Desarrollo de la plataforma PUNANA bajo la arquitectura de microservicios para la tokenización de activos digitales sobre Blockchain en Interia Group SAS.

Nayro Emanuel Garzón Toro

Informe de trabajo de grado como requisito para optar al título de :
Ingeniero de Telecomunicaciones

Tutor
Luis Alejandro Fletscher Bocanegra, Phd
Profesor Facultad de Ingeniería

Carlos Andrés Serna Carvajal
Director de Tecnología Interia Group SAS

Universidad de Antioquia
Facultad de Ingeniería
Ingeniería de Telecomunicaciones
Medellín, Antioquia, Colombia
2022

Resumen

La startup InteriaGroup SAS requería una solución tecnológica para el lanzamiento de una aplicación web (PUNANA) que permitiera realizar inversiones inmobiliarias mediante la tokenización de activos digitales. El frontend de la aplicación se desarrolló sobre Reactjs, utilizando la librería material ui para adicionar componentes visuales a la página, y el backend se manejó con el entorno de capa de servidor Nodejs. Se realizó la integración del frontend con el backend mediante las variables de entorno que permiten definir las url mediante las que se hacen peticiones al backend dependiendo del entorno en el que se trabaje (desarrollo, prueba, producción).

Para el despliegue de la aplicación web PUNANA en internet se utilizó la plataforma de Google Cloud mediante sus servicios Cloud Run y Cloud Build; el primero permitió el manejo de microservicios en contenedores y el segundo automatizar el despliegue a producción mediante una conexión directa con una rama existente en el repositorio Git. Para el manejo de la tokenización de activos, se realizó una investigación en la cual se definió el uso de la librería web3js para el llamado de los smart contracts codificados en el framework Solidity.

Introducción

Debido al impacto tecnológico y económico que ha traído consigo el surgimiento de las criptomonedas como primer caso de uso de la tecnología blockchain, se ha dado lugar a nuevas formas de comercializar, las cuales revolucionan el sistema financiero tradicional que opera de forma centralizada. Blockchain es un sistema descentralizado que da lugar a un nuevo tipo de divisa digital, la cual no está bajo el control de ninguna entidad financiera, se extiende a través de la internet y está soportada por activos digitales criptográficos, confiables e inmutables, que a través de Smart Contracts representan activos tangibles o intangibles del mundo real, como por ejemplo los bienes raíces.

A medida que la tecnología blockchain muestra sus fortalezas como la veracidad, confiabilidad e inmutabilidad de la información, garantiza la seguridad a través del cifrado y descentralización de los datos y potencializa el sistema tradicional financiero. Por esto las empresas empiezan a incursionar en este tipo de tecnologías emergentes, comerciando con diferentes "cripto activos digitales" los cuales son enviados sin intermediarios y en cuestión de segundos hacia cualquier lugar del mundo. Adicionalmente, implementan soluciones basadas en blockchain para el manejo transparente de la información y los recursos y/o aumentar la liquidez de capital de las compañías. Es allí donde la

Startup Interia Group SAS con su producto de tokenización NEKOT, desarrolla plataformas que permiten tanto a instituciones financieras como a personas naturales, tokenizar activos financieros de manera fácil y segura. Interia Group se esfuerza entonces en crear herramientas y soluciones personalizadas mediante el respaldo de la tecnología blockchain.

De esta manera, en alianza entre Interia Group y la agencia inmobiliaria a nivel global Century21, surge la idea de PUNANA, plataforma proptech sobre blockchain que permitirá ofrecer oportunidades de inversión en el sector inmobiliario. PUNANA ofrece una opción accesible para realizar inversiones inmobiliarias fraccionadas a través de la tokenización de activos, permitiendo a sus usuarios invertir desde cualquier lugar del mundo, montos más acordes con su presupuesto de inversión. Para el desarrollo de esta plataforma se estudió la arquitectura más acorde a implementar, definiendo los componentes que interactúan entre sí para lograr cumplir con los requerimientos de la solución tecnológica. Una vez definida la arquitectura pasamos a la etapa de despliegue de la plataforma, en dónde se realiza el desarrollo, todo bajo el modelo de microservicios. Cabe resaltar que para todo el desarrollo web se utilizará software open source y el despliegue de la infraestructura de microservicios bajo el runtime de docker en la nube de google cloud.

La participación como practicante en este proyecto, requirió de un aprendizaje autónomo acerca del core de negocio de la empresa (tokenización de activos sobre blockchain) y un aprendizaje guiado en el cual se establecen las bases para el desarrollo web, el manejo de los frameworks a utilizar y todos los servicios que presta la plataforma google cloud.

Objetivo general

Desarrollar el Front e implementar la arquitectura funcional de la plataforma web PUNANA para la startup Interia Group SA, mediante la integración de software open source, servicios de Google Cloud y Blockchain públicas.

Objetivos Específicos

Apropiar las potencialidades de la librería React para establecer las bases de desarrollo web, la integración con el Backend y la interconexión con los smart contracts y la blockchain Binance Smart Chain.

Evaluar qué tipo de servicios prestados por la plataforma de google Cloud pueden ayudar a la estructuración de la arquitectura de PUNANA.

Desarrollar bajo la metodología SCRUM los componentes asignados por el grupo de trabajo, incluyendo maquetación, elaboración de funciones y consumo de microservicios.

