
# Galería de NFT's en Web3

TITULACIÓN: Grado en Ingeniería Informática

AUTOR: Cristóbal José Jiménez Gómez

TUTORIZADO POR:
María Dolores Afonso Suárez

Fecha: 07/2023

# Agradecimientos

Gracias a mi familia por el apoyo diario. Sin duda, sin ellos este trabajo no sería lo mismo.

Gracias a mis amigos, entre los cuales destaco a José Illera, Rubén Santana, Aitor Ventura y Javier Domínguez, quienes siempre estuvieron dispuestos a prestar una oreja y dar consejos, echando una mano cuando la he necesitado.

Agradecer a la tutora por sus consejos, las experiencias otorgadas, por la paciencia, y la comprensión a lo largo del trabajo.

Finalmente, agradecer a todos los compañeros y profesores que me han acompañado en el grado, de los cuales he aprendido miles de experiencias, me han ayudado a crecer, a conocerme a mí mismo, y, sin ellos, no sería la persona que soy hoy.

Muchas gracias.

# Resumen

Desarrollo de una Aplicación Web en el marco de la Web3(blockchain), que gestiona una galería de Non Fungible Tokens (NFTs). Ofrece la gestión de contenidos a distintos tipos de usuarios cuyos niveles de interacción varían desde los usuarios sin registrar, a administradores. En esa escala de accesos los casos de uso establecerán el nivel de interacción de cada perfil. Las funcionalidades implementadas permitirán la gestión a distintos niveles de estos NFTs, desde la visualización hasta la gestión de cada cartera. Este último concepto es el definido para almacenarlos y, en caso de ser necesario comerciar (compraventa o intercambio) con ellos.

**Palabras claves:** Web3, Web3.0, Angular, Firebase, Vercel, Express

# Abstract

Development of a Web Application within the framework of Web3 (blockchain), which manages a gallery of *Non Fungible Tokens* (NFTs). It offers content management to different types of users whose levels of Interaction range from unregistered users to administrators. At this access scale, the use cases will establish the level of interaction of each profile. The functionalities implemented will allow the management at different levels of these NFTs, from the visualization to the management of each briefcase. This last concept is the one defined to store them and, in case it is necessary to trade (purchase or exchange) with them.

**Key Words:** Web3, Web3.0, Angular, Firebase, Vercel, Express

# Índice general

# Índice de figuras

# Índice de tablas

# Capítulo 1

# Introducción

## 1.1. Motivación

La razón principal para el desarrollo de este trabajo ha sido el auge de la *Blockchain* durante estos últimos años. La tecnología "*Blockchain* es un libro mayor compartido e inalterable que facilita el proceso de registro de transacciones y de seguimiento de activos en una red de negocios. Un activo puede ser tangible o intangible. Prácticamente cualquier cosa de valor, puede rastrearse y comercializarse en una red de *blockchain*, reduciendo así el riesgo y los costes para todos los involucrados"[1].

Esta tecnología ofrece la posibilidad de construir aplicaciones web que garanticen la privacidad y seguridad de los datos de un usuario. La Web 3 ofrece transparencia y *trazabilidad* a la hora de la interacción entre usuarios. Además, permite la creación o conversión de negocios ya afianzados en este entorno. Esta propuesta ofrece la posibilidad de interacción entre distintas carteras, permitiendo el acceso a sus direcciones y balances desde una única ubicación, funcionando, por tanto, como un nivel de abstracción o capa superior.

Se debe hacer constar que ya existen páginas afianzadas con el fin de mostrar y realizar compra-venta de NFTs. Este trabajo busca el aprendizaje, tanto del desarrollo de una aplicación web completa, como de las llamadas y conexiones para recoger y mostrar datos de NFTs[2] y criptocarteras[3].

## 1.2. Objetivos

El objetivo principal del proyecto se centra en el desarrollo de un sitio web que utiliza el nuevo concepto de descentralización dentro del marco Web3. El resultado ofrece como funcionalidad principal la búsqueda de NFTs y el almacenamiento de distintas carteras de criptomonedas. Para ello se plantea un conjunto de objetivos generales:

1. Estudiar las características y evolución de la Web 3.0 y su relación con la Web3.

2. Realizar un estudio de las tecnologías y herramientas a emplear.

3. Establecer criterios para su selección (tanto de las tecnologías como de las herramientas).

4. Seguir una metodología de desarrollo, incluyendo las pruebas y documentación del proyecto.

## 1.3. Estructura de la memoria

La memoria está dividida en 10 capítulos diferentes. En ellos se recoge toda la información, desde el inicio del proyecto hasta las conclusiones.

El **primer capítulo** consta de la introducción al proyecto. En este se contextualiza el trabajo, se explica la motivación detrás de la elaboración del proyecto, se dan a conocer los objetivos a cumplir y se explica la estructura del documento.

El **segundo capítulo** lista y justifica las competencias de la titulación abarcadas durante el desarrollo del trabajo mediante las tareas y actividades realizadas.

El **tercer capítulo** lista y justifica los objetivos iniciales del proyecto, y el estado actual del mismo.

El **cuarto capítulo** desarrolla el marco teórico, donde se explican y analizan algunos términos relevantes de este trabajo. Es el punto de partida, donde se realiza una investigación que explica y justifica las decisiones que se toman en el trabajo.

El **quinto capítulo** abarca el estado del arte, que explica la actualidad del desarrollo Web incluyendo el desarrollo dentro de la Web3.0 y la popularidad entre los distintos *frameworks*[1].

El **sexto capítulo** contiene los recursos y las tecnologías utilizadas para el desarrollo del trabajo. En este se definen y justifican.

El **séptimo capítulo** incluye un análisis de los requisitos y el diseño de las distintas estructuras y módulos que conlleva la realización del sitio web.

El **octavo capítulo** explica todo lo relacionado con la metodología llevada a cabo para la estructuración y realización del trabajo.

El **noveno capítulo** aporta los conocimientos básicos adquiridos para el desarrollo del trabajo, así como el desarrollo de la misma organizados por Sprints.

El **décimo capítulo** contiene la evaluación del sitio, por un lado haciendo uso de la herramienta Lighthouse y por otro realizando un análisis estático del software. Este capítulo también incluye la mejora realizada en el último Sprint de acuerdo a los informes de calidad.

El **undécimo capítulo** recoge las conclusiones de este proyecto, tanto personales como objetivas, e incluye propuestas de mejora para un futuro desarrollo.

Finalmente se incluyen los anexos, los cuales también contienen las referencias bibliográficas.

[1] Un *framework* es una herramienta de programación que te permite desarrollar software proporcionando una estructura con componentes integrados que sirven de base para construir proyectos nuevos[4]

# Capítulo 2

# Competencias específicas

Las competencias adquiridas en este proyecto se pueden encontrar en la Tabla 2.1 y a continuación se listan:

Tabla 2.1: **Competencias** cubiertas durante el desarrollo del proyecto.

| Código | Descripción |
|---|---|
| **CI8** | Capacidad para analizar, diseñar, construir y mantener aplicaciones de forma robusta, segura y eficiente, eligiendo el paradigma y los lenguajes de programación más adecuados. |
| **CI13** | Conocimiento y aplicación de las herramientas necesarias para el almacenamiento, procesamiento y acceso a los sistemas de información, incluidos los basados en web. |
| **CI16** | Conocimiento y aplicación de los principios, metodologías y ciclos de vida de la ingeniería del Software |
| **CI17** | Capacidad para diseñar y evaluar interfaces persona computador que garanticen la accesibilidad y usabilidad a los sistemas, servicios y aplicaciones informáticas. |
| **T12** | Capacidad para seleccionar, diseñar, desplegar, integrar, evaluar, construir, gestionar, explotar y mantener las tecnologías de hardware, software y redes, dentro de los parámetros de coste y calidad adecuados. |

**Fuente:** Universidad de Las Palmas de Gran Canaria (2023)

- La competencia CI8 se justifica debido a la necesidad de analizar, diseñar y construir los distintos componentes de la página web, así como las estructuras de los distintos documentos de la base de datos.

- La competencia CI13 se justifica por el uso de llamadas a APIs y la creación de un backend. Incluyendo el despliegue de la aplicación.

- La competencia CI16 se justifica debido a la metodología Scrum aplicada durante el transcurso del proyecto.

- La competencia CI17 se justifica por el uso de las herramientas para el análisis de calidad empleadas para garantizar la accesibilidad y usabilidad de la página web. Además de pedir *feedback* a usuarios potenciales, pidiéndoles información sobre su experiencia en el uso de la aplicación.

- La competencia T12 se justifica debido a que en eso ha consistido este proyecto:

  - **Diseño** de los distintos componentes y páginas de la aplicación.

  - **Despliegue** de la aplicación en sus diferentes ramas para poder ser **evaluadas** por distintos usuarios.

  - **Selección** de las herramientas como el *framework* sobre el que se ha trabajado (Angular), o la plataforma sobre la que se ha hecho el *despliegue*, siempre teniendo en cuenta los recursos, costes y tiempos en aras de desarrollar un producto de calidad en los plazos establecidos.

# Capítulo 3

# Objetivos

El principal objetivo de este Trabajo de Fin de Grado se centra en el desarrollo de un sitio que que hace uso del nuevo concepto de descentralización dentro del marco de la web 3.0 (Ver la evolución de la web en la sección 4.1).

El resultado ofrece como funcionalidad principal la búsqueda de NFTs (Ver la descripción en la sección 4.3) y el almacenamiento de distintas carteras de criptomonedas (Ver la descripción en la sección 4.4).

Para ello se plantea un conjunto de objetivos generales:

1. Estudiar las características y evolución de la Web 3.0 y su relación con la Web3.

    - Como se puede ver en el Capítulo 4, se han estudiado tanto las características, como la evolución de las Web3, sus diversos usos en la actualidad.

2. Realizar un estudio de las tecnologías y herramientas a emplear.

    - Firebase es una de las principales herramientas de Google para el manejo de usuarios.

- Hay una gran cantidad de APIs[1] que nos ayudan a acceder a la Web3, entre ellas:
  - Coinbase: https://docs.cloud.coinbase.com.
  - ThirdWeb: https://thirdweb.com.
  - Moralis: https://moralis.io.
  - Alchemy: https://www.alchemy.com.
- Para el *backend* se revisó el uso de Django y de Express.js:
  - Debido a mi actividad en Prácticas Externas, tuve la oportunidad de probar Django y pude ver que es de fácil uso y comprensión debido al uso de Python, pero eran necesarios demasiados pasos para el despliegue de la aplicación.
  - Express.js aportaba una mayor simplicidad a la hora de crear una API, ya que ejecutar un servidor que escuche un puerto en concreto, es la base del funcionamiento de este *Framework*.
- Para desplegar la página se ha revisado el funcionamiento de Amazon Web Services, GitHub Pages y Vercel:
  - Amazon Web Services es multiusos, lo vi como "matar una mosca a cañonazos", algo demasiado grande para lo que iba a ser el proyecto.
  - A la hora de desplegar un proyecto angular en GitHub Pages me resultó complicado, ya que se tenía que crear la *build* [2] del proyecto, y luego asignar el fichero que se mostraba. Si en algún momento me olvidaba de hacer una build, no se actualizaba aquí, por lo que fue descartada.

