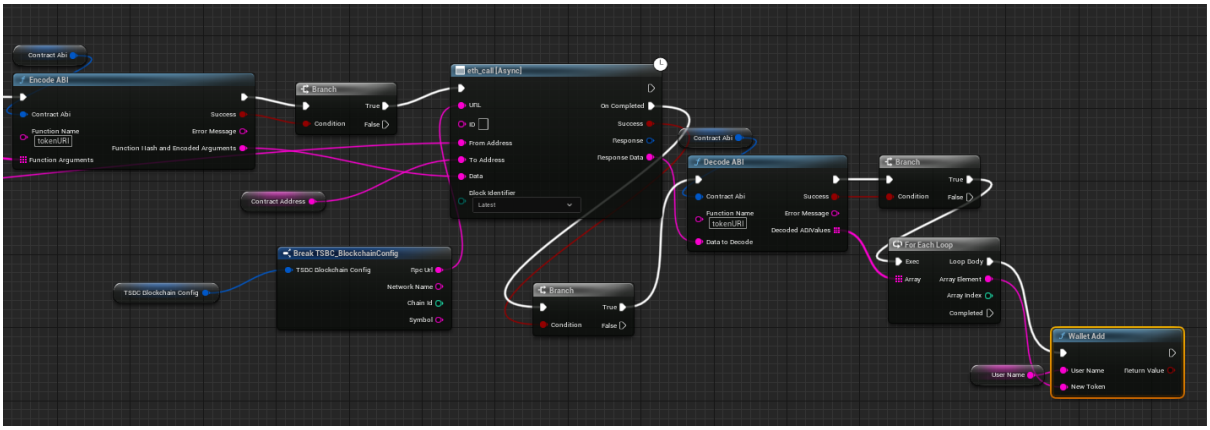

Figura 22: Obtención de URL de tokens del usuario

La misma librería permite hacer las operaciones de petición de los metadatos de los tokens solamente con el URL del Gateway provisto por Infura (Figura 22). Estos datos como están en un archivo JSON, son descargados a memoria, desde donde pueden ser leídos en cualquier momento.

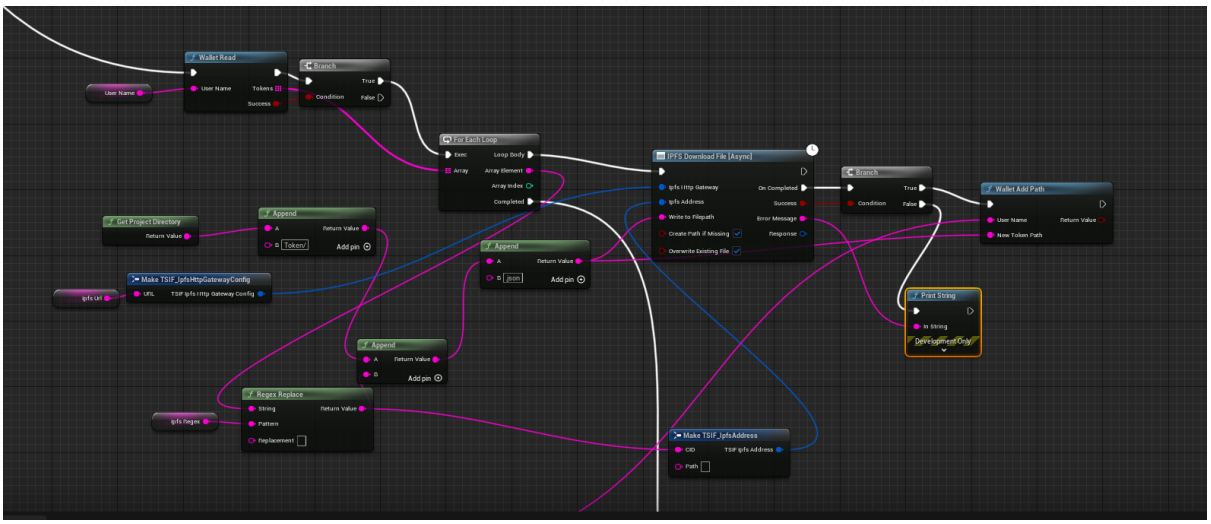

Figura 23: Obtención de metadatos del token

Finalmente se toman los archivos JSON en memoria y son convertidos en objetos, con lo cual podemos extraer los datos que necesitamos por sus llaves, con esto podemos generar objetos relevantes dentro de la experiencia virtual (Figura 24).