Realizar pruebas y validar el funcionamiento de los componentes desarrollados en cada etapa del proyecto por los otros integrantes del equipo de trabajo, tanto en entorno local como en la nube.

Marco Teórico

Plataformas de soporte para el desarrollo de la solución

Dentro del presente proyecto se utilizaron diferentes herramientas enfocadas al desarrollo de software tanto para el front-end como para el backend, además de plataformas que permiten realizar el correcto despliegue y consumo de los microservicios a implementar. A continuación, veremos la definición de estas plataformas

Nekot [1]

Nekot es una plataforma desarrollada por Interia Group SAS que brinda soluciones tecnológicas basadas en blockchain permitiendo tokenizar activos tangibles e intangibles para aumentar la liquidez de capital en las compañías a través de la emisión de token descentralizados y la integración al ecosistema DeFi. Dentro de los requerimientos del proyecto está la integración de Nekot con PUNANA, con el fin de interconectar este y otros proyectos al interior de la empresa con una plataforma propia de tokenización de activos.

ReactjS [2]

React es una librería open source lanzada en 2013 y desarrollada por facebook, cuya finalidad es el desarrollo de interfaces de usuario interactivas, por lo que en la actualidad es una de las librerías frontend más populares en el mundo del desarrollo. Al no ser un framework completo como Angular [3], React necesita de librerías externas para el desarrollo de una aplicación web, pero gracias a su arquitectura permite que el desarrollador no esté sujeto a limitaciones y pueda adoptar sus propios estándares.

React utiliza JavaScript como lenguaje de programación, pero también entrega la posibilidad de trabajar con JSX [2], una extensión de la sintaxis de JavaScript que permite crear elementos y renderizarlos en el DOM

(document object model), con la particularidad de que estos están directamente ligados a la lógica del elemento y comparten un mismo archivo. A través del código JSX, React crea una representación en memoria del DOM denominada Virtual DOM, la cual permite renderizar en el DOM definitivo del navegador solo al componente que cambie su estado.

Otra característica importante de React es la creación de los Hooks [4], los cuales son funciones de javaScript que permiten crear y acceder al estado de un componente y a los ciclos de vida de React, permitiendo asegurar la estabilidad de la aplicación y simplificar la lógica en el manejo de estados.

En interia group SAS, React es utilizado tanto para desarrollar aplicaciones web como aplicaciones móviles gracias a su ecosistema flexible de desarrollo, el cual cuenta con un gran número de autores de bibliotecas y creadores de contenido, que permiten interconectar su entorno con otras bibliotecas y servicios alojados en la nube.

Google Cloud [5]

Consiste en una plataforma creada por Google Inc la cual brinda soluciones a través de tecnología almacenada en la nube, ofreciendo rapidez de comunicación, escalabilidad en el consumo de los servicios, almacenamiento y seguridad. La computación en la nube permite que tanto el hardware como el software coexistan remotamente, trabajando en conjunto para ofrecer determinados servicios, por lo que Google Cloud ofrece un gran número de soluciones que pueden ser solicitadas con base en un servicio específico que se requiera (Redes, Bases de datos, Inteligencia Artificial etc) [6], o mediante requerimientos computacionales primarios ofreciendo tres tipos de computación en la nube [7]:

Iaas (infrastructure as a service): Los usuarios obtienen procesamiento, sistemas de almacenamiento, red y otros recursos computacionales, pero corren el sistema operativo y aplicaciones de acuerdo con sus necesidades.

Paas (platform as a service): Los usuarios escriben sus aplicaciones mediante lenguajes de programación y herramientas soportadas por proveedores y las ejecutan en una plataforma alojada en la nube.

Saas (software as a service): En este caso los proveedores de software corren sus programas en la nube, permitiendo a los usuarios acceder a estos programas a través de interfaces ligeras en los dispositivos de los clientes.

Al interior de Interia Group SAS, la infraestructura de los proyectos es desplegada en la nube de Google debido al tipo de soluciones que ofrecen, las cuales se ajustan con las demandas de los proyectos de la empresa, realizando el despliegue de aplicaciones y la interconexión con la plataforma en la nube mediante soluciones orientadas al manejo de microservicios.

Smart Contract bajo solidity [8]

Solidity es un lenguaje de programación de alto nivel mediante el cual es posible desarrollar smart contracts bajo los estándares ERC20, 721 u 1155; diseñado inicialmente para la red Ethereum y aplicable a otras redes desplegadas bajo un fork de Ethereum, como son Binance Smart Chain BEP20, BEP721, BEP1155.

Los smart contracts son aplicaciones que se implementan mediante la tecnología blockchain y se ejecutan de manera autónoma como parte de un sistema de validación de transacciones. Para llevar a cabo un contrato, se ejecuta una transacción de creación especial, que introduce un contrato a la cadena de bloques, siguiendo los protocolos descritos por la tecnología blockchain [9].

Al interior de la empresa, se trabaja con este framework para controlar y definir las reglas de negocio aplicadas a la solución y que permiten llevar todas las condiciones de cada caso de uso o necesidades de los clientes a la digitalización tecnológica en la Blockchain de Bainance Smart Chain.

Microservicios con el runtime de docker [10]

Docker es una plataforma abierta mediante la cual se despliega de manera ágil y rápida el envió y ejecución de software, por lo que está diseñada para permitir entregar aplicaciones mediante el uso de una plataforma ligera de virtualización de contenedores, la cual al interactuar con un conjunto de herramientas y flujos de trabajo, permiten llevar un control más eficiente de una aplicación.