[1] Las APIs son mecanismos que permiten a dos componentes de software comunicarse entre sí mediante un conjunto de definiciones y protocolos.[5]

[2] La *build* implica compilar el proyecto, Angular, en este caso, y que nos devuelva el fichero HTML y CSS puros que son los que se leerían y los que veríamos desde cualquier navegador

- Vercel aportaba simplicidad a la hora de desplegar el proyecto, ya que este, al ser Angular lo desplegaba directamente. A la hora de desplegar el backend, hecho en Express.js resultó ser sencillo, fue necesario crear un fichero de configuración escrito gracias a la documentación de Vercel.

3. Establecer criterios para su selección (tanto de las tecnologías como de las herramientas) Estos criterios se desarrollan en el Capítulo 6

   - Debido al tiempo de desarrollo establecido decidí escoger las herramientas según la complejidad de las mismas:

     - Para tener la información de los usuarios, las distintas colecciones a mostrar, etc. Se usó Firebase por ser de Google y de las más completas.

     - Para la obtención de NFTs se hizo uso de la API Alchemy, de fácil uso por estar bien documentada.

     - Para la obtención del balance de las carteras a partir de su dirección pública se hizo uso de la API de Moralis, ya que aún encontrando diversas formas de utilizarla, pero esta ha sido la más efectiva.

     - Para esta última parte se hizo un backend con Express.js, que nos permite hacer uso de la funcionalidad de JavaScript.

     - Para el despliegue de la aplicación se hizo uso de Vercel, por su simplicidad y la disponibilidad de documentación.

4. Seguir una metodología de desarrollo, las pruebas y documentación del proyecto.

# Capítulo 4

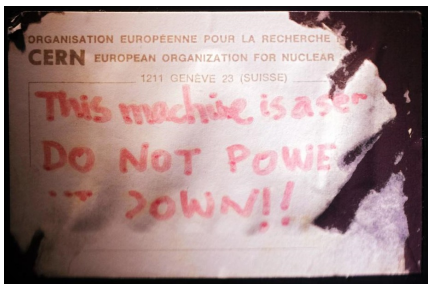
# Marco teórico

En esta parte del informe se darán los conceptos generales para la comprensión del mismo. Se hará un paso por la historia de la Web desde sus inicios hasta como la conocemos ahora y se darán los conocimientos básicos para comprender el funcionamiento de las criptomonedas, los *Tokens* no fungibles y las criptocarteras.

## 4.1. Historia de la Web hasta la Web 3.0

La **World Wide Web** fue creada en 1989 por el científico británico *Tim Berners-Lee* mientras trabajaba en el *Centro Europeo de Investigación Nuclear(CERN)*. La idea inicial era crear una red de información que pudiera ser compartida entre científicos y académicos de todo el mundo. Para evitar un apagado accidental se escribió una nota en tinta roja (Figura 4.1) que ponía: "*This machine is a server. DO NOT POWER IT DOWN!!*"[6] (Esta máquina es un servidor. ¡¡NO LO APAGUEN!!)

Figura 4.1: Pegatina del servidor del CERN con la WWW




**Fuente:** Reddit oficial del CERN [https://www.reddit.com/r/CERN/]

En 1991, *Berners-Lee* creó una serie de tecnologías que permitían la conexión de documentos en un sistema hiper-textual, utilizando el *protocolo* ***HTTP*** y la *codificación* ***HTML***. Esto permitió a los usuarios navegar por la red y acceder a documentos enlazados desde cualquier parte del mundo.

Con el tiempo, la Web se expandió y evolucionó, surgieron nuevas tecnologías como los motores de búsqueda, las redes sociales y las aplicaciones móviles, lo que llevó a la denominada **Web 2.0**.

La Web 2 se caracterizó por una mayor interactividad, el desarrollo de aplicaciones colaborativas y la creación de plataformas para la participación del usuario, como blogs, *wikis*[1] y redes sociales. La Web2 también permitió la creación de empresas en línea y el desarrollo de nuevos modelos de negocio basados en la publicidad y los servicios en línea.[7]

La **Web 3**, también conocida como *Web descentralizada*, es la siguiente evolución de la Web. La Web 3.0 pone el foco en la descentralización de la web, lo que significa que los usuarios tienen mayor control sobre sus datos y pueden interactuar directamente entre sí sin la necesidad de intermediarios centralizados.

La tecnología clave detrás de la Web 3 es la cadena de bloques (blockchain) y otras tecnologías de registro distribuido (DLT)[8], que permiten la creación de aplicaciones descentralizadas (dApps) y contratos inteligentes (*smart contracts*). Esto permite la creación de aplicaciones que no estén sujetas a la censura, la interferencia o la dependencia de un solo proveedor, lo que a su vez ofrece mayor privacidad y seguridad para los usuarios.

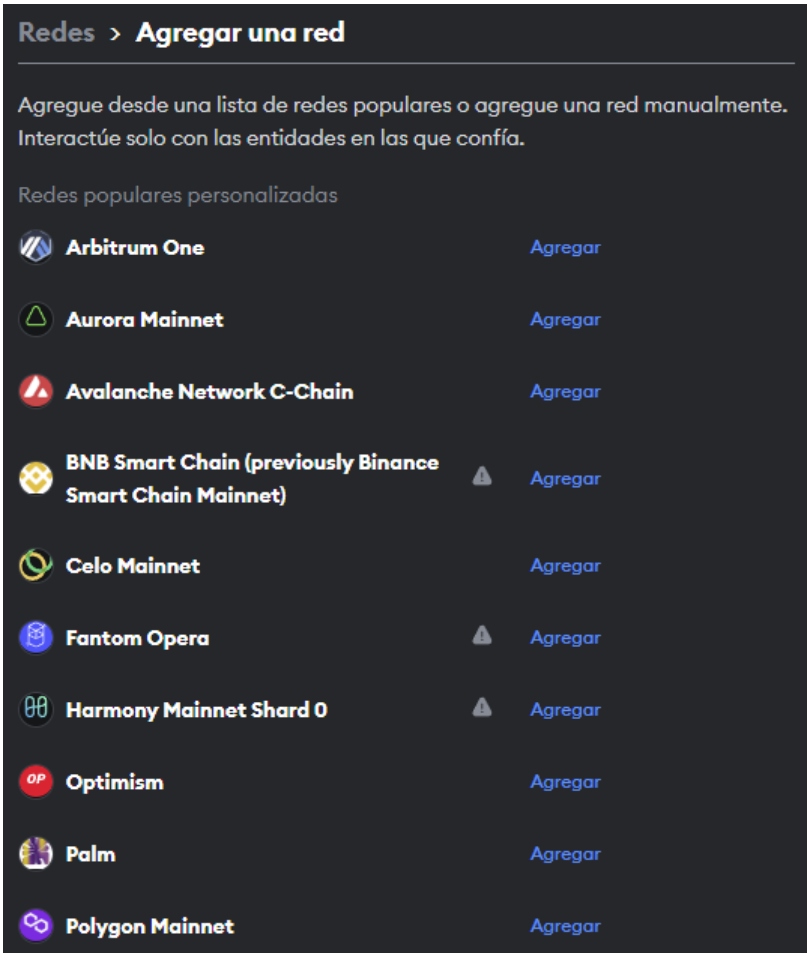
## 4.2. Concepto de Criptomoneda

Las **criptomonedas** son monedas digitales que utilizan tecnología criptográfica para asegurar y verificar las transacciones y para controlar la creación de nuevas unidades de la moneda. Las criptomonedas funcionan en una red descentralizada, lo que significa que no están controladas por ningún gobierno, banco central o entidad financiera.

[1] El término *wiki* hace referencia a un sitio web, cuyas páginas se editan desde el navegador y son los usuarios los que crean, modifican, corrigen o eliminan contenidos que comparten.

Cada criptomoneda tiene su propia **red de *blockchain*** (Figura 4.2), un registro público descentralizado que registra todas las transacciones de la moneda. La *blockchain* es una base de datos distribuida que contiene todas las transacciones realizadas con la moneda desde su creación, y se utiliza para verificar y validar las transacciones y para asegurar que no se puedan crear unidades adicionales de la moneda sin cumplir ciertas condiciones.

Figura 4.2: Las redes más populares según Metamask




**Fuente:** Cuenta personal de Metamask al querer agregar una nueva red a día 04/05/2023

Las **transacciones** en una criptomoneda se realizan de forma *peer-to-peer* (entre pares) y se validan a través de un proceso llamado **minería**. Este es un proceso mediante el cual los usuarios de la red utilizan su poder de cómputo para resolver problemas matemáticos complejos y validar las transacciones en la *blockchain*. Los usuarios que realizan esta tarea reciben recompensas en forma de nuevas unidades de la criptomoneda.

Las criptomonedas se pueden comprar y vender en plataformas de intercambio de criptomonedas, y su valor depende de la oferta y la demanda del mercado.

Ofrecen varias ventajas, como la descentralización, la transparencia, la seguridad y la privacidad, pero también presentan algunos desafíos, como la volatilidad del valor y la falta de regulación.

## 4.3. Concepto de NFT

Los ***tokens* no fungibles** (**NFT**, por sus siglas en inglés) son activos digitales únicos que se utilizan para representar elementos digitales como obras de arte, vídeos, música, juegos, entre otros. A diferencia de las criptomonedas tradicionales, los NFT no son intercambiables y cada uno es único e irrepetible.

Los NFT se basan en la **tecnología *blockchain*** y utilizan contratos inteligentes (*smart contracts*[2]) para garantizar la propiedad y autenticidad del activo digital que representan. Los contratos inteligentes se utilizan para definir las condiciones y términos de la transacción, y garantizan que solo el propietario del NFT tenga derecho a la propiedad del elemento digital que representa.

Los NFT se han vuelto muy populares en el mundo del arte digital, ya que permiten a los artistas vender sus obras de arte como activos únicos [Por ejemplo, la figura 4.3] y garantizar su autenticidad y propiedad. También se están utilizando en otros sectores como los videojuegos, donde se pueden utilizar para representar objetos y personajes únicos [Por ejemplo, la figura 4.4].