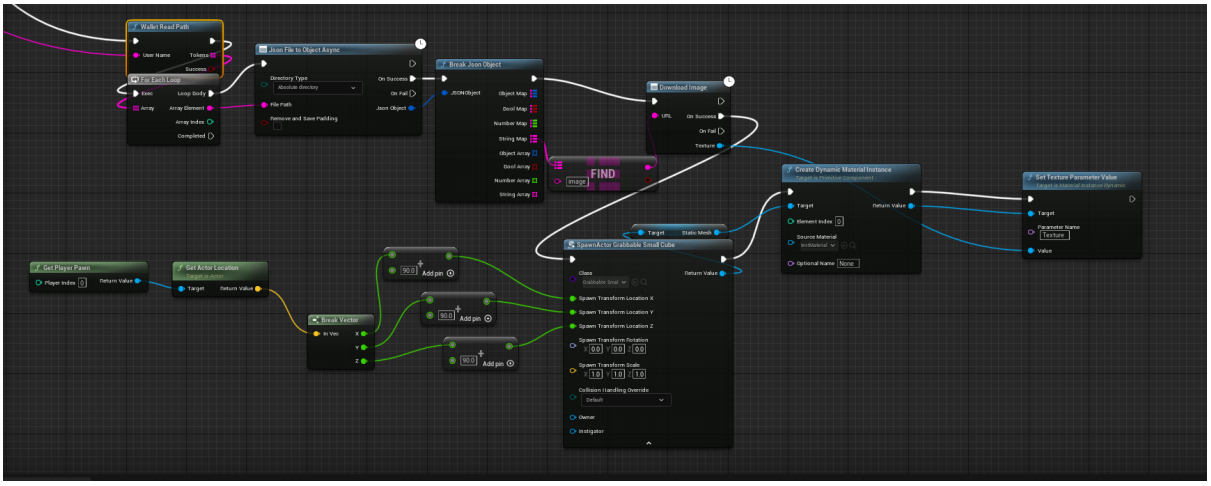

Figura 24: Utilización de metadatos en la experiencia virtual

Una vez terminada la parte de código referente a la interacción con los otros componentes del proyecto se prosiguió a crear un pequeño ambiente virtual con la ayuda de las herramientas del motor gráfico, algo simple para evitar extender el tiempo de desarrollo. Se configuró el proyecto con el estándar de conexión OpenXR que permite compatibilidad con los mayores

fabricantes de cascos de realidad virtual, y se utilizó blueprints dados por Unreal para configurar la interacción del usuario con el entorno gráfico, permitiendo que interactúe con los tokens. Un extracto del producto final se puede ver en la figura 25, donde se puede apreciar el entorno virtual con una pequeña vista de lo que vería el usuario desde su perspectiva, y en la figura 26, donde se crearon cubos y se asignó su color en base al campo de rango asignado en el archivo JSON de cada token.

