1. Caso de estudio: Fintech Pay2Meta

4.1 Arquitectura general

La arquitectura utilizada fue una de microservicios usando Google Cloud, en la figura 13 se ilustra el diseño utilizado, el cual consta de varias aplicaciones clientes entre móvil y web, además todos los datos provenientes de estas aplicaciones clientes serán encriptadas con RSA hacia los api Gateway el mismo que se encargará de balancear y distribuir las peticiones https hacia los diferentes microservicios, estos microservicios se encargarán de desencriptar la información y volverla a encriptar con AES usando las llaves privadas obtenidas de la base de dato criptográfica de IOTA Stronghold para posteriormente almacenar la información en diferentes base de datos relacionales y no relacionales. Finalmente, estos microservicios realizarán envíos de notificaciones utilizando el balanceador de carga y cloud pub/sub de Google; y también conexiones con plataformas externas como Stripe, Paymentez; almacenamientos en IOTA y elaboración de smart contracts con Iotex y Tatum.

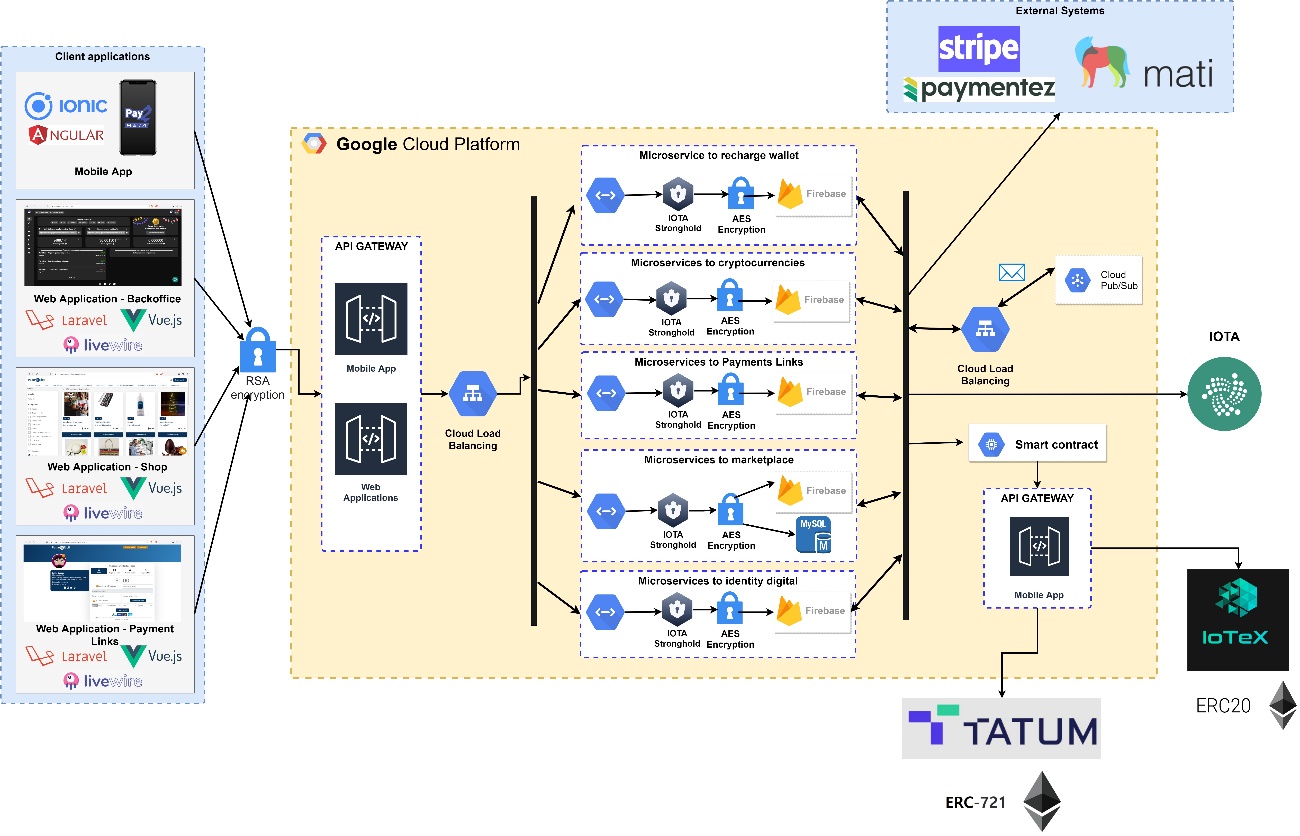


Figure 13: Pay2Meta architectural design

4.2 Microservicios

En la figura 14 se ilustra la utilización de dos instancias de tipo Google Cloud Engine para el almacenamiento de las aplicaciones web correspondientes al backoffice y al Marketplace que internamente también se configuraron manejo de eventos y notificaciones para las diferentes aplicaciones Fintech Pay2Meta.