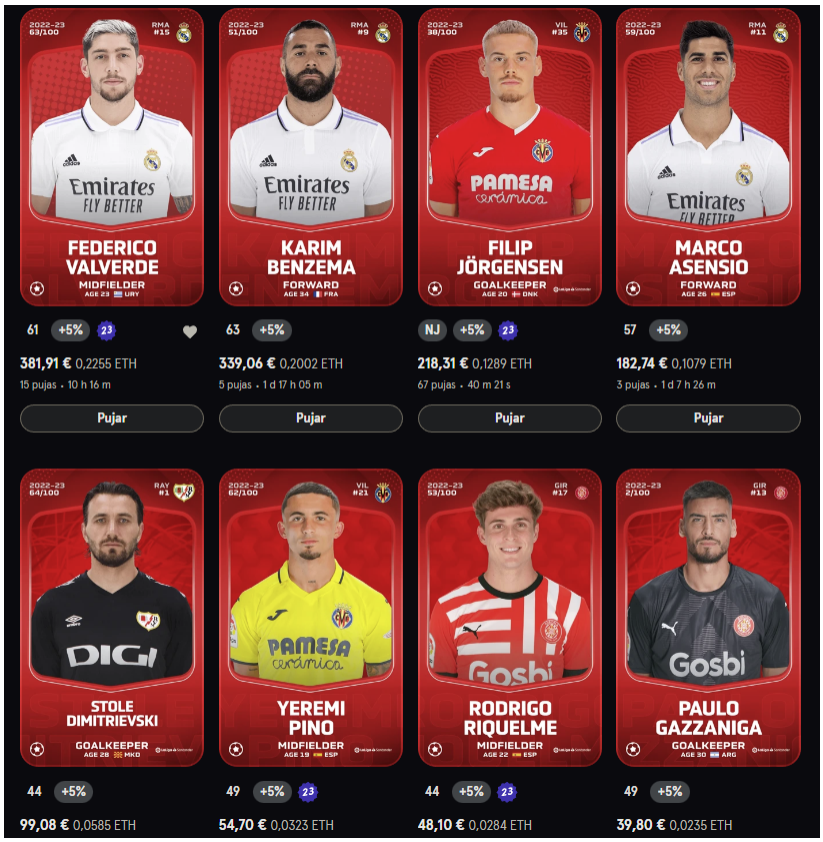
Figura 4.3: Muestra de algunos NFT's de la colección de Moonbirds

**Fuente:** Página de OpenSea [https://opensea.io]- Collección Moonbirds a día 03/04/2023

El valor de un NFT depende de la demanda del mercado y de la percepción del valor del activo digital que representa. Los NFT se pueden vender

[2]Se denomina *Smart Contract* a un programa informático que facilita, asegura, hace cumplir y ejecuta acuerdos registrados entre dos o más partes.[9]

Figura 4.4: Muestra de algunos NFT's de la Sorare




**Fuente:** Página de Sorare [https://sorare.com/]- Sección de mercado con filtros de jugador raro de la Liga Santander a día 03/04/2023

y comprar en plataformas especializadas y se utilizan principalmente como activos de inversión o de colección.

## 4.4. Concepto de Cripto Cartera

Una **criptocartera** es una herramienta que se utiliza para almacenar, enviar y recibir criptomonedas. Es similar a una billetera física, pero en lugar de contener billetes y monedas, una criptocartera contiene claves privadas y públicas que permiten acceder y administrar las criptomonedas.

Cada criptomoneda tiene su propia criptocartera, y hay varios tipos de criptocarteras disponibles, incluyendo carteras de hardware, software y en línea. Las **carteras de hardware** (Figura 4.5) son dispositivos físicos que se conectan a un ordenador y se utilizan para almacenar las claves privadas de manera segura.

Las **carteras de software** se descargan en un ordenador o dispositivo móvil y se utilizan para almacenar las claves privadas en un archivo cifrado.

Las **carteras en línea**(Figura 4.6) se almacenan en la nube y se pueden acceder desde cualquier dispositivo con una conexión a internet.

Figura 4.5: Criptocartera Hardware


**Fuente:** Página oficial de Ledger [https://shop.ledger.com/products/ledger-nano-s-plus]

Figura 4.6: Criptocartera Online

coinbase

**Fuente:** Logo oficial de Coinbase [https://www.coinbase.com/es/]

Las criptocarteras son importantes porque las criptomonedas no se almacenan en un lugar centralizado, como un banco, sino que se almacenan en la *blockchain* de la criptomoneda. Por lo tanto, es necesario utilizar una criptocartera para acceder y administrar las criptomonedas.

Las criptocarteras también permiten **enviar y recibir criptomonedas** de forma segura. Para enviar criptomonedas, se necesita la dirección pública de la criptocartera del destinatario. Para recibir criptomonedas, se proporciona la dirección pública de la criptocartera al remitente.

# Capítulo 5

# Estado del Arte

## 5.1. Industria de la Web 3.0

La industria de la web3.0 se compone de una variedad de proyectos y empresas que están trabajando en diferentes aspectos de esta tecnología. Algunos de estos proyectos incluyen:

### 5.1.1. Protocolos *Blockchain*

Son sistemas que permiten la creación y operación de redes descentralizadas y seguras. Estos protocolos permiten la creación de aplicaciones descentralizadas, contratos inteligentes y tokens criptográficos. Los protocolos *blockchain* están en constante evolución y crecimiento, y están impulsados por la comunidad y la colaboración. Algunos de los protocolos blockchain más populares son Bitcoin, Ethereum, Binance Smart Chain, Polkadot y Solana.

### 5.1.2. Criptomonedas

Son monedas digitales que se basan en la tecnología *blockchain* y se utilizan para realizar transacciones en línea de forma segura. Dentro de cada protocolo podemos encontrar distintas Criptomonedas, como pueden ser Bitcoin, Etheremun, Dólares Theter, Solana, Shiba Inu, etc.

### 5.1.3. Aplicaciones descentralizadas (dApps)

Este apartado se puede ver de forma más extendida dentro de la sección 5.2.

Son aplicaciones web que se ejecutan en la blockchain y utilizan contra-

tos inteligentes para permitir transacciones seguras y sin intermediarios. Las dApps pueden ser utilizadas para una amplia variedad de aplicaciones, desde finanzas descentralizadas (DeFi) hasta juegos y redes sociales. Algunos ejemplos de dApps populares incluyen:

- **Uniswap**: es una plataforma de intercambio descentralizada (DEX) que permite a los usuarios intercambiar criptomonedas sin la necesidad de intermediarios.

- **Decentraland**: es un mundo virtual descentralizado donde los usuarios pueden comprar, vender y construir propiedades virtuales utilizando criptomonedas.

- **Brave**: es un navegador web descentralizado que permite a los usuarios controlar su privacidad y monetizar su atención en línea.

- **OpenSea**: es un mercado de intercambio descentralizado para tokens criptográficos no fungibles (NFT) que permite a los usuarios comprar y vender obras de arte digitales, coleccionables y otros activos digitales únicos.

La industria de la web3.0 está en constante evolución y crecimiento, y está impulsada por la comunidad y la colaboración. Los proyectos y empresas en esta industria están trabajando juntos para construir una internet más segura, justa y descentralizada.

## 5.2. Páginas principales de la Web 3 en sus distintos ámbitos

A continuación, veremos las **dApps** más conocidas a día de hoy en distintos ámbitos [[10]]: Redes sociales, servicios de intercambio monetario, servicios de almacenamiento, servicios de *streaming* de vídeo y música

### 5.2.1. Redes Sociales Web3

**Steemit**[Figura 5.1][https://steemit.com]: Se ejecuta completamente en la blockchain Steem. Se describe mejor como una plataforma de recompensa descentralizada que ayuda a los contribuyentes a monetizar su contenido. Es una alternativa a Reddit[https://reddit.com].

Algunos beneficios que tendría esta página con respecto a otras en su ámbi-

Figura 5.1: Logo de Steemit


**Fuente:** Steemit[https://steemit.com]

to, que no se encuentran dentro de la web 3 serían:

- No hay una autoridad central que capture datos y los use.
- Hace poderosos a los usuarios al recompensarlos con algún tipo de activo.
- Mejora en las redes sociales de la Web 2.0 en casi todos los sentidos.
- Protege la privacidad de los usuarios.
- Los usuarios deciden qué quieren compartir y cuándo.
- Las grandes corporaciones y organizaciones pierden influencia.

### 5.2.2. Servicios de intercambio descentralizados

Cualquiera de estas aplicaciones podrían asociarse a lo que podría ser un banco, donde podemos introducir dinero en distintas divisas, realizar transacciones, y el precio de las distintas divisas.

**IDEX**: Es un servicio de intercambio descentralizado popular para el comercio de *tokens*. Proporciona una buena interfaz para los usuarios, y, cualquier persona con una cartera de ethereum puede comenzar a operar en la plataforma. Para hacer el mejor uso de IDEX o de cualquier intercambio descentralizado basado en ethereum, una de las mejores opciones es MetaMask.

**Coinbase** [Figura 5.2]: Es una plataforma de intercambio de criptomonedas en línea que permite a los usuarios comprar, vender y almacenar una variedad de criptomonedas, como Bitcoin, Ethereum, o Litecoin, entre otras. Fue fundada en 2012 y se ha convertido en una de las plataformas más populares y utilizadas en el mundo de las criptomonedas.

Además de la plataforma de intercambio, Coinbase también ofrece una

Figura 5.2: Logo de Coinbase


**Fuente:** Coinbase[https://www.coinbase.com]

billetera digital integrada para almacenar las criptomonedas y una API para desarrolladores que deseen integrar las funcionalidades de la plataforma en sus propias aplicaciones.

Una de las principales características de Coinbase es su interfaz de usuario intuitiva y amigable, lo que la hace atractiva tanto para principiantes como para usuarios avanzados. También se ha destacado por su enfoque en la seguridad y la protección de los activos de los usuarios, utilizando medidas de seguridad avanzadas como la autenticación de dos factores y la custodia de criptomonedas.

Hacer intercambios de criptomonedas de forma descentralizada tiene ciertos beneficios:

- Transacciones más baratas.
- Transacciones más rápidas.
- Difícil de piratear debido a la naturaleza descentralizada.
- Funciona bien con carteras de hardware.
- Los usuarios controlan sus propios fondos.

Sin embargo, los intercambios descentralizados no están libres de aspectos negativos. Pueden ser difíciles de usar y comprender para los usuarios. Además, la generación actual de intercambios descentralizados también sufre la falta de características y funcionalidades. Sin embargo, con el tiempo, veremos la red descentralizada más avanzada a la par con las contra partes centralizadas.

### 5.2.3. Servicios de Almacenamiento descentralizados

Las siguientes herramientas podrían ser comparadas en la Web2 con Dropbox, Google Drive o Microsoft OneDrive.

**Storj**[https://www.storj.io]: Storj es una de las principales soluciones de almacenamiento descentralizado. También es uno de los más antiguos. Con Storj, cualquiera puede almacenar datos. También es de código abierto y fácil de usar. Cualquiera puede comenzar a utilizarlo con solo 1-Clic. El modelo de pago se crea alrededor de los usuarios, ya que pueden pagar según lo utilicen. El *token* Storj se utiliza para alimentar la plataforma Storj.

**Sia**[Figura 5.3][https://sia.tech]: Proporciona almacenamiento de forma descentralizada y también se considera la mayor competencia de Storj. Sia divide el archivo en treinta segmentos y luego lo distribuye. También encripta el archivo mientras se transfiere.