Figura 25: Captura de zona principal

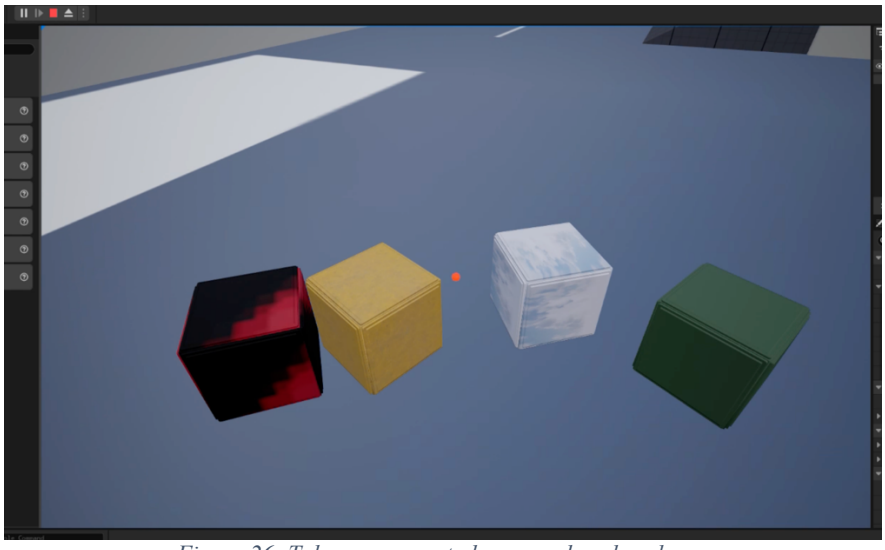

Figura 26: Tokens representados por cubos de colores

Pruebas

A continuación, se puede ver en la figura 27 un ejemplo del proceso de creación de token. En este ejemplo tenemos a Bob, un usuario de la plataforma, y a una cuenta administradora que funciona como el emisor. En la primera pantalla podemos ver los tokens que Bob ya tiene a su nombre, en la segunda el emisor crea un token direccionado a la cuenta de Bob, y en la tercera pantalla aparece el token en la billetera de Bob.

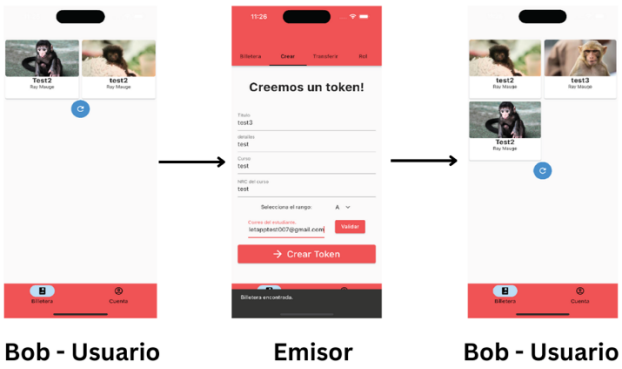

Figura 27: Ejemplo de creación de token

En la figura 28 la cuenta de administrador le asigna a Bob el rol de emisor en la primera pantalla, y en la segunda pantalla podemos ver como Bob gana acceso a la función de creación de tokens.

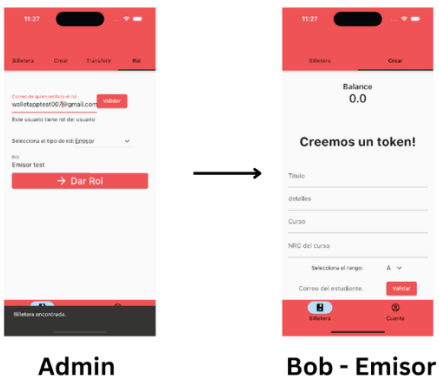

Figura 28: Ejemplo de asignación de rol

En la figura 29 se demuestra la función de transferencia de fondos. En este caso la primera pantalla muestra como la cuenta administradora envía a Bob una suma de 40 ether para crear tokens. Como vimos en la figura 28, la cuenta de Bob no tenía fondos, pero en la segunda pantalla de la figura 29, posee los fondos transferidos.

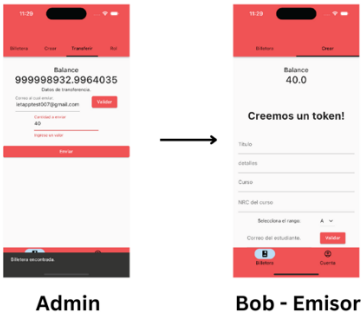

Figura 29: Ejemplo transferencia ether

En tanto a la experiencia virtual, en la figura 30 podemos ver los elementos generados con los datos de los tokens extraídos de la billetera del usuario. Estos son cuatro cubos cuyo color denota su rango.