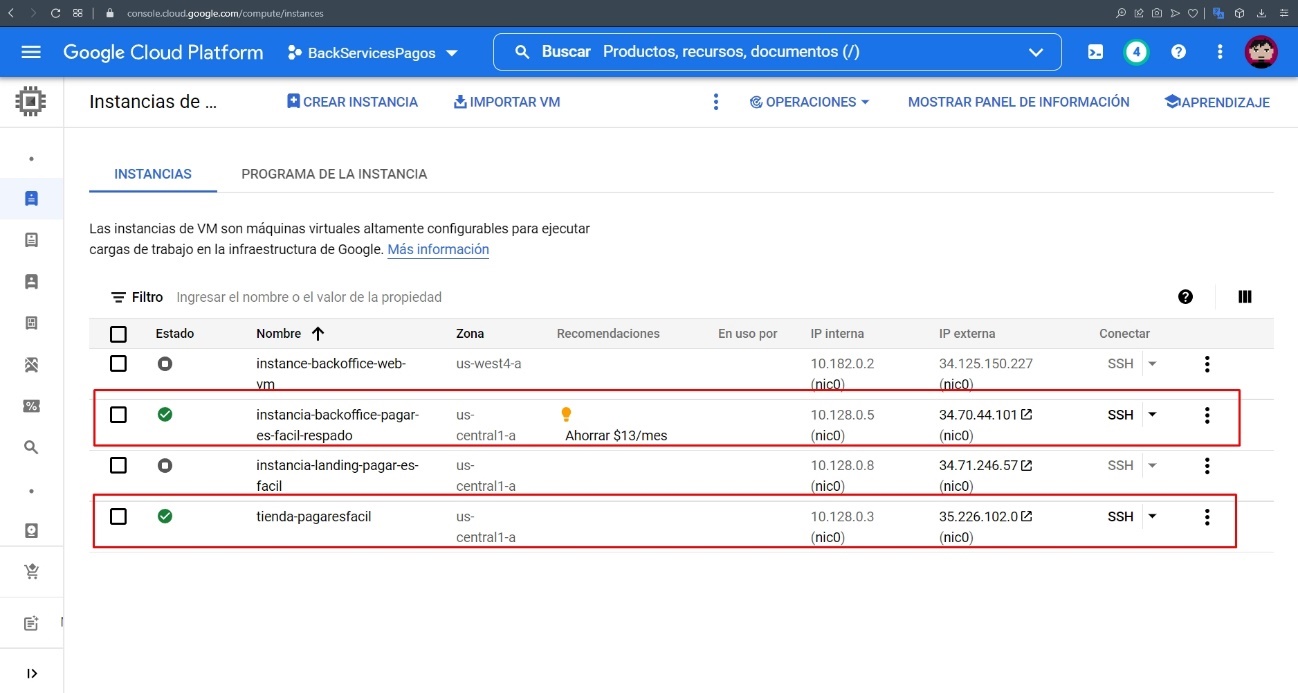


Figure 14: Instance Google Cloud

En cuestión de los microservicios, en la figura 15 se ilustra la implementación de endpoints utilizando Firebase functions de tipo https, entre ellas se destaca el endpoint “consultas2” utilizado para funcionalidades generales de las aplicaciones como pagos, transacciones, consultas etc y el endpoint “cryptos” utilizado para las funcionalidades correspondientes a los DLT como son los contratos inteligentes, NFT y pagos con criptomonedas. En la figura 15 también se destacan las extensiones de tipo Firebase Bigquery para la realización de consultas por lotes (batch). Cabe recalcar que estos microservicios fueron implementados en NodeJS 16, NPM 6.14.10 y Firebase 10.8.0.

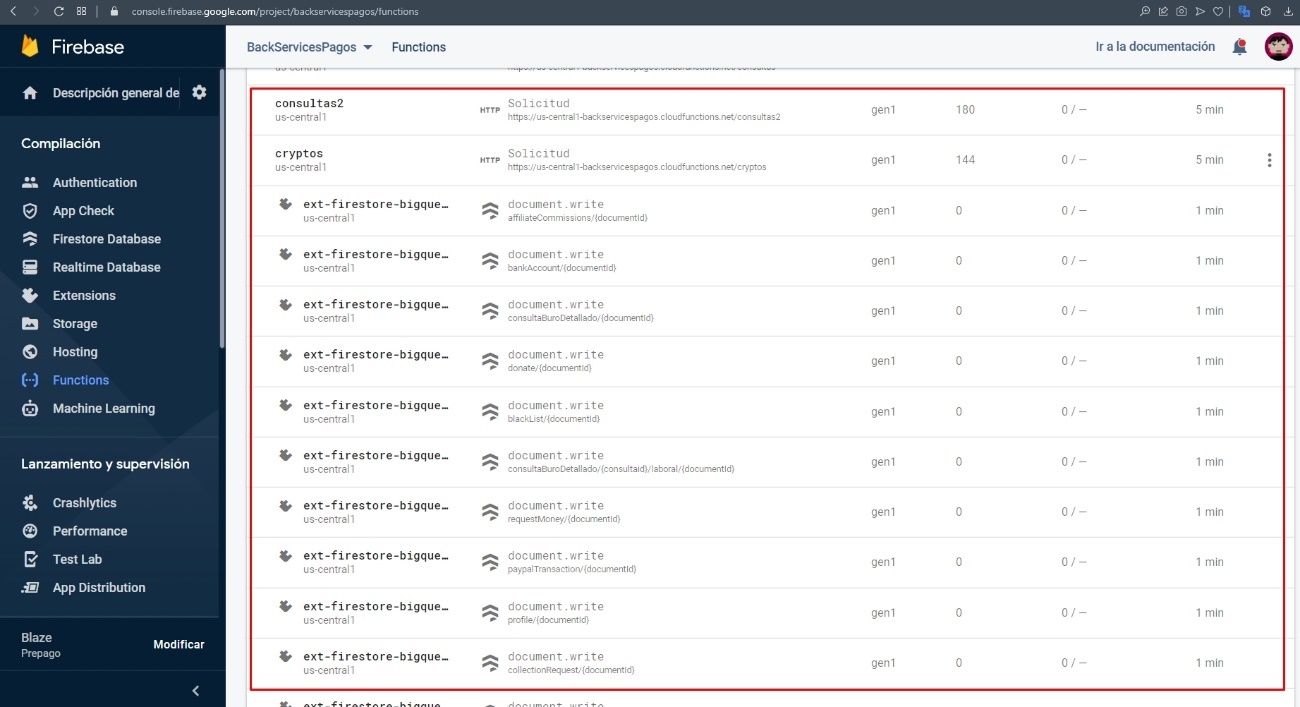


Figure 15: Https endpoint and bigquery extensions with Firebase Functions

Con respecto a IOTA, en la figura 16 se ilustra el endpoint de tipo https llamado “iota” utilizado para las funcionalidades donde involucren almacenamientos en IOTA, también existe una función de tipo Cloud Firestore llamado “notification” utilizado para el envío de notificaciones que se ejecutarán automáticamente cuando los usuarios terminen de realizar pagos y finalmente funciones de tipo Cloud/Pub programadas para que se ejecuten cada cierto tiempo y realizar alguna tarea en específica.

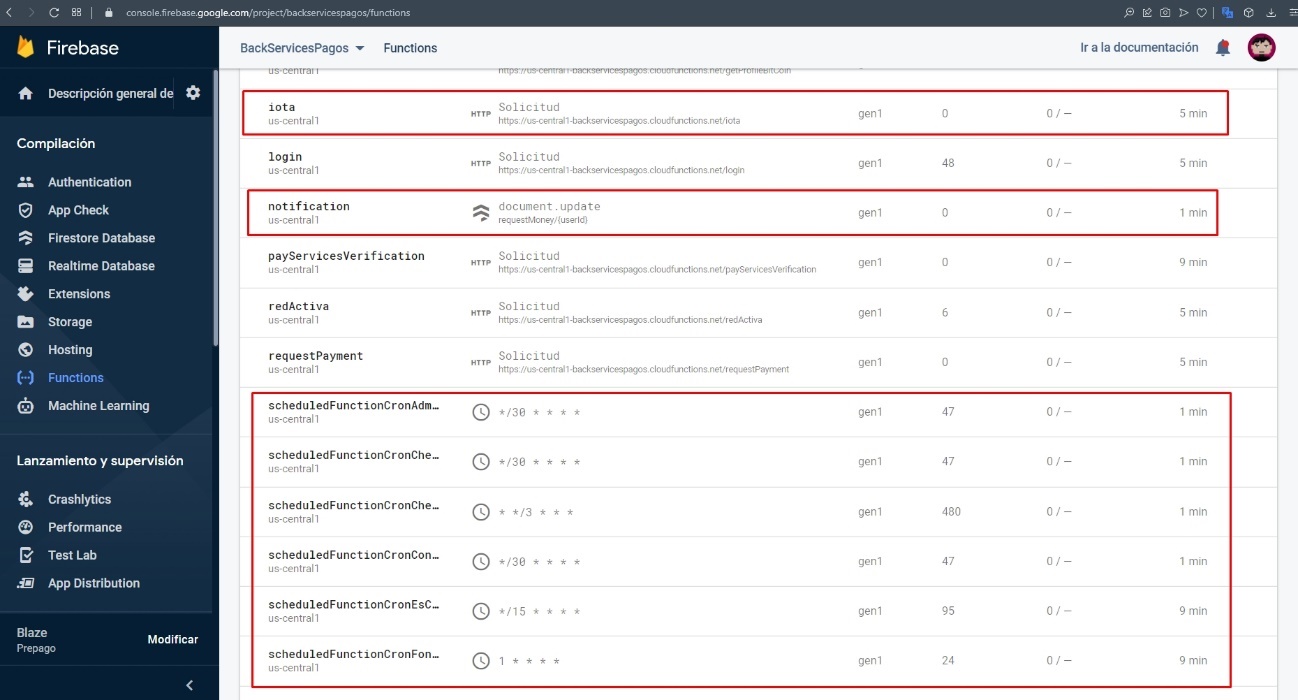


Figure 16: IOTA endpoint https, notification with Cloud Firestore and Cloud Pub/Sub