Figura 5.3: Logo de Sia


**Fuente:** Steemit

**IPFS (InterPlanetary File System)**[Figura 5.4][https://ipfs.tech]: Es una tecnología de almacenamiento distribuido que utiliza la red blockchain para almacenar archivos de manera descentralizada. IPFS permite a los usuarios acceder a los archivos de forma más rápida y segura, ya que los archivos se almacenan en múltiples nodos en la red.

Figura 5.4: Logo de IPFS


**Fuente:** wikipedia [https://es.wikipedia.org/wiki/Sistema_de_archivos_interplanetario]

Beneficios de las soluciones de almacenamiento descentralizado Funciona bien en diferentes plataformas o incluso en soluciones de blockchain. Protege los datos que se transfieren con cifrado fuerte. Ninguna entidad centralizada, significa que nadie puede usar los datos. Es barato y funciona bien con tecnologías de próxima generación como **IoT (Internet de las cosas)**.

### 5.2.4. *Streaming* de Vídeo y Música

Aquí podríamos comparar con aplicaciones como Twitch[https://www.twitch.tv], YouTube[https://www.youtube.com] o U-beat[https://ubeat.tv], en el apartado de *streaming* de vídeo.

Por otra parte, también podríamos comparar aplicaciones web como YouTube Music[https://music.youtube.com], Spotify[https://open.spotify.com/], o SoundCloud[https://soundcloud.com], para el *streaming* de música.

**LivePeer**[https://livepeer.org/es]: Proporciona un servicio de transmisión, es de código abierto y apunta a construir una pila de *streaming* para la Web 3.0.

**Audius**[Figura 5.5][https://audius.co]: Es una plataforma de música descentralizada que permite a los artistas compartir su música y monetizarla. Los usuarios pueden escuchar música de forma gratuita. Esta aplicación es la que pretende reemplazar a Spotify dentro del mundo descentralizado.

Figura 5.5: Logo de Audius


**Fuente:** Ecosystem.ipfs [https://ecosystem.ipfs.tech/project/audius/]

Los creadores de contenido pueden trabajar en un entorno transparente. Todos tienen las mismas oportunidades de promover su trabajo. Los problemas de derechos de autor serán insignificantes gracias a los contratos inteligentes y no hay autoridad central, por lo tanto no será una política absurda para los *streamers* y los creadores de contenido.

### 5.2.5. Navegadores descentralizados

Teniendo en cuenta los navegadores más utilizados en estos últimos 5 años [Figura 5.7], podemos ver que las comparaciones son abrumadoras, teniendo Google Chrome casi un 63 % del mercado, siguiéndole Safari, Mozilla Firefox y Samsung Internet. Por lo que cualquiera de los siguientes navegadores no pueden competir con este, aunque tienen sus beneficios al estar implementando la tecnología blockchain.

**Brave Browser**[Figura 5.6][https://brave.com/es/]: Brave tiene que ver con la privacidad donde los usuarios no son el producto. El navegador viene preinstalado con el bloqueador de anuncios. También permitirá a los usuarios vender sus datos a cambio de la criptomoneda, BAT, la cual puede ser usada para diversos propósitos, como pueden ser hacer *Staking*[1][12], ser cambiadas por otra Criptomoneda, o venderla para conseguir divisa.

- Beneficios de los navegadores descentralizados:

- Los usuarios pueden navegar de forma privada por internet.

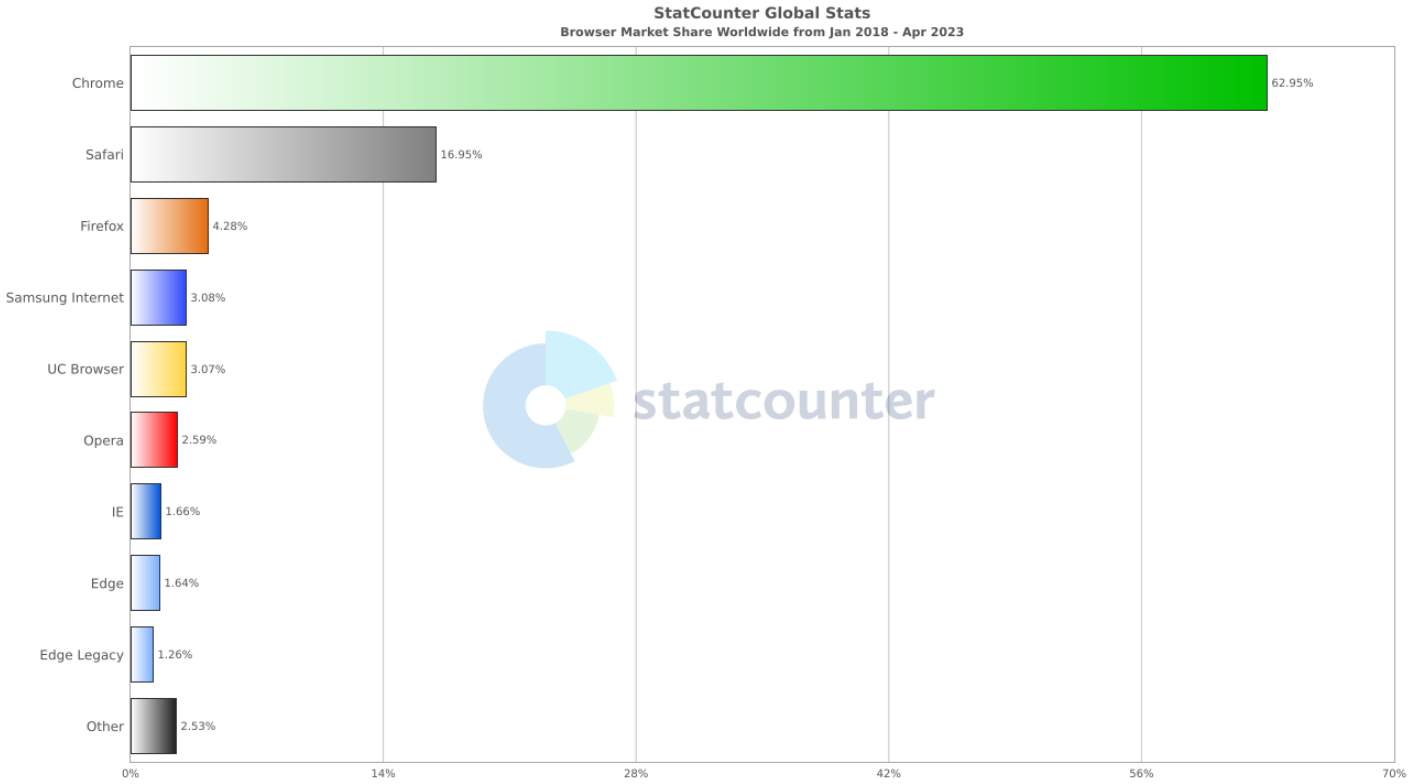
[1]Consiste en mantener fondos en un monedero de criptomonedas, para respaldar la seguridad y las operaciones de una red blockchain. En pocas palabras, staking es el acto de de dejar bloqueadas en depósito criptomonedas para recibir recompensas.[11]

Figura 5.6: logos de Brave

**Fuente:** logos.Wine

- Ninguna o menor cantidad de lagunas de seguridad.
- Los usuarios pueden vender sus datos a una organización y recibir pagos.
- Rápido y seguro.

Figura 5.7: Navegadores más usados entre 2018 y 2023




**Fuente:** StatCounter
[https://gs.statcounter.com/browser-market-share#monthly-201801-202304-bar]

## 5.3. Mercado

La web3 se está desarrollando rápidamente y se espera que cambie la forma en que interactuamos en línea. Con la tecnología blockchain y los contratos inteligentes, la web3 permitirá aplicaciones descentralizadas, transacciones financieras sin intermediarios y la propiedad verdadera de los datos. Algunos de los protocolos blockchain más populares para la web3 incluyen Ethereum, Polkadot y Solana.

Las criptomonedas continúan siendo una fuerza importante en el mercado financiero global. Bitcoin sigue siendo la criptomoneda más grande y popular, pero hay muchas otras criptomonedas importantes, como Ethereum, Binance Coin y Cardano. La capitalización total del mercado de criptomonedas ha aumentado significativamente en los últimos años, llegando a más de 2 billones de euros[Figura 5.8] en abril de 2021.

Figura 5.8: Capitalización total del mercado de criptomonedas entre abril del 2017 y mayo de 2023


**Fuente:** CoinMarketCap [https://coinmarketcap.com/es/charts/]

Los NFTs se han convertido en un nuevo mercado emocionante en la web3. Las ventas de NFTs han aumentado significativamente en desde 2020, con algunos NFTs vendiéndose por decenas de miles de dólares.

La regulación de las criptomonedas y los NFTs sigue siendo un tema importante. A medida que estos mercados crecen, los reguladores de todo el mundo están considerando cómo regularlos para proteger a los inversores y prevenir el fraude. En algunos países, como China, se han tomado medidas más drásticas para prohibir las criptomonedas y las transacciones de NFTs.

# Capítulo 6

# Recursos y tecnologías

En este capítulo se tratan los recursos y las tecnologías empleadas para el diseño y desarrollo de la aplicación Web.

## 6.1. Angular

Angular[https://angular.io] es un *framework* de desarrollo web para crear aplicaciones de una sola página (SPA - *Single Page Application*) y aplicaciones web dinámicas. Fue desarrollado por Google y se basa en el lenguaje de programación TypeScript.

Figura 6.1: Logo de Angular

**Fuente:** Página oficial de Angular [https://angular.io]

Angular proporciona una estructura sólida y coherente para el desarrollo de aplicaciones web complejas, lo que permite a los desarrolladores construir aplicaciones de alta calidad y escalables de manera más rápida y eficiente.

Algunas de las características clave de Angular incluyen:

- **Inyección de dependencias**: Permite que los componentes se comuniquen entre sí de manera eficiente.

- **Enlazado de datos bidireccional**: Facilita la actualización automática de la interfaz de usuario en tiempo real.

- **Directivas**: Permite crear componentes personalizados y hacer uso de componentes predefinidos.

- **Servicios**: Permite compartir datos y lógica de aplicación entre los componentes.

Angular también viene con una amplia variedad de herramientas y bibliotecas adicionales, como Angular Material, que ayudan a los desarrolladores a crear aplicaciones web robustas y de alta calidad.

Figura 6.2: Logo de Angular Material


**Fuente:** Página oficial de Angular Material [https://material.angular.io]

Se ha hecho uso de un algunos componentes de Angular Material, como son:

Material Table [https://v5.material.angular.io/components/table/overview], junto con Paginator:

El **componente mat-table** proporciona una tabla de datos de estilo Material Design que se puede utilizar para mostrar filas de datos.