La arquitectura de Docker es de tipo cliente-servidor, y está conformada por los siguientes elementos [11]:

Docker daemon: Escucha las solicitudes de la API de Docker y administra objetos de docker como imágenes, contenedores, redes y volúmenes.

Cliente: Es la manera principal en la que muchos usuarios interactúan con el docker daemon mediante el uso de comandos propios de la plataforma.

Registro: Es un registro público en el cual se almacenan imágenes de Docker de diferentes distribuidores de software, además de permitir subir y descargar imágenes propias.

Imágen: Una imágen es una plantilla de solo lectura con las instrucciones para generar un nuevo contenedor. Usualmente las imágenes se basan en otras imágenes con alguna personalización adicional.

Contenedor: Es una instancia ejecutable de una imagen que cuenta con un ciclo de vida en el cual se puede crear, iniciar, detener, mover o eliminar un contenedor mediante la API o la CLI de Docker. Un contenedor se define por su imagen, así como por las opciones de configuración que le proporcione al crearlo o iniciarlo.

Gracias a su arquitectura, esta plataforma permite mediante el uso de un conjunto de comandos implementar, probar y ejecutar aplicaciones y/o microservicios en diferentes instancias aisladas de nuestro espacio de trabajo, situación ideal para soluciones de alta transaccionalidad y/o cumplimiento de SLAs (acuerdos de niveles de servicio) en la disponibilidad del servicio tecnológico; por lo que este tipo de herramienta es indispensable al interior de la empresa para la realización de pruebas de consumo de microservicios y para el lanzamiento final de la solución.

Metodología de trabajo

Además de las plataformas que permiten el desarrollo de software y el consumo y alojamiento de servicios, es importante definir la metodología de trabajo adoptada al interior de la empresa.

Scrum [12]

Es un marco que permite el trabajo colaborativo de un equipo, el cual se centra en metodologías que buscan la mejora continúa fomentando el trabajo ágil y colaborativo. Esta metodología se basa en el aprendizaje continuo y en la adaptación a factores fluctuantes que afecten el desarrollo de un proyecto, buscando que el equipo evolucione a través de la experiencia. Scrum tiene como finalidad ayudar al grupo de trabajo a adaptarse de forma natural a las condiciones cambiantes y a los requisitos de los usuarios mediante el cambio de prioridades en un proceso y ciclos de lanzamiento cortos.

Cada una de las etapas de Scrum forma parte de una meta en común que busca satisfacer las exigencias y necesidades planteadas por los Project Manager, y al mismo tiempo, cumplir con los plazos de entrega de un proyecto. Scrum cuenta de 5 etapas de implementación [13]:

Inicio: En la primera etapa se analiza y se estudia el proyecto identificando las necesidades básicas del sprint. Un sprint es un mini proyecto con una duración no mayor de un mes que se interconecta con otros mini-proyectos para cumplir los objetivos generales del proyecto.

Planificación y estimación: En la segunda fase se crean e identifican tareas, buscando establecer metas fijas para cumplir con los plazos del proyecto. En esta etapa el líder de grupo o Master Scrum reparte las tareas correspondientes a cada grupo de trabajo, realizando estimaciones de entrega y clasificando tareas por prioridad.

Implementación: En esta fase se explora cómo optimizar el trabajo de cada grupo, buscando darle una forma definitiva al proyecto.

Revisión y retrospectiva: Es la fase de autocrítica y evaluación interna del grupo respecto a su propio trabajo, allí se realiza una validación del sprint y se aportan soluciones viables a inconvenientes presentados en la etapa de implementación.

Lanzamiento: En la última fase se ocupa del desenlace del proyecto y se realiza la entrega del producto.

En Interia Group SAS, se adopta está metodología para el trabajo dentro del equipo de desarrollo, y se lleva un control mediante la plataforma Jira [14], herramienta que permite la administración y gestión operativa de proyectos, especialmente dirigida a empresas dedicadas al desarrollo de software.

Metodología

Fase 1. Apropiación de las potencialidades de React

En esta fase se exploró la librería de React la cual fué utilizada para la maquetación de PUNANA, el consumo de servicios en la nube y la interconexión con entornos como la Bainance blockchain y la base de datos proporcionada por Google.

Actividad 1.1 Se realizó la Instalación y configuración de React en un entorno de desarrollo basado en kernel de linux.

Actividad 1.2 Se definió la estructura de carpetas, el manejo de componentes y la asignación de rutas al interior del proyecto.

Actividad 1.3 Se realizó una recopilación de las diferentes librerías y directivas que pueden ser implementadas en React además de su implementación.

Actividad 1.4 Se evaluaron diferentes alternativas para el manejo del backend y su interconexión con React.

Actividad 1.5 Se estudió la Interconexión entre smart contracts y la blockchain de Binance Smart Chain a través de la librería web3.

Fase 2: Evaluación de los servicios de Google Cloud

En esta fase se definieron los servicios ofrecidos por Google Cloud que serán parte del desarrollo de PUNANA, así como su implementación e interconexión con el framework implementado.

Actividad 2.1 Se llevó a cabo una familiarización con la plataforma de Google Cloud para la recopilación de servicios que fueron útiles para el proyecto.