Otros de los microservicios relevantes implementados en esta investigación fueron los triggers de tipo Cloud Firestore, en la figura 17 se ilustra los diferentes triggers que se ejecutarán dependiendo de los cambios que se realicen en los documentos de Firebase por las acciones de los usuarios en las aplicaciones clientes.

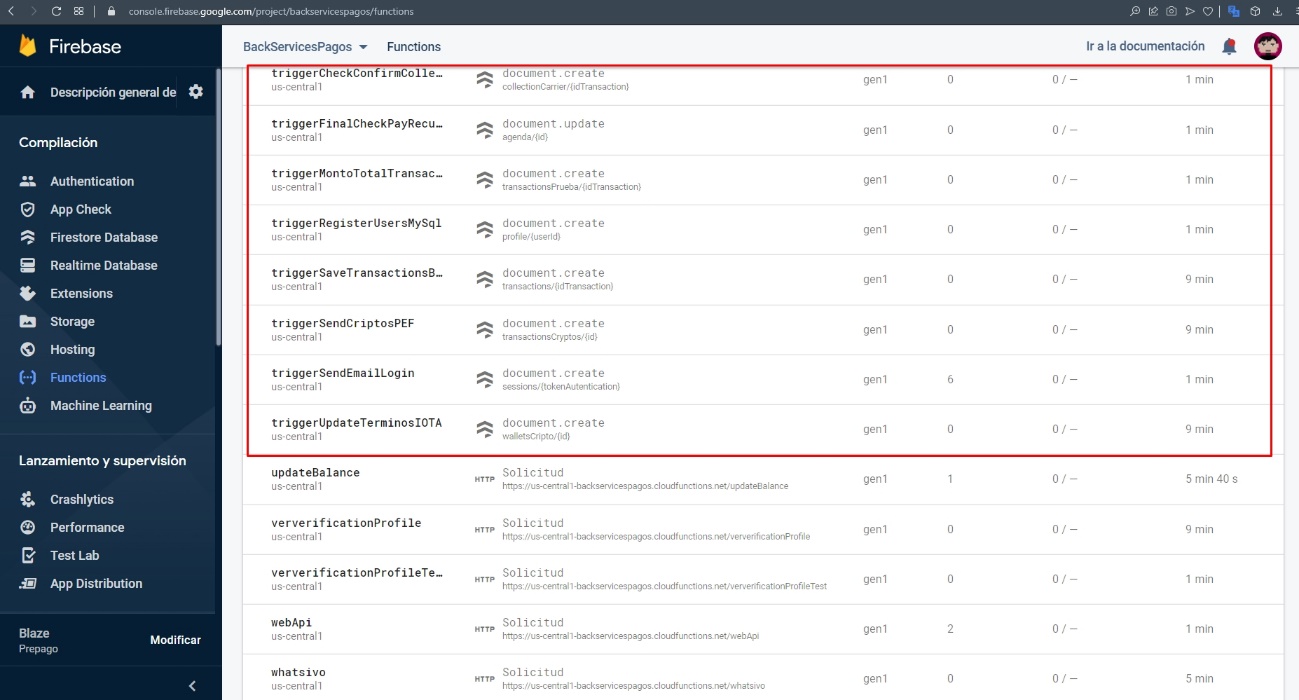


Figure 17: Triggers with Cloud Firestore

4.3 Frontend interfaz de usuario

A continuación, se presenta las interfaces más importantes realizadas en las diferentes funcionalidades y artefactos. La aplicación móvil ilustrada en la figura 18 fue realizada con el framework Angular 12 e IONIC 5.

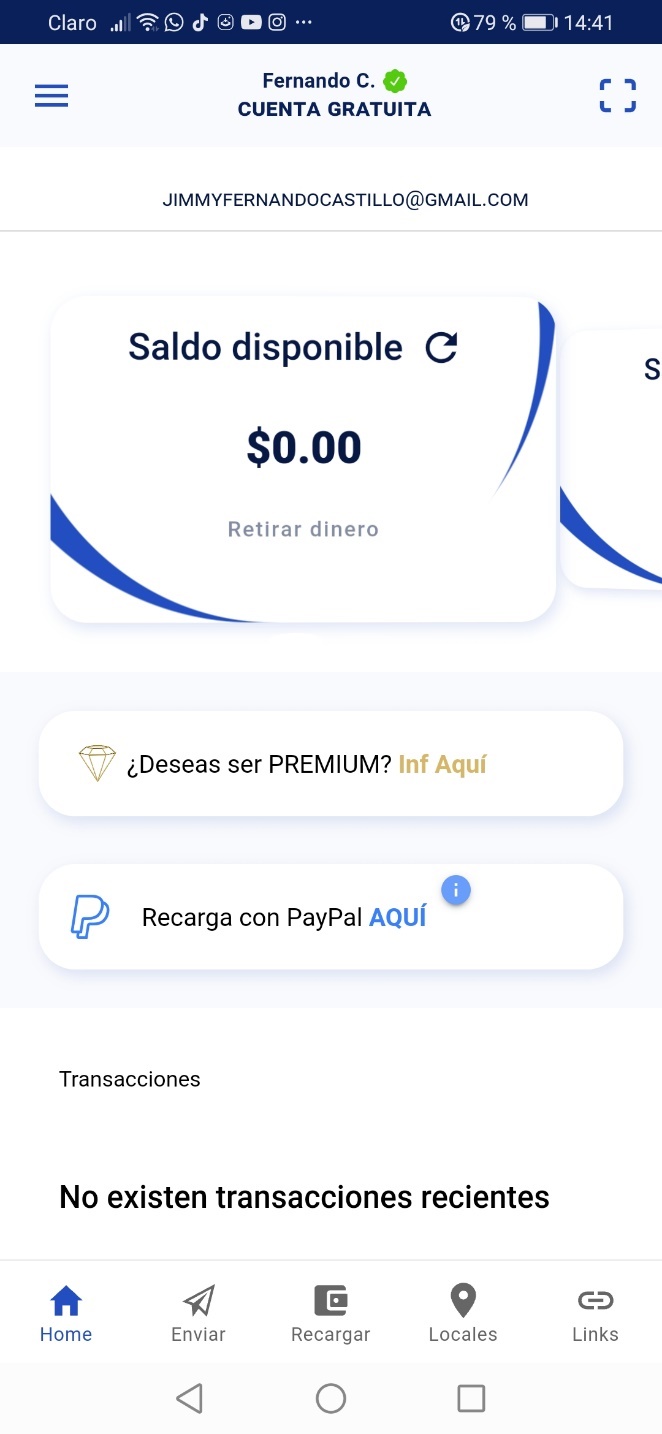


Figure 18: Mobile App Pay2Meta

La funcionalidad de links de pagos ilustrada en la figura 19 fue implementada en una instancia de Google Cloud y desarrollado con el framework Laravel 8 como tecnología backend, los frameworks Livewire 2 y VueJS 3 como tecnología frontend y Firebase y Mysql como bases de datos.