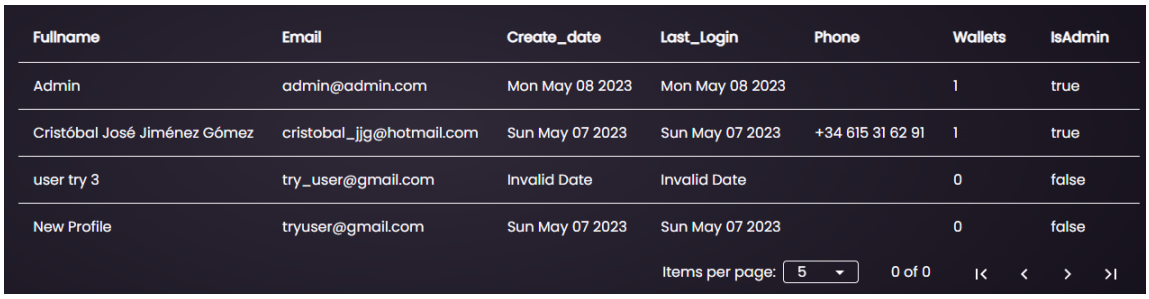
El **componente mat-paginator** Proporciona navegación para la información paginada, normalmente utilizada con una tabla.

## 6.2. Express.JS

Como indica su página web, ExpressJS es una opción rápida, sin opiniones y minimalista para Node.js.

Es un marco de aplicación web para Node.js que simplifica el proceso de creación de aplicaciones web y APIs (interfaces de programación de aplicaciones). Ofrece una capa delgada sobre Node.js que proporciona una variedad de características y herramientas para crear aplicaciones web y APIs de manera rápida y eficiente.

Figura 6.3: Muestra de tabla y paginator en Trabajo propio



| Fullname | Email | Create_date | Last_Login | Phone | Wallets | IsAdmin |
|---|---|---|---|---|---|---|
| Admin | admin@admin.com | Mon May 08 2023 | Mon May 08 2023 | | 1 | true |
| Cristóbal José Jiménez Gómez | cristobal_jjg@hotmail.com | Sun May 07 2023 | Sun May 07 2023 | +34 615 31 62 91 | 1 | true |
| user try 3 | try_user@gmail.com | Invalid Date | Invalid Date | | 0 | false |
| New Profile | tryuser@gmail.com | Sun May 07 2023 | Sun May 07 2023 | | 0 | false |

Items per page: 5 0 of 0

**Fuente:** Trabajo Propio en la zona de administrador [https://tft-galeria-nft-web3.vercel.app/admin/users]

Figura 6.4: Logo de ExpressJS


**Fuente:** Página oficial de ExpressJS [http://expressjs.com]

ExpressJS es un marco minimalista que proporciona una estructura básica para construir aplicaciones web, pero permite a los desarrolladores agregar funcionalidad adicional utilizando *middleware* y complementos. ExpressJS también proporciona un *enrutamiento* simple y flexible, lo que facilita la creación de rutas para diferentes páginas y recursos.

Además, ExpressJS es altamente personalizable y se integra fácilmente con otros módulos y herramientas de Node.js, lo que lo convierte en una opción popular para construir aplicaciones web y APIs.

## 6.3. APIs

### 6.3.1. Firebase

Firebase es una plataforma de desarrollo de aplicaciones móviles y web que ofrece una variedad de herramientas y servicios en la nube para ayudar a los desarrolladores a crear, mejorar y escalar aplicaciones.

Figura 6.5: Logo de Firebase

**Fuente:** Página oficial de Firebase [https://firebase.google.com/?hl=es]

Firebase fue adquirido por Google en 2014 y desde entonces ha evolucionado para ofrecer una amplia gama de servicios, incluyendo:

- Autenticación de usuarios[Capítulo 6.3.1.1]: permite a los desarrolladores agregar fácilmente funciones de registro, inicio de sesión y autenticación a sus aplicaciones.

- Almacenamiento en la nube[Capítulo 6.3.1.2]: proporciona almacenamiento en la nube para archivos y datos de aplicaciones.

- Base de datos en tiempo real: una base de datos en tiempo real en la nube que permite a los desarrolladores crear aplicaciones en tiempo real con actualizaciones automáticas y en tiempo real.

- Alojamiento de aplicaciones web: permite a los desarrolladores alojar sus aplicaciones web en los servidores de Firebase.

- Notificaciones *push*: permite enviar notificaciones *push* a los usuarios de la aplicación.

- Analítica de aplicaciones: proporciona análisis de usuarios y datos de uso de la aplicación.

Para este proyecto no se ha hecho uso de todas las utilidades, solamente de:

#### 6.3.1.1. Autenticación de usuarios

Se ha hecho uso de esta utilidad debido a que nos proporciona una forma sencilla de registrar usuarios, hacer que estos inicien sesión, y que puedan cerrar sesión de una forma simple. Además, y como veremos, nos permite hacer uso de la autenticación de Google, lo que nos permite tener una mayor seguridad en el inicio de sesión de los usuarios.

#### 6.3.1.2. Almacenamiento en la nube

Se ha hecho uso de esta utilidad debido a que nos proporciona una base de datos no relacional en la nube, lo que nos permite tener una mayor seguridad en el almacenamiento de los datos de los usuarios, y de la aplicación en general.

En este caso, guardamos 3 tipos de datos:

- Los **usuarios** registrados, con la información del correo, si es administrador, el nombre, el apellido, su número de móvil, su nombre de usuario,

Figura 6.6: Tabla de autenticación proporcionada por Firebase

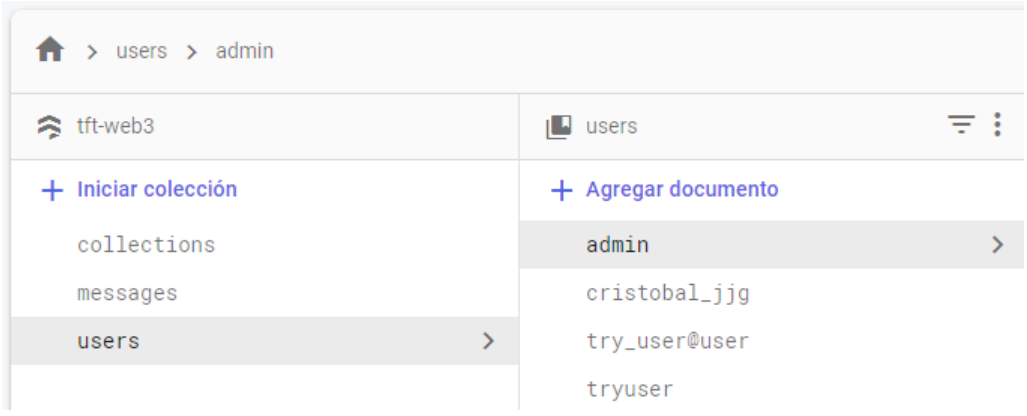

**Fuente:** Consola de Firebase, apartado de Autenticación

que está compuesto por el inicio del correo y su primer nombre, y las carteras que este ha agregado.

- Las **colecciones** que se mostrarán de NFTs, que contienen las direcciones de las colecciones para poder acceder a las mismas, la *url* de la colección, el nombre real y la foto de la marca.

- También se ha hecho un apartado para **mensajes**, el cual contiene mensajes que han dejado los usuarios con intenciones de mejora. Este documento contiene el correo del usuario que lo ha mandado, el mensaje, y un asunto.

Figura 6.7: Conjunto de colecciones y documentos proporcionada por Firebase




**Fuente:** Consola de Firebase, apartado de Firestore Database

### 6.3.2. Alchemy

Alchemy es una plataforma de desarrollo con soporte *multicadena* y con alcance global, pensada en facilitar el desarrollo de aplicaciones descentralizadas (dApps). Su principal objetivo es ofrecer todo lo que los desarrolladores necesitan para construir y hacer realidad la Web3.[[13]]

Figura 6.8: Logo de Alchemy


**Fuente:** Página oficial de Alchemy [https://www.alchemy.com]

Su relevancia en el sector le ha válido ser reconocida como el "AWS de la Web3", manejando más de 10 millones de usuarios, movilizando más de 100 mil millones de dólares en activos digitales y con un concurrencia de más de 100 mil millones de peticiones, lo que le ha llevado a tener una valoración de mercado de más 10 mil millones de dólares.

- **Infraestructura escalable:** Está diseñada para manejar una alta carga de solicitudes y proporcionar una infraestructura confiable y escalable para los desarrolladores.

- **API optimizada:** Ofrece una API eficiente y de alto rendimiento que permite a los desarrolladores acceder y utilizar datos blockchain sin tener que gestionar la infraestructura subyacente.

- **Análisis y monitorización de datos:** Proporciona herramientas de análisis y monitorización que permiten a los desarrolladores obtener información detallada sobre el estado de la blockchain, el rendimiento de las transacciones y otros datos relevantes.

- **Integraciones fáciles:** Ofrece integraciones con diversas bibliotecas y marcos de desarrollo populares, lo que facilita a los desarrolladores la incorporación de la funcionalidad de blockchain en sus aplicaciones.

- **Seguridad y privacidad:** Se enfoca en la seguridad y la privacidad de los datos y utiliza medidas de seguridad robustas para proteger las claves privadas y los datos confidenciales de los usuarios.

- **Soporte y documentación:** Cuenta con un equipo de soporte dedicado y una documentación completa que ayuda a los desarrolladores a utilizar eficazmente sus servicios y solucionar problemas.

### 6.3.3. Moralis

Moralis [https://moralis.io] es una plataforma de infraestructura *blockchain* de la web3 que proporciona herramientas y servicios para desarrolladores que construyen aplicaciones descentralizadas (dApps) en la cadena de bloques Ethereum. Ofrece una amplia gama de herramientas, incluyendo APIs y nodos de red, para que los desarrolladores puedan interactuar con Ethereum de manera eficiente y escalable.

Figura 6.9: Logo de Moralis


**Fuente:** Página oficial de Moralis [https://moralis.io]

Moralis aporta a los usuarios una gran cantidad de APIs dentro de la web 3 para poder llevar sus aplicaciones al mundo descentralizado:

- **API de Datos del Mercado**: Esta API proporciona información en tiempo real sobre el mercado de criptomonedas y tokens. Puedes obtener datos como precios, volúmenes de negociación, capitalización de mercado y otros indicadores relevantes para diferentes criptomonedas y tokens.

- **API de NFTs:** Esta API te permite acceder a información y funcionalidades relacionadas con los NFTs (Tokens No Fungibles). Puedes obtener detalles sobre NFTs específicos, como su nombre, descripción, propietario, imagen y otros metadatos asociados. Además, puedes realizar operaciones como la creación, actualización y transferencia de NFTs.

- **API de Tokens:** Con esta API, puedes obtener información sobre tokens específicos, como su nombre, símbolo, dirección de contrato y decimales. También puedes acceder a detalles sobre las transacciones y los saldos de tokens de una dirección de billetera específica.