Actividad 2.2 Se realizó la implementación de los servicios de google cloud en el framework de trabajo.

Actividad 2.3 Se definieron de los servicios a utilizar para el almacenamiento de datos y el manejo de apis al interior de PUNANA

Fase 3. Desarrollo de los componentes de PUNANA

En esta fase se realizó la maquetación de la plataforma, se implementaron las funciones y se consumieron los servicios necesarios para que PUNANA funcionara adecuadamente.

Actividad 3.1 Familiarización con la plataforma Jira y todos sus componentes, en dónde se llevó a cabo el control del desarrollo grupal de PUNANA.

Actividad 3.2 Se participó activamente en los espacios definidos por la metodología SCRUM y adoptados por el equipo de trabajo al interior de Interia Group SAS.

Actividad 3.3 Se realizó la maquetación de PUNANA de acuerdo con los diseños suministrados por el equipo de trabajo, bajo los estándares establecidos.

Actividad 3.4 Se desarrollaron las funciones para conectar la interfaz del usuario con los diferentes microservicios manejados al interior del proyecto.

Actividad 3.5 Se definió la lógica de negocio para los smart contracts codificados en solidity a través de los estándares BEP20 y BEP1155.

Actividad 3.6 Se realizó el llamado de los smart contract y la invocación de cada una de sus funciones a través de la librería web3 desde el front de la solución.

Fase 4: Ejecución de pruebas y validación

En esta fase se realizaron las pruebas de los componentes desarrollados de acuerdo con los lineamientos establecidos.

4.1 Se realizaron las pruebas en las diferentes etapas del proyecto de acuerdo con la asignación de roles establecida en las reuniones del equipo de trabajo.

4.2 Se llevó un control del desarrollo del proyecto en github, permitiendo generar un ambiente de pruebas en la nube, previo al despliegue de la versión de producción de PUNANA.

Resultados y análisis

El desarrollo de la plataforma PUNANA está compuesto por la implementación del frontend, su conexión con el backend y la configuración realizada mediante los servicios de Google para ejecutar su despliegue mediante el uso de microservicios. En la figura 1 podemos ver un diagrama de bloques en donde se detalla la estructura de la plataforma, y de cómo se comunican sus elementos.

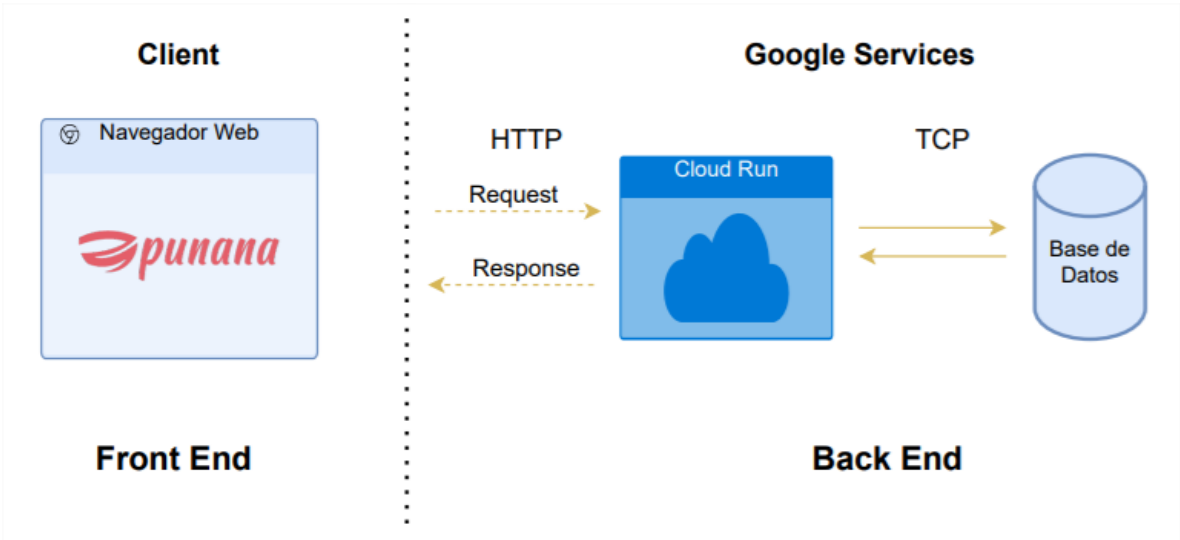


Figura 1. *Estructura de la Plataforma PUNANA*

A continuación, veremos con más detalle cómo se ha desarrollado cada uno de estos bloques y cómo se realiza la configuración para conectarse a la binance Smart chain, dando lugar a la tokenización de activos digitales en el mundo inmobiliario.

Desarrollo front end:

Luego de revisar la documentación acerca de React js (versión 17.0.2), se procede a realizar la instalación de la librería en un entorno de desarrollo basado en kernel de linux, Ubuntu 20.04.3 LTS. Como editor de código se utiliza Visual Studio Code, gracias a la cantidad de extensiones compatibles con react que este posee y sus herramientas integradas que permiten optimizar el desarrollo web. Una de las configuraciones principales para el desarrollo del frontend es la implementación de la librería Material-UI, la cual es una biblioteca que permite importar y usar diferentes componentes para crear interfaces de usuario en aplicaciones de React.