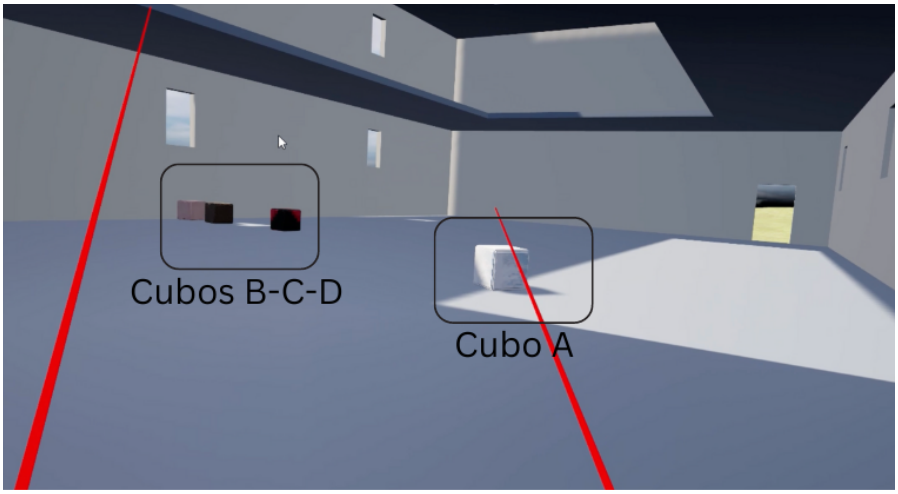

Figura 30: Espacio virtual

Expandiendo sobre la experiencia virtual, en la figura 31 se muestra la interacción con estos objetos, específicamente el cubo A.

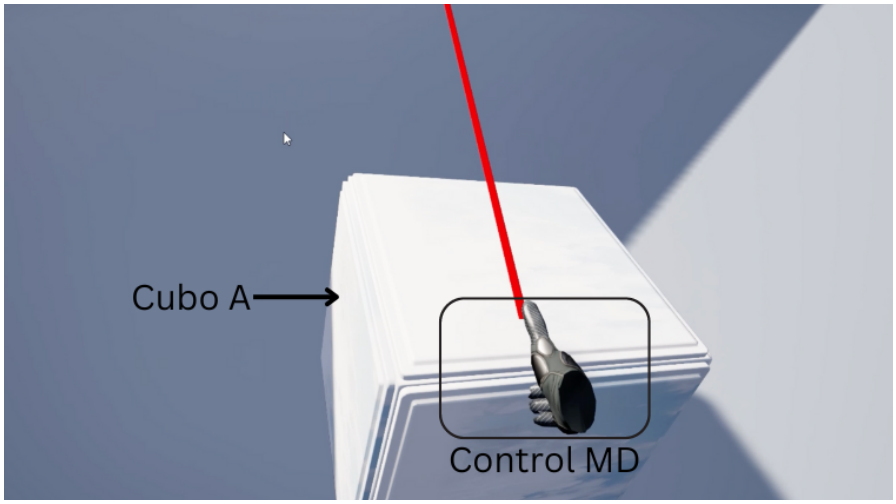

Figura 31: Manipulación de tokens

Conclusiones

Este proyecto comenzó con el objetivo de ofrecer una opción para la gamificación de la experiencia universitaria por medio de la integración de múltiples tecnologías. Dentro de la planificación del proyecto y la investigación de las tecnologías, esta integración tenía el aspecto de ser posible dentro del tiempo estipulado para el proyecto, pero su complejidad resultó mucho más grande de la esperada.