- **API de Balances:** Esta API te permite obtener información sobre los saldos de criptomonedas y tokens de una dirección de billetera específica. Puedes obtener el saldo total y detallado de cada criptomoneda o token en una billetera, lo que resulta útil para mostrar balances y realizar cálculos relacionados.

- **API de Transacciones:** Esta API permite acceder a información sobre transacciones realizadas en la cadena de bloques. Puedes obtener detalles sobre transacciones específicas, como las direcciones involucradas, los montos transferidos, las tarifas pagadas y otros datos relacionados.

- **API de Bloques:** Con esta API, puedes obtener información sobre bloques individuales en la cadena de bloques. Puedes acceder a detalles como el número de bloque, las transacciones incluidas en el bloque, las direcciones de origen y destino, y otros datos relevantes.

- **API de Eventos:** Esta API te permite suscribirte a eventos específicos en la cadena de bloques, como transacciones, cambios de saldo o eventos personalizados. Puedes recibir notificaciones en tiempo real cuando ocurran estos eventos, lo que resulta útil para mantener actualizada tu aplicación y responder a cambios en la cadena de bloques.

- **API de DeFi:** Esta API ofrece información y funcionalidades relacionadas con las finanzas descentralizadas (DeFi). Puedes acceder a detalles sobre contratos inteligentes y protocolos DeFi, obtener datos sobre préstamos, intercambios, rendimientos y otras actividades DeFi.

- **API de IPFS:** Con esta API, puedes interactuar con el *InterPlanetary File System* (IPFS), un sistema descentralizado de almacenamiento de archivos. Puedes subir archivos a IPFS, obtener su dirección de almacenamiento y acceder a archivos almacenados previamente.

- **API de *Utils*:** Esta API proporciona varias utilidades y funciones auxiliares que pueden ser útiles en el desarrollo de aplicaciones descentralizadas. Esto incluye funciones para el cálculo de *hashes*, la conversión de formatos de datos, la validación de direcciones y otras operaciones comunes.

Entre estas empresas, podemos encontrar algunas antes nombradas, como son Metamask[https://metamask.io] o Polygon [https://polygon.technology]

## 6.4. Trello

Trello es una herramienta de gestión de proyectos basada en la nube, que permite a los usuarios organizar tareas y proyectos en tableros visuales y colaborativos. Fue lanzado en 2011 y se ha vuelto muy popular entre equipos de trabajo de diversas industrias y tamaños.

Figura 6.10: Logo de Trello

**Fuente:** Iconduck [https://iconduck.com/icons/95002/trello]

Trello utiliza un sistema de tableros que representan los diferentes proyectos o áreas de trabajo, y dentro de cada tablero, los usuarios pueden crear listas de tareas y tarjetas que representan cada tarea o actividad. Estas tarjetas pueden contener información como descripciones, listas de verificación, etiquetas, fechas de vencimiento, comentarios y archivos adjuntos.

Además, Trello permite la colaboración entre equipos de trabajo, ya que los miembros pueden comentar en las tarjetas, asignar tareas a otros miembros, establecer fechas de vencimiento y recibir notificaciones de actualizaciones. Trello también se integra con otras herramientas populares de productividad, como Slack, Google Drive y Jira, lo que lo hace aún más útil para equipos de trabajo que utilizan diferentes herramientas.

## 6.5. Figma

Figma es una herramienta de diseño gráfico en línea que se utiliza para crear interfaces de usuario, diseños de sitios web, aplicaciones móviles y otros proyectos digitales. Fue lanzado en 2016 y se ha vuelto muy popular en la industria del diseño debido a su capacidad para trabajar en tiempo real y permitir la colaboración en tiempo real entre los miembros del equipo.

Figura 6.11: Logo de Figma

**Fuente:** Iconduck [https://iconduck.com/icons/39864/figma]

Figma nos permite crear diseños con herramientas como vectores, formas, capas, texto y efectos, entre otras. También tiene características como *prototipado* y animación, lo que lo hace ideal para diseñar interfaces interactivas. Además, todos los archivos se guardan en la nube, por lo que se peuden acceder a los mismos desde cualquier lugar con conexión a Internet.

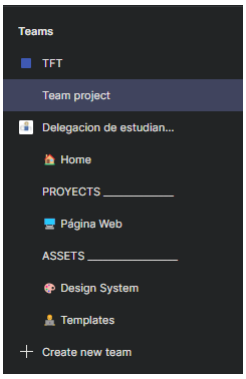
### 6.5.1. Componentes de Figma

El componente más grande que tiene Figma son los **Equipos**. En ellos se pueden añadir diversos **proyectos**[Figura 6.12], y dentro de estos es donde están los ficheros, **Ficheros de Diseño** y **Ficheros de FigJam** [Figura 6.13]

#### 6.5.1.1. Ficheros de Diseño

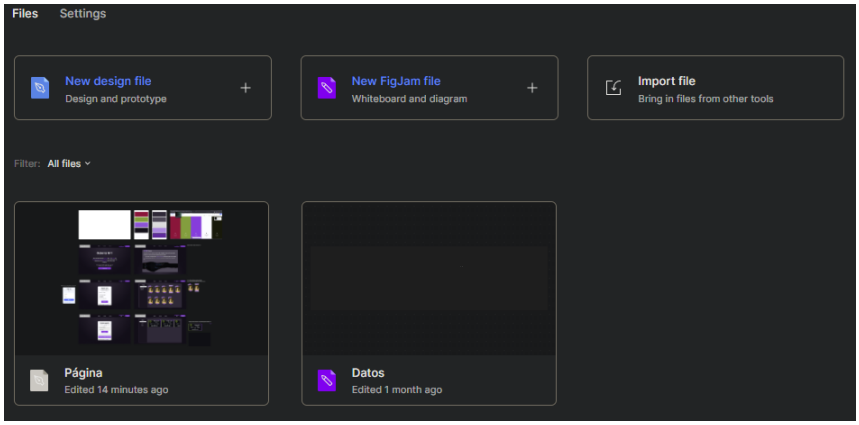
Los Ficheros de Diseño nos permiten:

Figura 6.12: Equipos y proyectos dentro de Figma




**Fuente:** Página personal de Figma

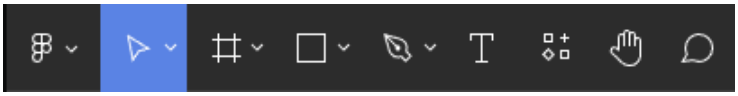
Figura 6.13: Ficheros dentro del Proyecto de TFG




**Fuente:** Página personal de Figma

**Creación de interfaces de usuario**: Permite crear y diseñar interfaces de usuario para aplicaciones de cualquier tipo. Se puede utilizar las herramientas de diseño [Figura 6.14] de Figma, como formas, texto, imágenes y estilos, para crear y personalizar los elementos visuales de la interfaz.

Figura 6.14: Herramientas de Figma



**Fuente:** Proyecto personal de Figma de TFG https://www.figma.com/file/DBYpdtBDwonogrz2P2dpAo/P%C3%A1gina?type=design&node-id=0-1&t=3SC5aJfljVJd5JWF-0

***Prototipado* de interacciones**: Permite crear prototipos interactivos para simular la experiencia de uso de una interfaz.

**Colaboración en tiempo real**: Un fichero de diseño en Figma se puede compartir y colaborar en tiempo real con otros miembros del equipo.

**Organización de elementos**: Figma permite organizar los elementos de diseño en capas, grupos y marcos.

**Bibliotecas y componentes**: Facilita la creación de bibliotecas de componentes y estilos reutilizables. Se pueden crear componentes personalizados y estilos predefinidos que se pueden utilizar en diferentes proyectos y mantenerlos actualizados en todo el fichero de diseño.

#### 6.5.1.2. Ficheros FigJam

A diferencia de los ficheros de diseño estándar de Figma, los ficheros de FigJam están diseñados específicamente para la colaboración en tiempo real y la comunicación visual. Algunas de las funciones y usos principales de los ficheros de FigJam son:

**Diagramas y flujos de trabajo**: Los ficheros de FigJam permiten crear diagramas y flujos de trabajo visualmente. Se pueden utilizar formas, líneas y otros elementos gráficos [Figura 6.15] para representar procesos, organigramas, mapas de ruta y otros tipos de diagramas.

**Lluvia de ideas**: Es una herramienta ideal para realizar sesiones de lluvia de ideas en equipo ya que se pueden crear notas adhesivas, dibujar bocetos, escribir ideas y organizarlas de forma visual en un espacio de trabajo compartido.

Figura 6.15: Herramientas de FigJam

**Fuente:** Proyecto personal de Figma de TFG https://www.figma.com/file/8lLnZjk4bICcc18adGK7Gp/Datos?type=whiteboard&t=WhvBR5RgGiQSDCDI-0

## 6.6. IDE - Visual Studio Code

Visual Studio Code (también conocido como VS Code) [https://code.visualstudio.com] es un editor de código fuente desarrollado por Microsoft, que es compatible con múltiples lenguajes de programación. Está disponible de manera gratuita para Windows, Linux y macOS, y es muy popular por su facilidad de uso, flexibilidad y personalización.

VS Code incluye características útiles como resaltado de sintaxis, autocompletado de código, depuración de código en vivo, integración con control de versiones y herramientas de construcción y despliegue de aplicaciones.

Figura 6.16: Logo de Visual Studio Code

**Fuente:** Iconduck [https://iconduck.com/icons/102490/file-type-vscode]

Además, los usuarios pueden personalizar el editor con extensiones y temas para satisfacer sus necesidades específicas de programación.

Se han utilizado algunas extensiones básicas para el desarrollo web como son:
Angular Language Service, Auto Close Tag, Auto Rename Tag, Console Ninja, IntelliCode, GitHub Copilot Nightly, Prettier - Code formatter, SCSS Formatter, VSCode-pdf

## 6.7. Vercel

Vercel es una plataforma en la nube que se utiliza para **implementar y alojar** aplicaciones web estáticas y de servidor sin problemas. Proporciona un entorno de implementación y alojamiento simplificado para proyectos de desarrollo web.

Destaca por su capacidad para implementar y escalar aplicaciones rápidamente. Admite diferentes tecnologías web, como React, Vue.js, Next.js, Angular y otros frameworks y bibliotecas populares.

Figura 6.17: Logo de Vercel


**Fuente:** Página oficial de Vercel [https://vercel.com]

Una de las características más destacadas es su capacidad de despliegue automático y continuo. Permite configurar flujos de trabajo de implementación que se activan automáticamente cuando se realizan cambios en el repositorio de código fuente. Esto agiliza el proceso de implementación y permite una entrega continua de nuevas características y actualizaciones a medida que se desarrolla el proyecto.

## 6.8. Validaciones

En este último curso se ha profundizado intensamente sobre la calidad del producto y se han usado algunas herramientas para comprobar objetivamente

la calidad del mismo. En esta sección se tratarán las herramientas usadas para validar el proyecto de forma objetiva.