La landing page de PUNANA tiene como finalidad dar a conocer el mercado de inversión Inmobiliaria a partir de la tokenización de activos, mostrando como respaldo el grupo de desarrolladores y de inversionistas detrás del proyecto. En la tabla 1 podemos ver los componentes principales que han sido desarrollados:

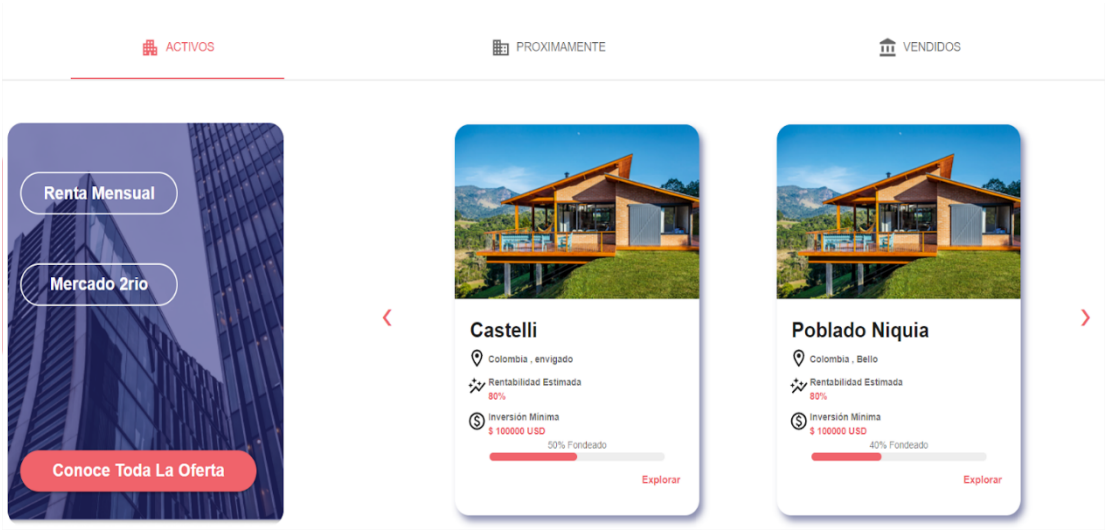
Tabla 1. *Componentes principales frontEnd*

Navbar	• Es la barra de navegación dentro de la aplicación y forma parte de la plantilla principal de PUNANA por lo que está posicionada de manera fija en la parte superior de la página web. En este componente podemos encontrar el logotipo de la aplicación y un menú con las rutas de login y de ayuda.
Home	• En el home de PUNANA, podemos encontrar un video introductorio acerca del uso de la plataforma y el manejo de tokenización de activos basados en bienes raíces.
About Company	• Recopilación de la información de la compañía y su perfil en el mercado

How Works	• Breve descripción del procedimiento de tokenización de activos en el mundo de los bienes raíces.
How Invest	• Procedimiento para realizar una inversión al interior de la plataforma PUNANA
Simulator	• Simulador online de inversión, para que los clientes puedan hacerse una idea acerca de la rentabilidad y las prestaciones de este tipo de inversiones sustentadas por activos digitales.
Opportunities	• Encontramos una lista de propiedades en las cuales se puede realizar inversión mediante la compra de token digitales, allí podremos filtrar las propiedades por su estado (activo, vendido) y el tipo de mercado en el que se encuentren.
Comparative	• Se encuentra una gráfica comparativa entre la rentabilidad de diferentes métodos de inversión con la rentabilidad pronosticada con la plataforma.
Team	• En este componente se encuentran los diferentes miembros de los equipos ejecutivos de mercadeo y desarrollo que hacen parte del despliegue y funcionamiento de la plataforma PUNANA.
Partners	• Recopilación de inversionistas y asociados con el proyecto
Footer	• El pie de página contiene información básica acerca de la compañía, los términos de uso e información de contacto.

Al tratarse de una aplicación web, y debido al gran número de usuarios que acceden a internet desde sus dispositivos móviles, es necesario establecer un diseño para cada componente que pueda adaptarse a diferentes tipos de resoluciones (mobile, tablet,laptop); en la figura 2 podemos ver una comparativa entre el diseño realizado para el componente 'Opportunities' en su versión web y mobile.

Versión web



Versión Mobile

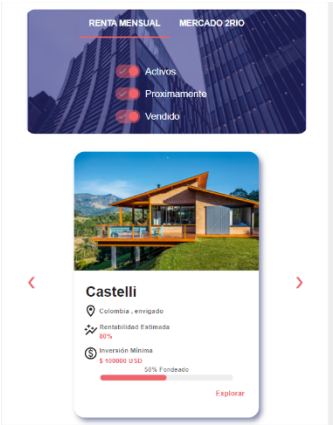


Figura 2. *Comparación componente Opportunities en versión web y mobile*

Podemos observar cómo el cambio de resolución, obliga a realizar una nueva distribución de elementos. En este caso, cambia el sistema de filtrado de las propiedades, pero siempre se mantiene una continuidad visual entre el diseño web y el diseño mobile.