Se puede observar que mucha de esta complejidad se debe al constante desarrollo de estas tecnologías, siendo por ejemplo el caso de que la librería que se utilizó para la conexión entre la red blockchain y el motor gráfico no estuvo disponible al público si no hasta 3 semanas de su implementación en este proyecto. Igualmente se realizaron varias revisiones de la red privada, ya que entre la concepción de la idea y la investigación de esta hubo un cambio radical en el consenso de la red Ethereum.

Estos problemas encontrados a lo largo del desarrollo del proyecto no permitieron que se pudiera desarrollar a la escala deseada. Siendo grandes sumideros de tiempo el aprendizaje de conceptos y lenguajes nuevos, a medida que el proyecto se adaptaba. Con un conocimiento más profundo en mano desde el principio de la ejecución del proyecto se hubiera podido tener un producto mucho más acorde a la visión original.

A pesar de estos percances se podría decir que el producto final logra su cometido de ofrecer una plataforma sobre la cual se podrá desarrollar múltiples proyectos a futuro, los cuales se beneficiarán del trabajo ya realizado.

Expansión del proyecto

Si bien este proyecto logra juntar las tecnologías de blockchain, realidad virtual y aplicaciones móviles, las posibles expansiones para este proyecto son numerosas. Dos posibles direcciones en las que se podría tomar esta base son: transaccionar a un sistema de calificaciones totalmente en redes blockchain, o expandir sobre el contenido en los tokens para poder ser usados en una gran variedad de medios.

La primera, migrar todo un sistema universitario con todos sus sistemas puede resultar en una mayor seguridad. Utilizar sistemas de rol para mantener toda operación estrictamente restringida al personal autorizado, y tener un historial completo de cada acción tomada dentro del sistema, puede dar luz a un sistema extremadamente transparente ante actos de corrupción y ataques externos. Para esto se pueden diseñar contratos que generen tokens de identificación únicos, que solo puedan ser asociados a una persona, contratos que regulen la cantidad de veces que se modifica una nota y mande alertas si hay alguna incongruencia, se puede modificar los consensos de aprobación de operaciones para ser más meticulosos, e incluso se puede proveer un título universitario en un token no transferible, tal que este sea irrevocablemente legítimo.

En otra instancia, se podría expandir la funcionalidad de los tokens para que sean útiles fuera de solo esta aplicación, e incluso de medios virtuales. Por medio de un llavero con NFC, por ejemplo, se podría guardar los datos del usuario para ser consultados en cualquier momento. Esto podría resultar útil para pases de entrada, sea a un concierto como a algún lugar que se necesite identificación.

Referencias bibliográficas

[1] OECD, “How many students drop of tertiary education?”, Highlights from education at a Glance, OECD, Paris, France. 2010

[2] B.S. Frey, y J. Gallus, “Awards as non-monetary incentives”, Evidence-based HRM, vol.4, no.1, pp 81-91, 2016

[3] S. Nakamoto. “Bitcoin: A peer-to-peer electronic cash system”, Bitcoin.org, *https://bitcoin.org/bitcoin.pdf*. (accessed nov 20, 2022)

[4] International Organization for Standardization, “ISO 23257:2022 Blockchain and distributed ledger technologies - Reference architecture”. iso.org. https://www.iso.org/standard/75093.html?browse=tc (accessed jan 17, 2023)

[5] Wang, Guizhou, Zhang, Si, Yu, Tao and Ning, Yu. "A Systematic Overview of Blockchain Research" Journal of Systems Science and Information, vol. 9, no. 3, 2021, pp. 205-238. https://doi.org/10.21078/JSSI-2021-205-34

[6] G. Wood, “Ethereum: A Secure Decentralized Generealised Transaction Ledger Berlin Bersion”, ethereum.github.io, https://ethereum.github.io/yellowpaper/paper.pdf (accessed november 20, 2022)

[7] M. D’Aliessi, “How Does Blockchain Work”, onezero.medium.com. https://onezero.medium.com/how-does-the-blockchain-work-98c8cd01d2ae (accessed september 10, 2022)