### 6.8.1. Lighthouse

Lighthouse es una herramienta automatizada de código abierto que ofrece informes sobre el rendimiento, la accesibilidad, o el posicionamiento SEO, entre otros de las aplicaciones web objeto de análisis.

Al auditar una página, Lighthouse ejecuta un aluvión de pruebas contra la página y luego genera un informe sobre el comportamiento de la página. Desde aquí pueden utilizar las pruebas de falla como indicadores de qué acciones se pueden llevar a cabo para mejorar la aplicación.[14]

Figura 6.18: Logo de Lighthouse


**Fuente:** Chrome Web Store [https://chrome.google.com/webstore/detail/lighthouse/blipmdconlkpinefehnmjammfjpmpbjk?hl=es]

### 6.8.2. SonarQube

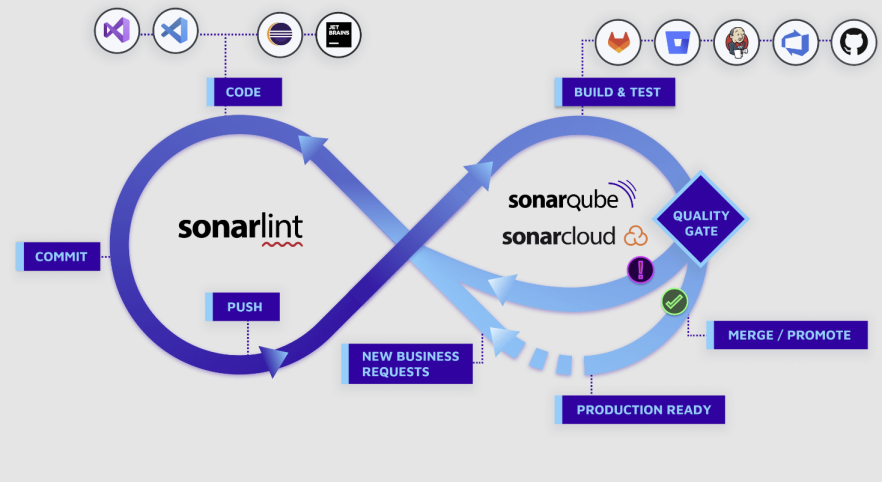
SonarQube es una herramienta de revisión de código automática y *autogestionada* que facilita de forma sistemática [Figura 6.19] la entrega de código limpio. Es una herramienta de análisis estático del software. SonarQube se integra en un flujo de trabajo existente y detecta problemas en el software para ayudar a realizar inspecciones continuas del código desarrollado. La herramienta incluye más de 30 lenguajes de programación diferentes para garantizar que una gran variedad de código cumpla con los estándares de alta calidad[15].

Con esta herramienta se pretende llegar a tener la menor cantidad de *Code Smells*[1] posible, así como que la deuda técnica[2] sea muy baja o inexistente. Otro de sus objetivos es controlar la aparición de *bugs*, o vulnerabilidades en el código. En definitiva, es una herramienta que facilita el control de la calidad del software desarrollado en la que se pueden configurar los parámetros de calidad que queremos controlar.

[1]El **code smell** es una indicación de la mala calidad del código. Si hay *code smells*, quien lea el código tendrá la sensación de que algo está mal. Este se soluciona refactorizando código.[[16]]

[2]La **deuda técnica** es el costo del *retrabajo* adicional causado por la elección de la solución más rápida en lugar de la más efectiva.[17]

Figura 6.19: Ciclo de revisión de calidad con SonarQube




**Fuente:** Página oficial de SonarQube https://docs.sonarqube.org/latest/

## 6.9. Control de versiones

### 6.9.1. Git

La información de este apartado se ha obtenido desde la página de Atlassian[18]

Git es un sistema de control de versiones gratuito y de código abierto, creado originalmente por Linus Torvalds en 2005. Con Git, cada desarrollador tiene el historial completo de su repositorio de código localmente. Esto hace que la clonación inicial del repositorio sea más lenta, pero las operaciones posteriores, como confirmar, diferenciar, fusionar y registrar, son mucho más rápidas.

Git también tiene un excelente soporte para bifurcar, fusionar y reescribir el historial del repositorio, lo que ha dado lugar a muchos flujos de trabajo y herramientas innovadoras y potentes. Las solicitudes de incorporación de cambios son una de esas herramientas populares que permiten a los equipos colaborar en ramas de Git y revisar de manera eficiente el código de los demás. Git es el sistema de control de versiones más utilizado en el mundo actual y se considera el estándar moderno para el desarrollo de software.

Las órdenes básicas de Git son:

- git clone: clona un repositorio existente en un nuevo directorio.

- git add: agrega cambios al área de preparación (*staging area*) para que estén listos para ser confirmados.

- git commit: crea un nuevo *commit* (instantánea) con los cambios agregados al área de preparación y agrega un mensaje de confirmación que describe los cambios.

- git status: muestra el estado actual del repositorio, incluyendo los cambios sin confirmar y los archivos sin seguimiento.

- git log: muestra una lista de todos los *commits* en orden cronológico inverso.

- git pull: actualiza el repositorio local con los cambios más recientes del repositorio remoto.

- git push: envía los cambios locales al repositorio remoto.

- git branch: muestra una lista de todas las ramas en el repositorio.

- git checkout: cambia a otra rama o *commit*.

### 6.9.2. GitKraken

GitKraken es una herramienta de gestión de versiones de código que ofrece una interfaz visual y fácil de usar para trabajar con Git. Es una aplicación de escritorio que permite a los desarrolladores y equipos de desarrollo colaborar en proyectos de software de manera eficiente.

Con GitKraken, se pueden realizar operaciones comunes de Git, mencionadas en el apartado anterior. Además, cuenta con características adicionales, como la posibilidad de integrarse con plataformas de gestión de proyectos como Trello, la capacidad de ver el historial de cambios en el código, y la opción de visualizar y comparar ramas de Git.

Grandes empresas hacen uso de esta herramienta, empresas tales como Netflix, Philips, Amazon, Unity y Disney entre otras.

# Capítulo 7

# Metodología

## 7.1. Marco de Trabajo SCRUM

Para la creación de la aplicación Web3 se ha seguido un marco de trabajo inspirado en **SCRUM**[19] para un desarrollo ágil. Este marco de trabajo está fundamentado por la existencia de etapas de tiempo previamente establecidas donde se enfoca la carga de trabajo, consiguiendo un producto mínimamente viable en cada una de ellas. Estas etapas de tiempo se conocen por el nombre de *Sprints*, y cada uno de ellos se divide en diferentes fases: planificación del *Sprint*, realización del *Sprints*, con reuniones diarias, y retrospectiva del *Sprint*. En la Figura 7.1 se muestra el flujo de trabajo que se suele seguir en la metodología SCRUM.

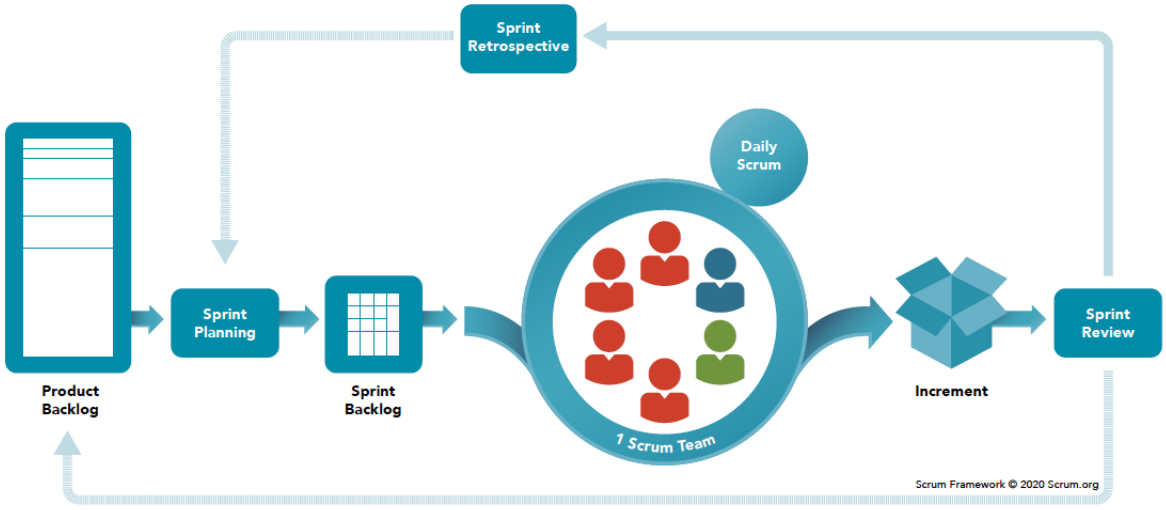
**Planificación del *Sprints***: En esta fase, se definen los objetivos del *Sprint*, identifican las tareas necesarias para cumplir esos objetivos y determinan la cantidad de trabajo que se puede realizar en el *Sprint*.

**Sprint**: Esta es la fase en la que se realiza el trabajo. Durante el *Sprint*, se trabaja en las tareas definidas en la planificación del *Sprint* para lograr los objetivos establecidos.

**Reunión diaria**: En esta fase, el equipo Scrum se reúne diariamente para revisar el progreso del trabajo, identificar cualquier problema y hacer ajustes necesarios para alcanzar los objetivos del *Sprint*.

**Revisión del *Sprint***: Al final del *Sprint*, el equipo Scrum se reúne para

Figura 7.1: Metodología SCRUM




**Fuente:** www.scrum.org [https://www.scrum.org/learning-series/what-is-scrum]

revisar el trabajo completado y demostrar los resultados a los interesados. Esto ayuda a identificar los logros y también a las áreas donde se pueden mejorar.

**Retrospectiva del *Sprints***: En esta fase, el equipo Scrum reflexiona sobre el *Sprint* anterior y discute qué se hizo bien y qué se puede mejorar en el próximo *Sprint*.

**Planificación de lanzamiento**: Si se han completado múltiples *Sprints*, el equipo Scrum se reúne para planificar la entrega del producto final al cliente.

A la hora de hacerlo, al no tener un equipo al uso, sino ser únicamente una persona, la **planificación** del *Sprints* se realizaba mediante el uso de Trello[6.4], añadiendo tareas, y colocando sus respectivos ”tests”, que en su mayoría son validaciones por parte de personas externas.

La **realización del *Sprints*** se realizaba en diferentes apartados, los cuales son:

En el **Análisis de requisitos** se recopilan y comprenden los requisitos de las historias de usuario planteadas:

- Se observaban páginas ya funcionales dentro del mundo de la web3, como

son Coinbase, OpenSea o Binance, antes mencionadas.

- Se analizaban las funcionalidades que ofrecían, y se planteaban las que se querían implementar en la aplicación.

A continuación, venía la parte de **diseño**, realizado en Figma[Sección 6.5], en esta etapa se creaba un diseño estructural de la página, de los componentes y de la interfaz de usuario:

- Diseño de los distintos **componentes**
- Diseño de **páginas** agregando los componentes individuales diseñados

La parte de **desarrollo de código** realizado en Visual Studio Code[Sección 6.6] haciendo uso de las distintas herramientas antes mencionadas[Capítulo 6].