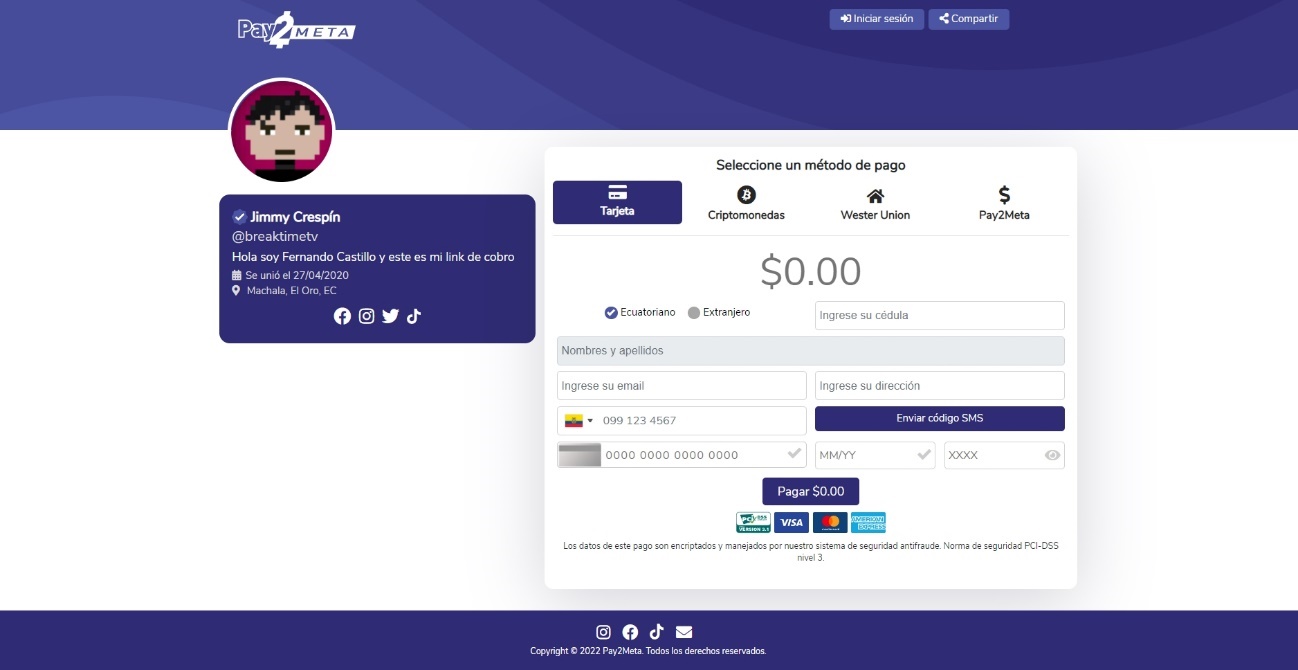


Figure 19: Functionality Payments Links

El artefacto del backoffice ilustrada en la figura 20 fue implementada igualmente en una instancia de Google Cloud con el framework Laravel 8 como tecnología backend, los frameworks Livewire y VueJS 3 como tecnología frontend y firebase como base de datos. En esta ventana los usuarios realizan las recargas de billeteras para tener saldo dentro de la plataforma.

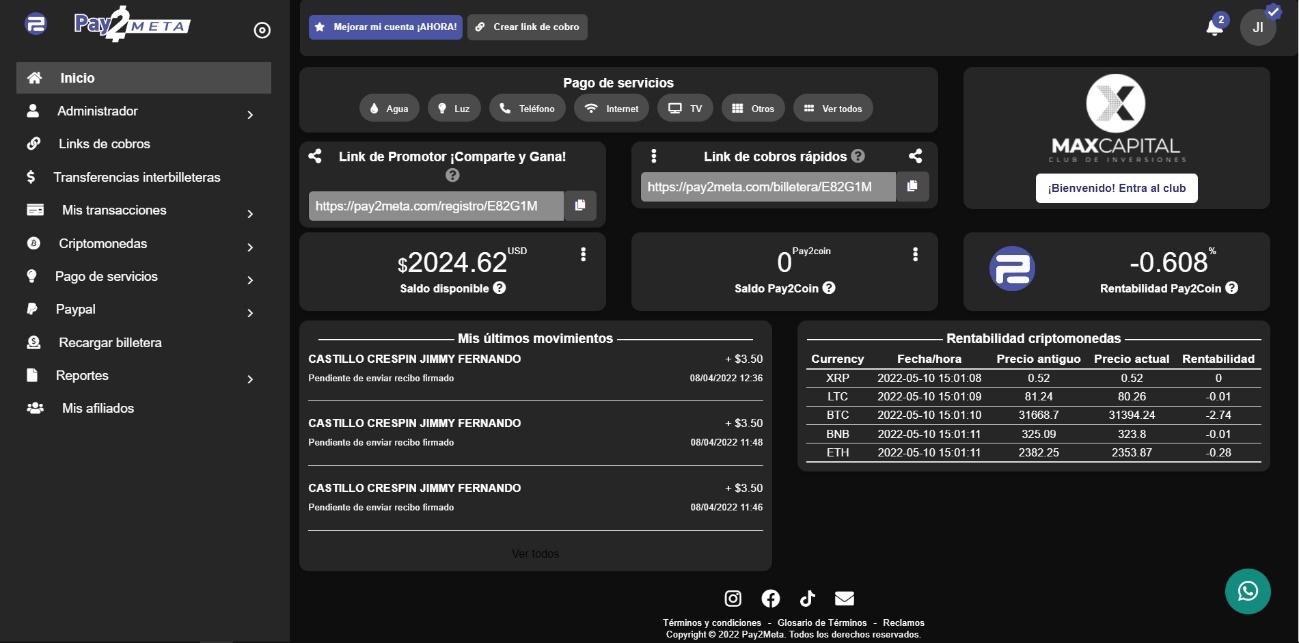


Figure 20: Web Application Backoffice Pay2Meta

El artefacto del trading con criptomonedas ilustrada en la figura 21 fue realizada con el framework Laravel 8 como tecnología backend, los frameworks Livewire y VueJS como tecnología frontend, firebase como base de datos y Tatum como tecnología DLT.