[8] Jakobsson, M., Juels, A. “Proofs of Work and Bread Pudding Protocols”, Secure Information Networks. vol 23, pp. 258-272. Springer, Boston, MA. https://doi.org/10.1007/978-0-387-35568-9_18

[9] The White House, “Climate and Energy Implications of Crypto-Assets in the United States”, Press Release, 2022

[10] A. Baliga, “Understanding blockchain consensus models”. Persisten.com. https://www.persistent.com/wp-content/uploads/2017/04/WP-Understanding-Blockchain-Consensus-Models.pdf (accessed nov. 20, 2022)

[11] B. Lashkari, & P. Musilek, “A Comprehensive Review of Blockchain Consensus Mechanisms”. IEEE Access. PP. 1-1. 10.1109/ACCESS.2021.3065880, 2021

[12] “Proof of stake (POS)” Ethereum.org. https://ethereum.org/en/developers/docs/consensus-mechanisms/pos/ (accessed sept. 25, 2022)

[13] B. Matturo. “Remote Procedure Call”. Techtarget.com. https://www.techtarget.com/searchapparchitecture/definition/Remote-Procedure-Call-RPC (accessed oct. 10, 2022)

[14] W. Entriken. “EIP-721: Non-Fungible Token Standard” eips.ethereum.org. https://eips.ethereum.org/EIPS/eip-721(accessed sept. 25, 2022)

[15] “ERC 721” doc.openzeppelin.com. https://docs.openzeppelin.com/contracts/2.x/api/token/erc721(accessed sept. 25, 2022)

[16] “Solidity” docs.soliditylang.org. https://docs.soliditylang.org/en/v0.8.17/ (accessed sept. 25, 2022)

[17] “What is Truffle”. Trufflesuite.com. https://trufflesuite.com/docs/truffle/ (accessed oct. 19, 2022)

[18] “The Good and the Bad of Flutter App Development” altexsoft.com. https://www.altexsoft.com/blog/engineering/pros-and-cons-of-flutter-app-development/ (accessed oct. 2, 2022)

[19] M. Vasarainen, S. Paavola, and L. Vetoshkina, “A Systematic Literature Review on Extended Reality: Virtual, Augmented and Mixed Reality in Working Life”, IJVR, vol. 21, no. 2, pp. 1–28, oct. 2021.

[20] A. Sinicki. “What is Unity? Everything you need to know”. Androidauthority.com. https://www.androidauthority.com/what-is-unity-1131558/ (accessed sept. 18, 2022)

[21] “Unreal Engine 5” unrealengine.com. https://www.unrealengine.com/en-US/unreal-engine-5 (accessed sept. 18, 2022)

[22] Autores de go-ethereum. “Go Ethereum”, github.com. *https://github.com/ethereum/go-ethereum*.

[23] “IPFS Documentation” docs.ipfs.tech. https://docs.ipfs.tech (accessed jan. 3, 2022)

[24] “IPFS” docs.infura.io. https://docs.infura.io/infura/networks/ipfs (accessed jan. 5, 2022)