Integración con el backend :

El backend de la aplicación web es desarrollado en el entorno de ejecución Node js. El backend contiene las rutas de las apis que son consumidas desde el frontend en la aplicación web. Las rutas principales son las encargadas de gestionar y validar el registro y la autenticación de usuarios además de las diversas consultas a las bases de datos que son necesarias para renderizar componentes en PUNANA. En la tabla 2 podemos encontrar las dependencias necesarias para el funcionamiento del backend:

Tabla 2. *Dependencias utilizadas en el backend*

Dependencia	**versión**	**Funcionalidad**
express	4.17.2	Framework de aplicaciones backend para Nodejs.
express-validator	6.14.0	Conjunto de MIddlewares que permiten realizar validaciones.
jsonwebtoken	8.5.1	Permite la creación de tokens de acceso para la propagación de identidad y privilegios en una aplicación web.
mongoose	6.1.4	Biblioteca de programación orientada a objetos para crear una conexión entre la base de datos MongoDB y el framework express.

La integración con el backend se dá mediante las consultas http realizadas desde el frontend a las rutas ya existentes en el backend, pero al tratarse de un proyecto tan completo, es necesario tener en cuenta diferentes entornos de trabajo como lo son el entorno de desarrollo donde las consultas se realizan de manera local, y el de pruebas y producción donde las consultas se realizan de manera remota. En la figura 3 podemos observar una petición al back donde se solicitan las propiedades localizadas en la base de datos para renderizar un componente en el frontend con su información.

Figura 3. *Petición http al backend*

La petición es realizada mediante la función de javaScript fetch() que permite realizar una petición http a una url válida, retornando una promesa la cual al resolverse permite manipular la información contenida en la respuesta a la petición; en este caso permite renderizar un componente con los datos de las propiedades disponibles para invertir.

Una forma de controlar el tipo de entorno de trabajo y la manera en la que se realizan las consultas al backend, es mediante las variables de entorno. Estas variables permiten asignar a una constante la url (uniform resource locator) de consulta a las apis ubicadas en el backend, y la cual será accedida dependiendo del entorno que se esté ejecutando.

Servicios de Google Cloud:

Una vez configuradas las variables de entorno en nuestro proyecto, es necesario definir un hosting o alojamiento en línea que permite publicar todo el sitio web de PUNANA en internet. Para esto hacemos uso de la plataforma de Google Cloud la cual tiene un gran número de soluciones en la nube que permiten desarrollar e implementar aplicaciones en contenedores altamente escalables.

Una de estas soluciones es Cloud Run, plataforma serverless ideal para el manejo de microservicios en contenedores que no requieren de un alto nivel de configuración. En la figura 4 podemos ver la configuración inicial de Cloud Run en la plataforma de Google Cloud. Allí se selecciona la opción de implementar de forma continua, para enlazar el servicio con un repositorio en Github, se configura el nombre del servicio y la región de acceso.(Se selecciona Carolina del Sur debido a que es la región con menor latencia para consultas desde Colombia).

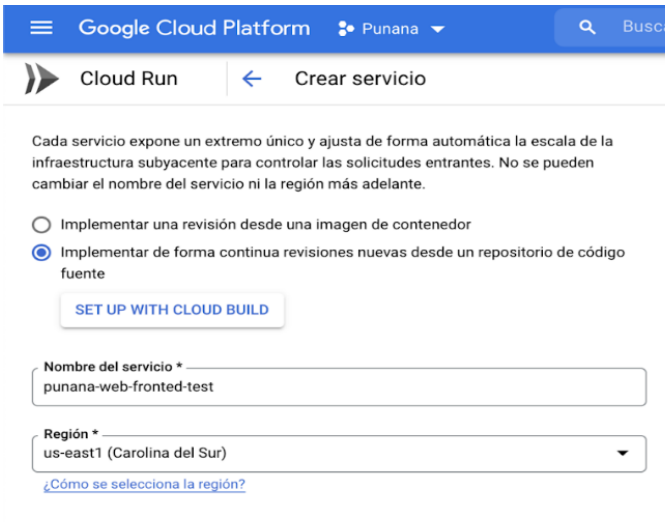


Figura 4. *Configuración inicial Cloud Run.*

Cloud Run simplifica la administración de la infraestructura mediante un escalamiento vertical automático dependiendo del tráfico en la red, además ofrece una mejor experiencia para los desarrolladores al simplificar el desarrollo e implementación de aplicaciones.

Gracias a la adaptación de esta plataforma para el despliegue de PUNANA, podemos hacer uso de la implementación continua desde Git mediante el servicio de Cloud Build, el cual nos permite automatizar la implementación realizada previamente en Cloud Run, con la finalidad de compilar e implementar el código de forma automática una vez se realicen cambios sobre una rama específica en el repositorio de Git. En la figura 5 podemos ver la configuración realizada para el despliegue continuo de la rama test ubicada en el repositorio de PUNANA.