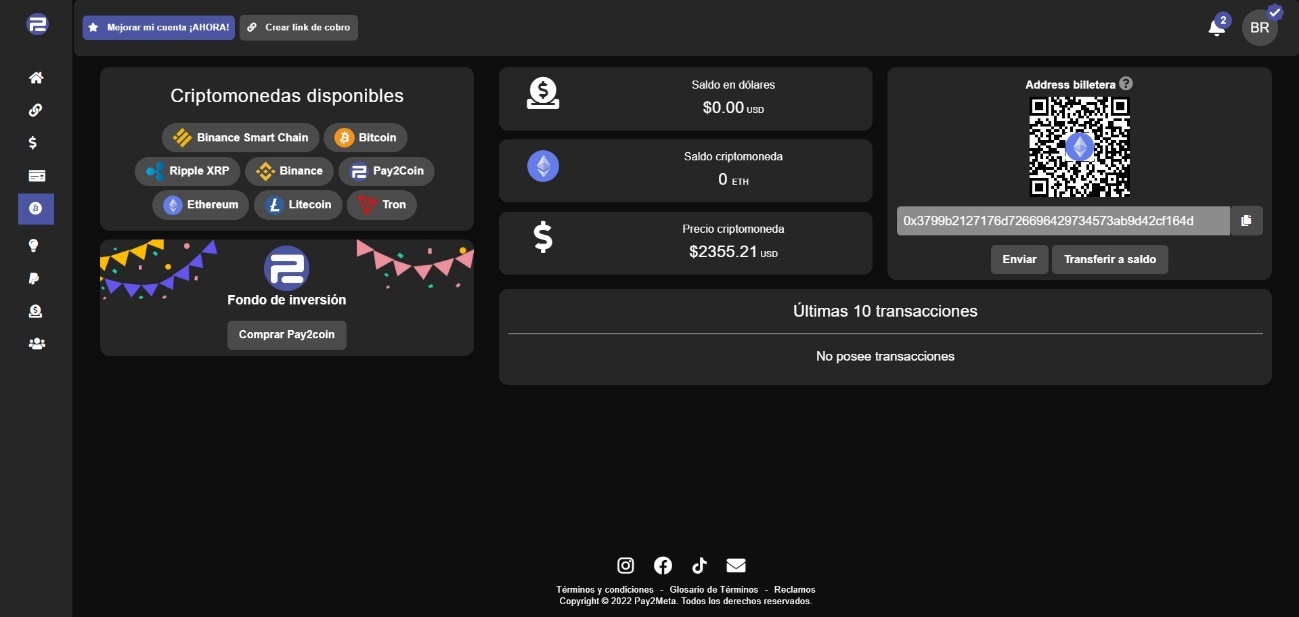


Figure 21: Web Application Marketplace Cryptocurrencies

4.4 Backend diagrama de procesos

Fueron tres diagramas de procesos elaborados en esta investigación, el primero se ilustra en la figura 22 el cual consiste en el proceso de verificación biométrica de los usuarios, para eso se utiliza un sistema externo llamado Mati. Cuando termina la verificación, se procede a crear un QR de los datos del perfil del usuario verificado y esta imagen posteriormente se convertirá en una identidad digital utilizando un smart contract con Iotex y NFT’s de la plataforma Tatum y el resultado de esto se almacenará en IOTA para asegurar su inmutabilidad. Cabe recalcar que, para realizar está identidad digital, los usuarios previamente deberán tener una billetera de criptomoneda de Ethereum, la misma que se puede crear en la plataforma Fintech estudiada y esta billetera deberá tener al menos un saldo de 5 centavos en ether para deployar el smart contract en Iotex.

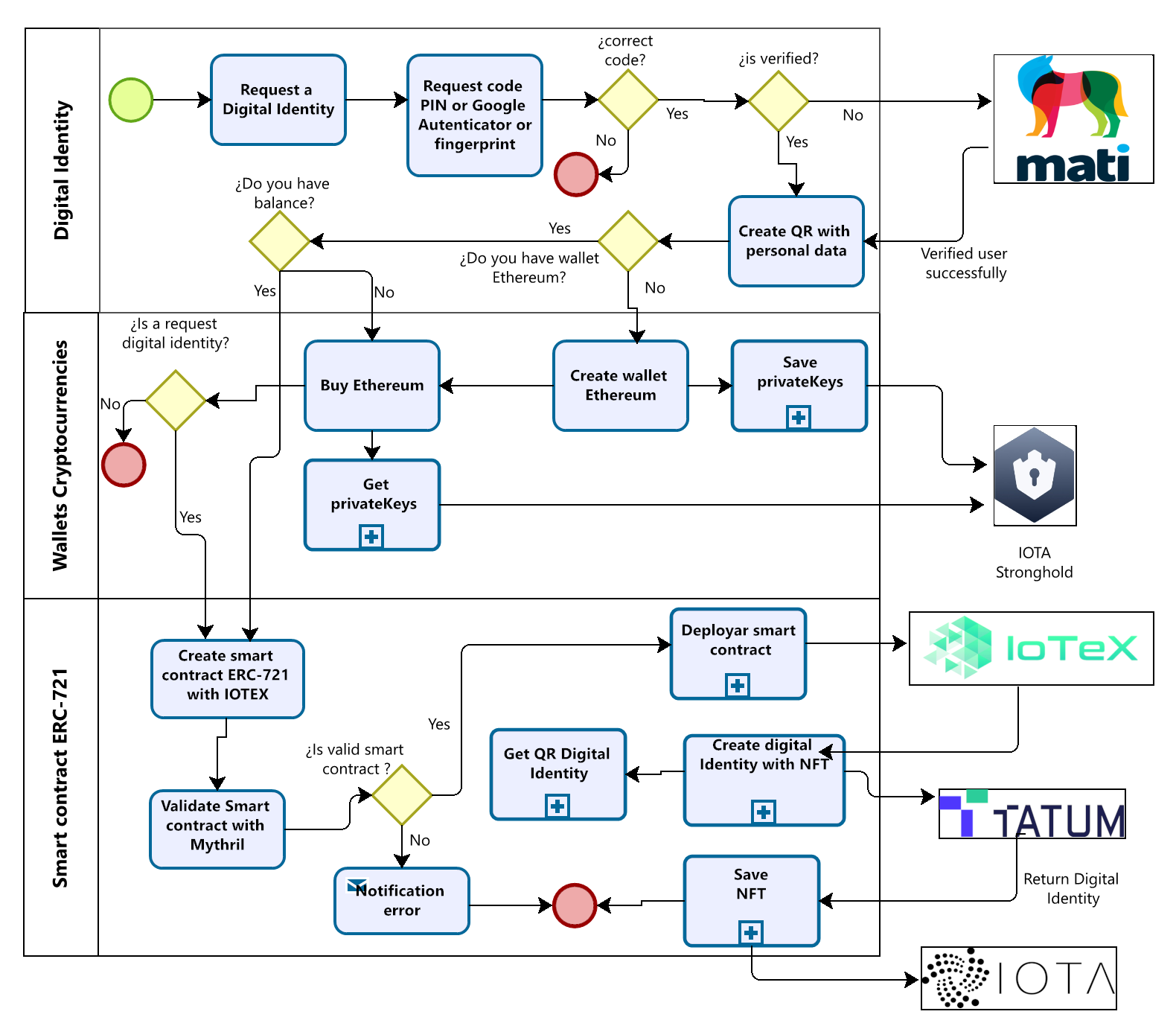


Figure 22: Digital Identity with NFT

El segundo proceso ilustrado en la figura 22 es acerca de la funcionalidad de recargar billetera ofrecida por la aplicación Fintech estudiada en esta investigación. Esta funcionalidad fue escogida por el motivo de que involucra pagos con tarjetas de crédito o débito de los usuarios verificados biométricamente para utilizar los NFT’s creados anteriormente y en conjunto con los datos de la transacción almacenadas en IOTA; tratar de ganar las disputas financieras provocados por fraudes ofreciendo está información adicional a los bancos.