Anexo A: Genesis.json


```
{
  "config": {
    "chainId": 420042,
    "homesteadBlock": 0,
    "eip150Block": 0,
    "eip150Hash": "0x0000000000000000000000000000000000000000000000000000000000000000",
    "eip155Block": 0,
    "eip158Block": 0,
    "byzantiumBlock": 0,
    "constantinopleBlock": 0,
    "petersburgBlock": 0,
    "istanbulBlock": 0,
    "berlinBlock": 0,
    "clique": {
      "period": 2,
      "epoch": 30000
    }
  },
  "nonce": "0x0",
  "timestamp": "0x638466e1",
  "extraData":
      "0x0000000000000000000000000000000000000000000000000000000000000000877fd1934f16294097eda2e94
      e5ac2ecded0d03a0000000000000000000000000000000000000000000000000000000000000000000000000000
      0000000000000000000000000000000000000000000000000000000000000000",
  "gasLimit": "0x47b760",
  "difficulty": "0x1",
  "mixHash": "0x0000000000000000000000000000000000000000000000000000000000000000",
  "coinbase": "0x0000000000000000000000000000000000000000",
  "alloc": {
    "10232203467b10d40319d191573878ad9fde6b74": {
      "balance": "0x200000000000000000000000"
    },
    "684780c7e262855d28af7ccc101cc16e4ac06adb": {
      "balance": "0x200000000000000000000000000"
    },
    "7821d0f096abfcb6dbe9bf8dc65e3722685e992e": {
      "balance": "0x200000000000000000000000000"
    },
    "877fd1934f16294097eda2e94e5ac2ecded0d03a": {
      "balance": "0x200000000000000000000000000"
    },
    "8798fe282eec40207d38cf3a7a2caf9f134c027a": {
      "balance": "0x200000000000000000000000000"
    }
  },
  "number": "0x0",
  "gasUsed": "0x0",
  "parentHash": "0x0000000000000000000000000000000000000000000000000000000000000000",
  "baseFeePerGas": null
}
```

Anexo B: Contrato Inteligente

```
//SPDX-License-Identifier: MIT
pragma solidity ^0.8.17;

import "@openzeppelin/contracts/token/ERC721/extensions/ERC721URIStorage.sol";
import "@openzeppelin/contracts/token/ERC721/extensions/ERC721Enumerable.sol";
import "@openzeppelin/contracts/access/AccessControl.sol";
import "@openzeppelin/contracts/utils/Counters.sol";
import "@openzeppelin/contracts/utils/math/SafeMath.sol";


contract TokenMerito is ERC721URIStorage, ERC721Enumerable, AccessControl {
    bytes32 public constant MINTER_ROLE = keccak256("MINTER_ROLE");
    using SafeMath for uint256;
    using Counters for Counters.Counter;

    Counters.Counter private _tokenIds;

    string public baseTokenURI;

    constructor() ERC721("TokenMerito", "TOME") {
        _setupRole(DEFAULT_ADMIN_ROLE, msg.sender);
    }

    function awardAchievement(address stud, string memory uri)
        public
        returns (uint256)
    {
        require(hasRole(MINTER_ROLE, msg.sender), "Caller is not a minter");

        uint256 newItemId = _tokenIds.current();
        _mint(stud, newItemId);
        _setTokenURI(newItemId, uri);

        _tokenIds.increment();
        return newItemId;
    }

    function tokensOfOwner(address _owner) external view returns (uint[] memory) {
        uint tokenCount = balanceOf(_owner);
        uint[] memory tokensId = new uint256[](tokenCount);

        for (uint i = 0; i < tokenCount; i++) {
            tokensId[i] = tokenOfOwnerByIndex(_owner, i);
        }
        return tokensId;
    }

   function _beforeTokenTransfer(address from, address to, uint256 tokenId, uint256 batchSize)
        internal
```

```
    override(ERC721, ERC721Enumerable)
  {
    super._beforeTokenTransfer(from, to, tokenId, batchSize);
  }

  function _burn(uint256 tokenId) internal override(ERC721, ERC721URIStorage) {
    super._burn(tokenId);
  }

  function tokenURI(uint256 tokenId)
    public
    view
    override(ERC721, ERC721URIStorage)
    returns (string memory)
  {
    return super.tokenURI(tokenId);
  }

  function supportsInterface(bytes4 interfaceId)
    public
    view
    override(ERC721, ERC721Enumerable, AccessControl)
    returns (bool)
  {
    return super.supportsInterface(interfaceId);
  }

}
```

Anexo C: Repositorios

Repositorio proyecto VR:
https://github.com/WaywardCentaur/ProyectoIntegradorUnreal

Repositorio aplicación móvil:
https://github.com/WaywardCentaur/ProyectoIntegradorApp

Repositorio aplicación móvil:
https://github.com/WaywardCentaur/ProyectoIntegradorBlockchain