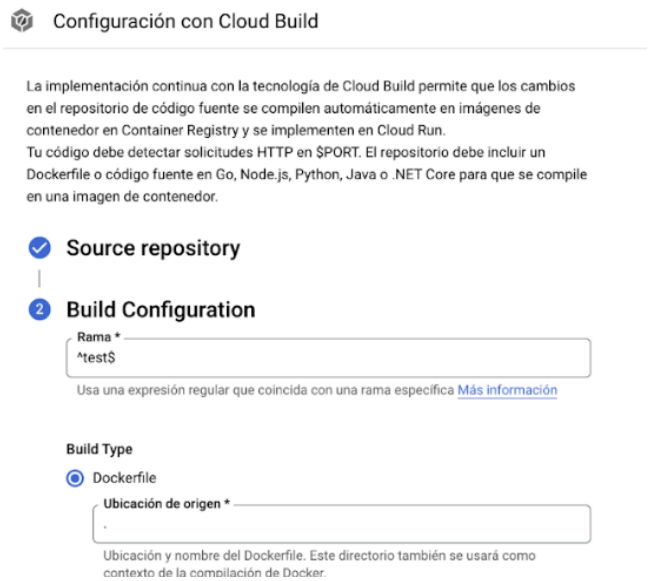


Figura 5. *Configuración Cloud Build para rama test*

Para hacer uso de este servicio, contamos con dos repositorios de Git independientes uno para el frontend y otro para el backend, donde ambos contienen un Dockerfile con la configuración necesaria para correr los microservicios dentro de los contenedores proporcionados por Cloud Run.

Conexión con la blockchain de Binance Smart Chain:

Una vez establecido el despliegue del frontend y el backend de PUNANA, es necesario abordar la interconexión con la blockchain a utilizar para llevar a cabo todo el sistema de smart contracts y tokenización de activos al interior de la plataforma, para esto se desarrolló un WebSocket para la conexión del frontend con la librería web3 permitiendo interactuar con los nodos de la blockchain BSC.

Binance Smarth Chain es una blockchain desarrollada por Binance y su comunidad que implementa una visión de un intercambio de activos digitales de manera descentralizada, donde el BNB es el token nativo de su ecosistema.

Al tratarse de un fork de ethereum, la Binance Smart Chain es compatible con Ethereum Virtual Machine, y para su implementación es necesario una herramienta que permita interactuar con un nodo de Ethereum tanto de manera local como remota.

Web3js es una colección de bibliotecas que permiten cumplir con esta tarea, dando la posibilidad de desarrollar aplicaciones cliente que interactúan con la blockchain de Ethereum. Web3js se comunica con Ethereum mediante el protocolo JSON RPC realizando peticiones a un nodo individual que pertenece a la red de Ethereum. En la figura 6 podemos ver la estructura de conexión de Punana con la red de Ethereum mediante el uso de Web3js.

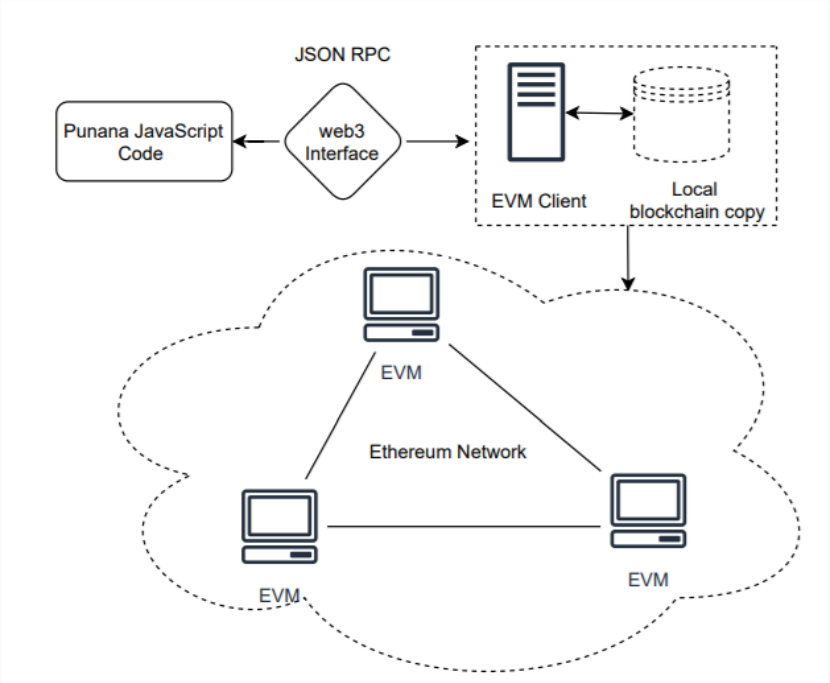


***Figura 6**. Conexión a la red de Ethereum mediante web3js*

Una vez listo el contrato inteligente a utilizar en nuestro proyecto, utilizamos esta librería para interactuar y llamar las funciones de dicho contrato, gracias a los métodos ya existentes y suministrados por el api.

Control de pruebas y manejo de Repositorio

En el desarrollo de PUNANA se utiliza GIT como sistema de control de versiones para llevar un seguimiento de las piezas de código que aporta cada miembro del equipo de trabajo. Esta es una herramienta de gran utilidad para el trabajo en conjunto, pero como vimos anteriormente en el despliegue del microservicio mediante Cloud build, el uso de git es fundamental para la compilación y despliegue a producción de la aplicación. En el proyecto se enlazan directamente dos ramas de los repositorios tanto de frontend como de backend con el servicio de Cloud build, una rama de prueba y otra rama de producción.

A medida que se desarrolla el código y se tienen partes funcionales de la aplicación, se compila el código existente en la rama test para realizar pruebas en la nube tanto en la parte de maquetación y diseño como en el consumo de apis, la interconexión con el backend y la base de datos, siendo estas últimas controladas por las ya configuradas variables de entorno.