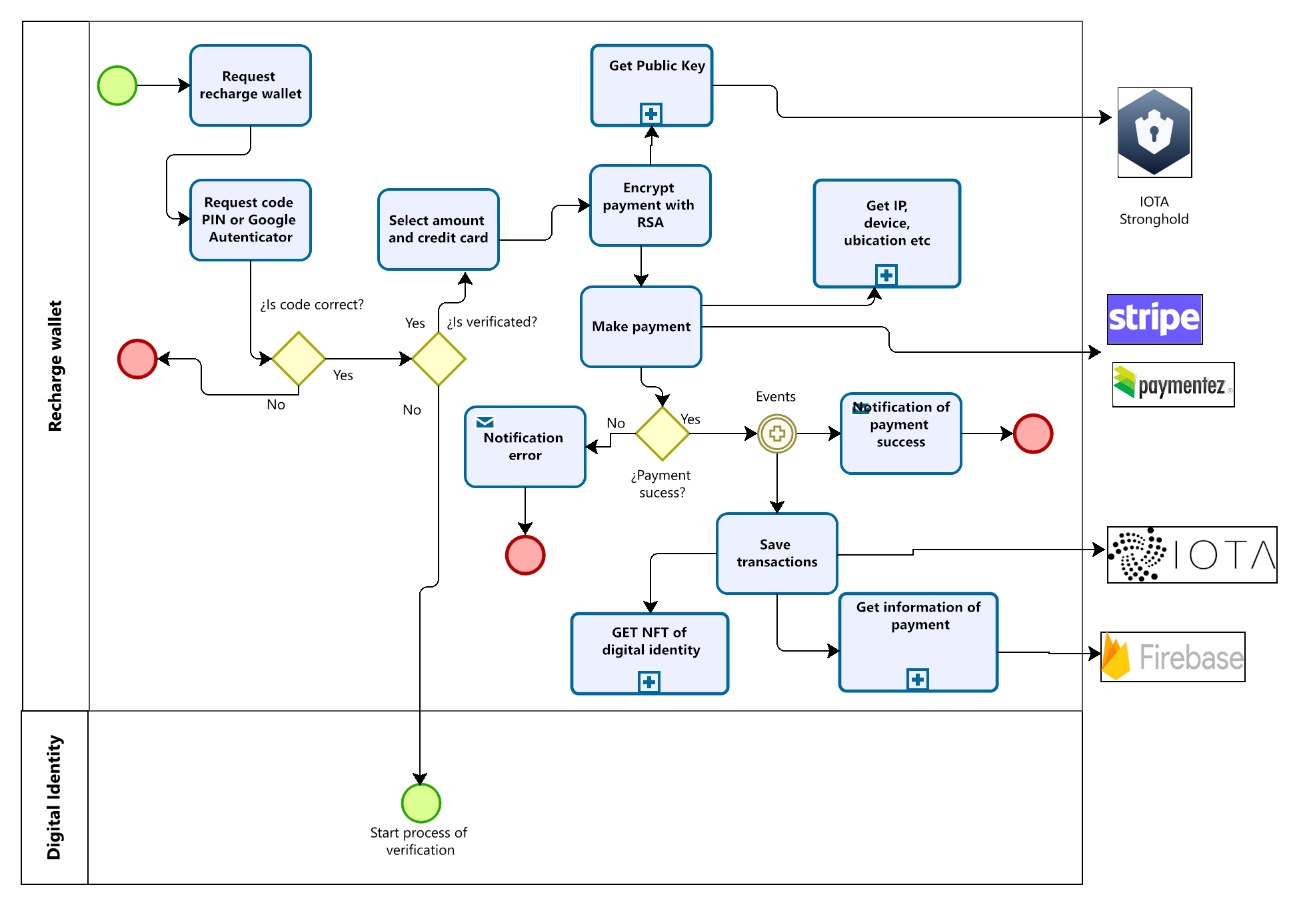


Figure 23: Process of recharging wallet with NFT and IOTA

Finalmente, el tercer proceso ilustrado en la figura 24 es acerca de la funcionalidad de links de cobros y marketplace ofrecidas por la aplicación Fintech estudiada en esta investigación. Estas funcionalidades fueron escogidas por el motivo de que involucran a usuarios externos que no poseen una identidad digital. Para esta solución se utilizó contratos inteligentes ERC-20 donde el comercio deberá enviar al administrador de la plataforma algún documento firmado por parte del usuario que realizó la compra, que indique que el producto vendido haya sido enviado al cliente. Una vez que se cumpla con este acuerdo, el contrato inteligente se ejecutará y se depositará el dinero en criptomonedas al comercio. La transacción resultante se almacenará en IOTA para asegurar su integridad e inmutabilidad.

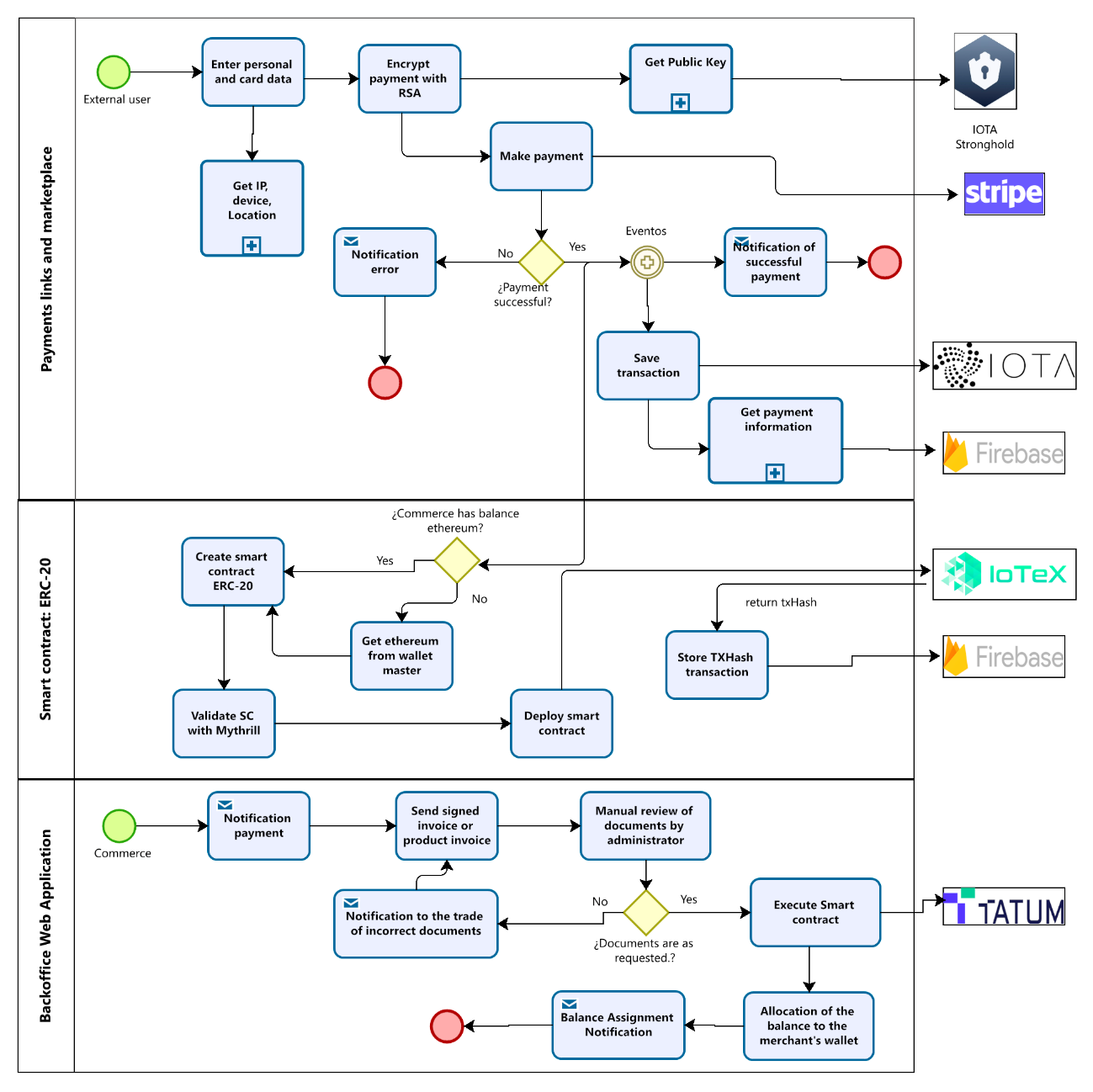


Figure 24: Buying - selling process on Marketplace

4.5 Subsistemas DLT

Para el registro de transacciones de coste cero con IOTA se utilizó el código de programación ilustrada en la figura 25 donde se detallan aspectos como la ip, características del dispositivo, coordenadas de ubicación, nombre del navegador, sistema operativo, fecha, hora, userId y la cantidad de la transacción que el usuario realizó en la plataforma Fintech Pay2Meta. Estos datos servirán posteriormente como prueba transaccional inmutable en caso de reportar fraude por parte de la entidad bancaria.

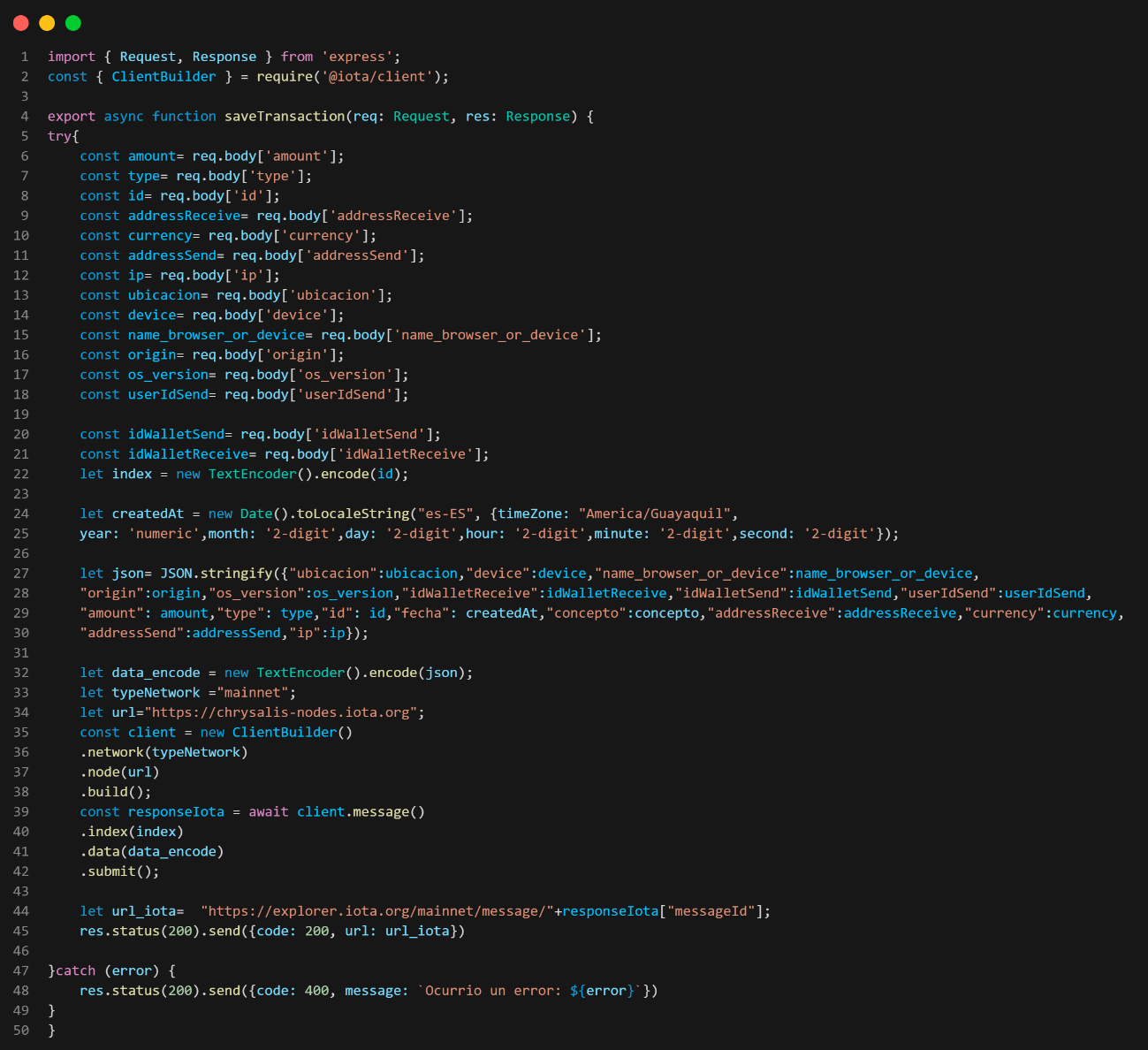


Figure 25: Programming code to store a transaction in IOTA

Con respecto a los smart contract ERC-20 se utilizó el lenguaje de programación solidity, en la figura 26 se ilustra una porción del código utilizado para las transferencias de criptomonedas de Ethereum cuando se cumple los acuerdos establecidos por el comercio y los compradores en los marketplaces de productos y de criptomonedas disponibles en la aplicación Fintech Pay2Meta.