La realización de pruebas está dividida en dos áreas, una de testing y una de QA. En el área de testing se realizan pruebas sobre el código ya existente, buscando posibles errores en la funcionalidad, en el diseño, la resolución y el consumo de apis, y una vez finalizado se retroalimenta al desarrollador para arreglar los bugs existentes. En el área de QA se realiza otro tipo de análisis y otro tipo de pruebas con la finalidad de prevenir posibles fallos futuros e inconsistencias en el flujo de la aplicación que puedan afectar el correcto funcionamiento de esta o la experiencia de usuario.

Una vez que la rama test ha sido probada, se procede a compilar y desplegar mediante Cloud build la rama máster, que es la versión de producción oficial de la aplicación que se mostrará a los usuarios para su interacción. Cabe destacar que las ramas de develop y test están en constante actualización, pero la rama master solo es actualizada una vez que se tiene una nueva versión, o cuando existe un inconveniente que debió ser solucionado rápidamente.

Conclusiones

- En mi participación en el proyecto PUNANA al interior de la Start up Interia Group SAS trabajé en el área de desarrollo frontend, diseñando y programando los componentes que hacen parte de la página web, aplicando los conocimientos de desarrollo adquiridos en mi formación académica. Además, tuve la oportunidad de realizar la configuración del backend y de llevar a cabo el despliegue de la plataforma mediante los servicios de Google Cloud.
- Reactjs fué la librería utilizada para el desarrollo del frontend gracias a su arquitectura y a todo el soporte brindado por su comunidad al ser una de las herramientas frontend más trabajadas a nivel mundial. Para su implementación estudié la librería, haciendo uso de sus Hooks que permiten controlar el estado de la aplicación y del lenguaje de programación jsx el cual permite combinar la lógica de javaScript para realizar el renderizado de componentes en el DOM.
- Para el despliegue de la plataforma se utilizó el servicio de Google Cloud Run, el cual permite realizar el manejo de microservicios en contenedores sin la necesidad de solicitar configuraciones avanzadas y brindando un escalamiento vertical automático dependiente del tráfico existente en la plataforma. Además de este servicio también utilizamos Cloud Build para automatizar los despliegues a producción enlazando el servicio de Google con el repositorio Github de la aplicación.
- Durante mi participación en el equipo de desarrollo trabajamos mediante la metodología SCRUM, en dónde participé activamente de los sprints realizados, aportando a la solución de problemas tanto en el área de diseño y maquetación, como en el área de desarrollo web.
- Al interior del equipo de desarrollo hay un ingeniero encargado de realizar el QA de la aplicación, por lo que mi participación en el área de pruebas estuvo enfocada en realizar el testing local, además de revisar que los componentes desarrollados se comporten de igual manera en el entorno de desarrollo y producción.
- Me capacité acerca de la conexión de la plataforma PUNANA con la blockchain de binance smart chain, que al ser un fork de la red de ethereum puede ser accedida mediante una librería compatible con dicha red. Web3js fué la librería seleccionada para realizar dicha configuración debido a que permite conectarse a la red de Ethereum mediante la realización de peticiones a un nodo individual perteneciente a la red mediante el protocolo JSON RPC.

Referencias Bibliográficas

[1] Interia Group SAS. (2021).Plataforma Nekot.Obtenido de

www.nekot.io

[2] de Sousa, M., & Gonçalves, A. (2020, June). humanportal–A React. js case study. In 2020 15th Iberian Conference on Information Systems and Technologies (CISTI) (pp. 1-6). IEEE.

[3] Hamerník, B. M. (2020). Development of Modern User Interfaces in Angular Framework.

[4] Reactjs.(2021).Presentando Hooks. Obtenido de

https://es.reactjs.org/docs/hooks-intro.htm

[5] Google Cloud.(2021).Descripción general de Google Cloud. Obtenido de

https://cloud.google.com/docs/overview

[6] Acronis.(2021. Google Cloud Platform. Obtenido de

https://www.acronis.com/es-mx/articles/google-cloud-platform/

[7] Ma, W., & Zhang, J. (2012). The survey and research on application of cloud computing. In 2012 7th International conference on computer science & education (ICCSE) (pp. 203-206). IEEE.

[8] Solidity-es(2021).Introducción a los Contratos Inteligentes. Obtenido de

https://solidity-es.readthedocs.io/es/latest/introduction-to-smart-contracts.html

[9] Wohrer, M., & Zdun, U. (2018). Smart contracts: security patterns in the ethereum ecosystem and solidity. In 2018 International Workshop on Blockchain Oriented Software Engineering (IWBOSE) (pp. 2-8). IEEE.

[10] Google Cloud(2021).Contenedores en Compute Engine. Obtenido de

https://cloud.google.com/compute/docs/containers

[11] Jaramillo, D., Nguyen, D. V., & Smart, R. (2016). Leveraging microservices architecture by using Docker technology. In SoutheastCon 2016 (pp. 1-5). IEEE.

[12] Atlassian.(2021).Scrum. Obtenido de

https://www.atlassian.com/es/agile/scrum

[13] Schwaber, K. (1997). Scrum development process. In Business object design and implementation (pp. 117-134). Springer, London.

[14] Jira Software. (2021). Funcionalidades para el desarrollo de software,Obtenido de

https://www.atlassian.com/es/software/jira/features

Anexos

Url test : https://punana-web-frontend-test-7j64vz62ia-ue.a.run.app/

Url producción : https://punana.com