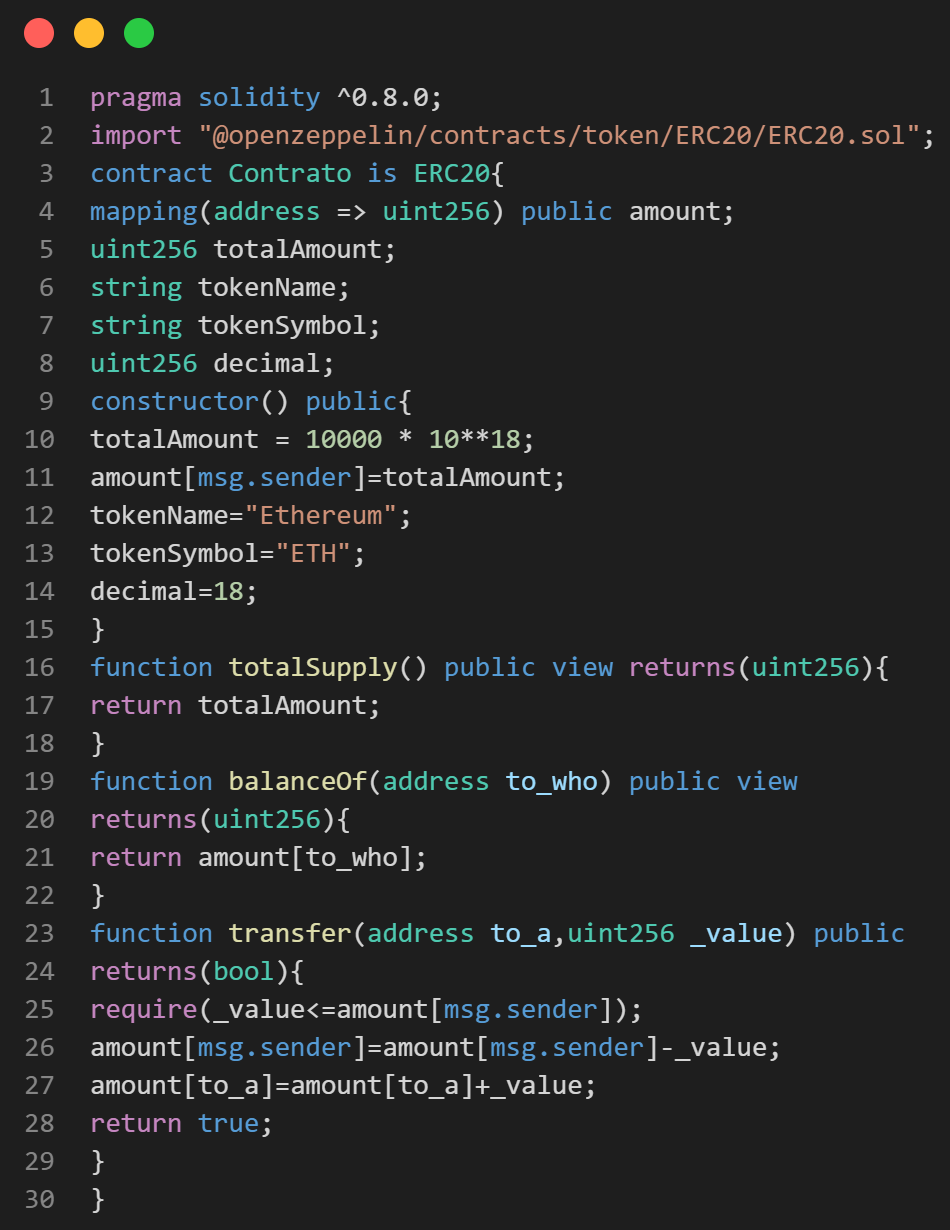


Figure 26: Programming code for smart contract ERC-20

Con respecto a los smart contract ERC-721 se utilizó el lenguaje de programación python, en la figura 27 se ilustra una porción del código utilizado para la creación del NFT el cual obtiene de un archivo json todos los IPFS con las imágenes de los códigos QR de la identidad digital de los usuarios.

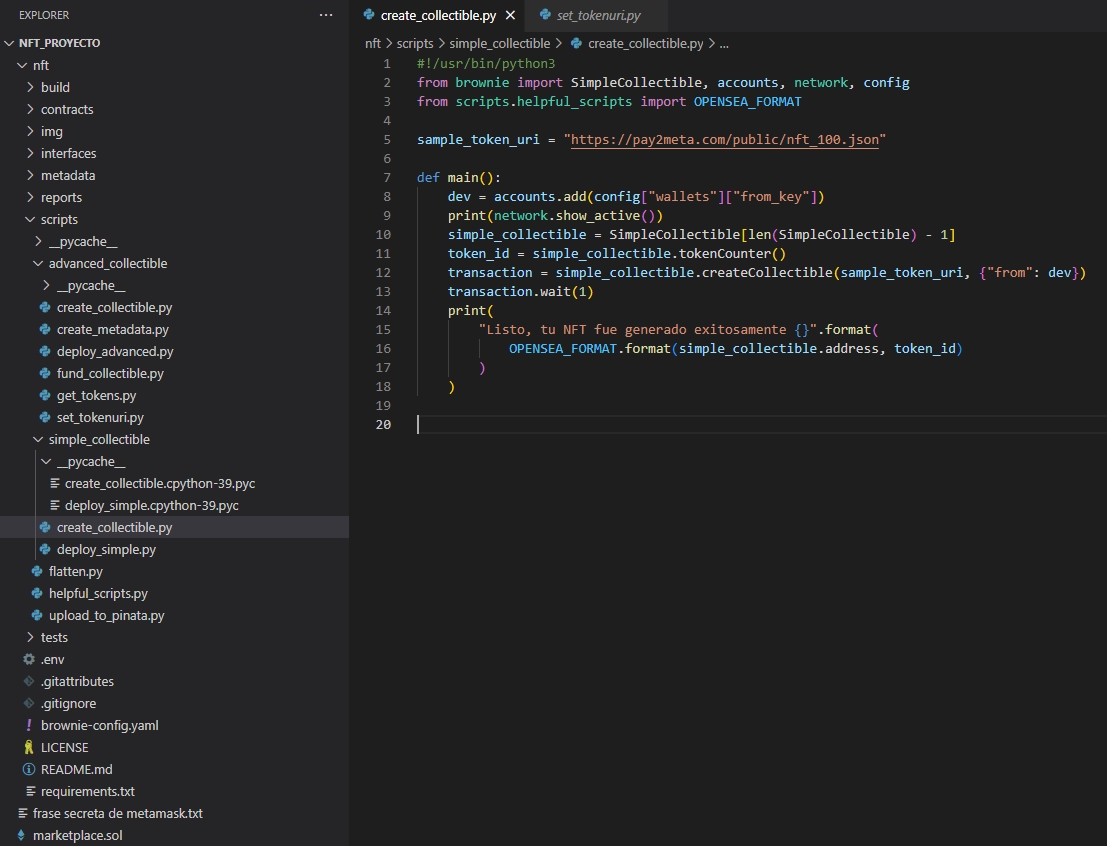


Figure 27: Programming code for smart contract ERC-721 in Python