- Desarrollo de la **página** para poder colocar los componentes
- Desarrollo de los **componentes** e insertarlos de forma adecuada en la página
- Desarrollos de los distintos **servicios** necesarios para el correcto funcionamiento de los componentes y de las funcionalidades
- Desarrollo de las **estructuras de datos** necesarios para tener un mejor control de los mismos en la aplicación

La parte de **despliegue y validación**. El despliegue se hace con Vercel 6.7, para las validaciones se hace uso de Lighthouse[Sección 6.8.1] y SonarQube[Sección 6.8.2]

- Siguiendo el consejo antes mencionado de SonarQube, se ejecuta el test antes de hacer un *push*, comprobando la posible existencia de *Code Smells*, *bugs*, vulnerabilidades, etc. En caso de una solución rápida, se solucionaba al momento. Por el contrario, si es algo complejo se creaba una nueva historia técnica de *refactorización* para poder mejorarlo.

- Despliegue de la rama en Vercel y comprobación del correcto funcionamiento de la página estando al público

- Despliegue de la rama principal (*main*) con todos los cambios realizados en el *Sprints*

- A la hora de pasar la validación de Lighthouse se hace cada vez que se hacía un *merge* a la rama principal (*main*), se realizaba este test con la intención de ver las posibles fallas de accesibilidad que tenía la página en ese momento y poder solucionarlas. Se hacía igual que en el apartado anterior, en caso de ser sencilla, se solucionaba al momento, y en caso de requerir un mayor tiempo de mejora, se creaba una nueva historia técnica.

## 7.2. Historias de usuario principales

A continuación se explicarán algunas historias de usuario [20] que fueron las prioridades para el primer sprint:

Como usuario no registrado quiero poder registrarme [Figura 7.2].
En la figura 7.2 se puede observar el nombre de la historia de usuario, una descripción de la misma, y las tareas a las cuales se le asigna esta historia de usuario. En este caso, se ve que las tareas asignadas a esta historia de usuario son *Diseño de registro*, *Firebase - Autenticación Registros* y *Implementación de registro*

Dentro de las distintas tareas se pueden observar un número de objetivos a cumplir para poder asumir completa tal tarea [Figura 7.3].

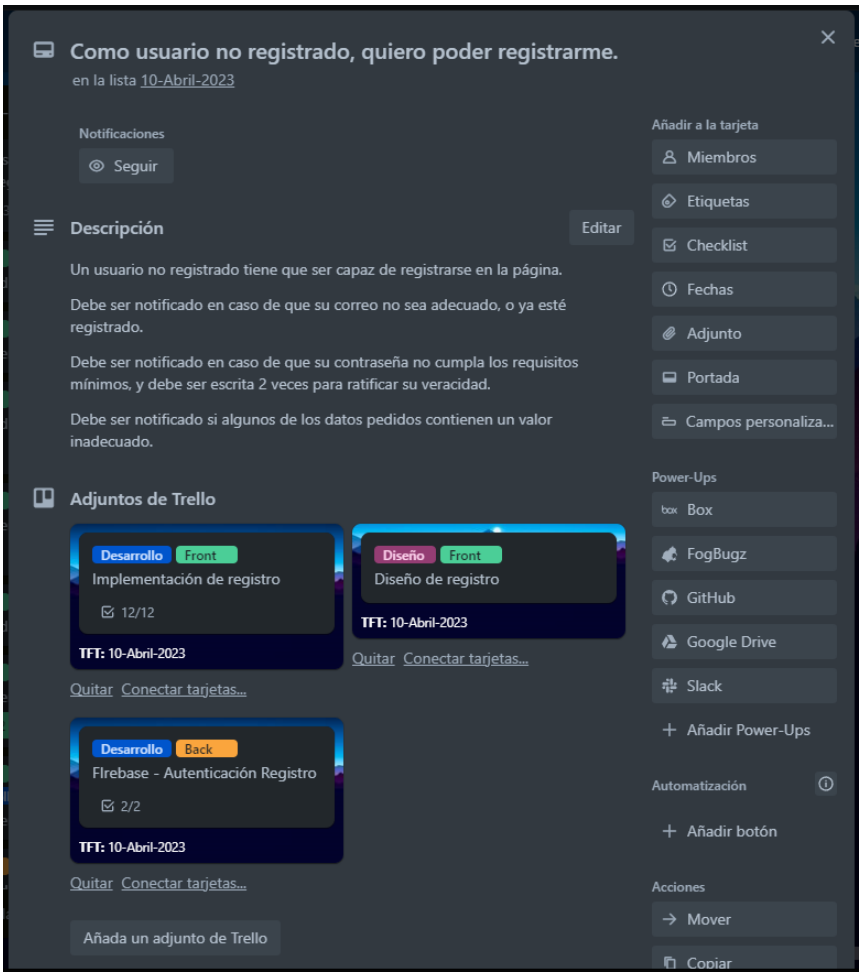
Como usuario registrado quiero poder acceder a la página de carteras.[Figura 7.4].

Las historias de usuario básicas para el administrador fueron: Como usuario administrador quiero poder auditar los usuarios registrados.

Como usuario administrador quiero poder CRUD[1] de las colecciones que se puedan visualizar en la página de la galería.

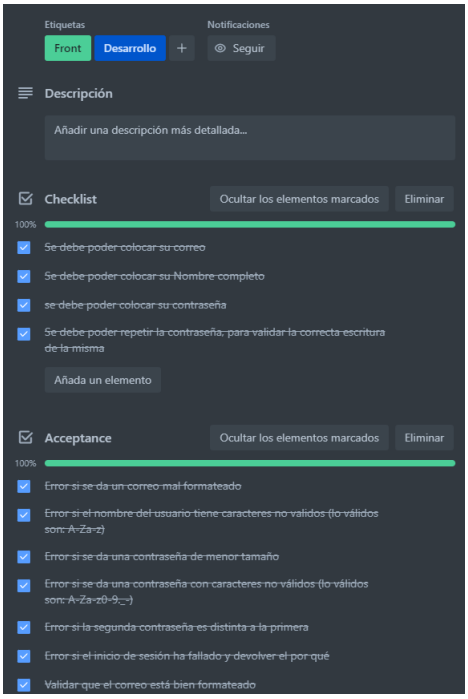
[1]CRUD hace referencia al acrónimo Crear, Leer, Actualizar y Eliminar, en inglés (Create, Read, Update y Delete)

Figura 7.2: Historia de usuario - usuario no registrado - registro



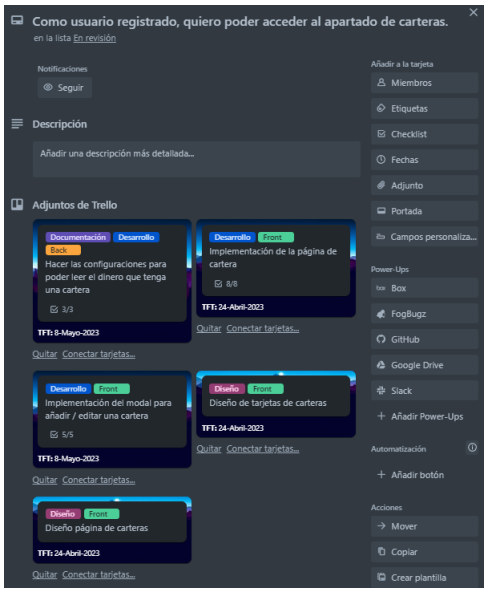

**Fuente:** Trello del proyecto

Figura 7.3: Objetivos a cumplir para la tarea Implementación de registro




**Fuente:** Trello del proyecto

Figura 7.4: Historia de usuario - usuario registrado - acceso a página de carteras




**Fuente:** Trello del proyecto

# Capítulo 8

# Análisis y Diseño

## 8.1. Análisis de páginas de compraventa de NFTs

### 8.1.1. OpenSea

**OpenSea** [https://opensea.io] es uno de los mercados más grandes y populares para NFTs (Tokens No Fungibles). Fue fundado en 2017 y se ha convertido en un punto de referencia importante en el ecosistema de la web 3.0. Su objetivo es proporcionar una plataforma abierta y descentralizada para que los usuarios compren, vendan y descubran una amplia gama de NFTs.

Una de las características distintivas de OpenSea es su **amplia selección de NFTs** disponibles para su compra y venta. La plataforma alberga una gran variedad de categorías de NFTs, que incluyen arte digital, coleccionables, dominios virtuales, avatares, *items* de juegos y mucho más. Los usuarios pueden explorar diferentes colecciones, filtrar por categorías y descubrir nuevas y emocionantes obras de arte digital y otros activos digitales únicos.

La plataforma también proporciona una **interfaz intuitiva y fácil de usar**. Los usuarios pueden navegar por los listados de NFTs, ver detalles como la descripción, imágenes y precio, así como interactuar con los vendedores y realizar transacciones de compra o venta directamente en la plataforma.

Además de la compra y venta de NFTs, OpenSea también ofrece otras funcionalidades interesantes. Por ejemplo, los usuarios pueden crear sus propias tiendas y exhibir sus NFTs en ellas. También pueden crear subastas para vender sus activos digitales al mejor postor, lo que permite una mayor flexibilidad

en términos de precios y descubrimiento de nuevos compradores.

OpenSea ha ganado una gran popularidad debido a su amplia oferta de NFTs, su facilidad de uso y su enfoque en la descentralización. Ha sido adoptado por una amplia comunidad de artistas, coleccionistas y entusiastas de la tecnología blockchain. La plataforma ha sido testigo de algunas ventas destacadas de NFTs, incluyendo obras de arte digitales, tarjetas de coleccionables y objetos de juegos, lo que ha contribuido a su crecimiento y reconocimiento en el espacio de los NFTs.

### 8.1.2. Binance

Aunque **Binance** [https://www.binance.com/es/nft/home] inicialmente se centró en el comercio de criptomonedas, también ha ampliado su oferta para incluir la compraventa de NFTs. Algunas razones por las que Binance puede considerarse una buena opción para la compraventa de NFTs:

**Amplia selección de NFTs**: Binance, al igual que OpenSea, ofrece una amplia selección de NFTs de diferentes categorías, como arte digital, coleccionables, juegos y más. La plataforma colabora con diversos creadores, proyectos y marcas para ofrecer una amplia variedad de opciones a los usuarios interesados en adquirir NFTs.

**Integración con la plataforma de criptomonedas**: Permite a los usuarios comprar y vender NFTs utilizando criptomonedas, como Bitcoin (BTC), Ethereum (ETH) y su propia moneda nativa, Binance Coin (BNB). Esto brinda a los usuarios la comodidad de utilizar una plataforma consolidada y confiable para realizar transacciones tanto de criptomonedas como de NFTs.

**Seguridad y confianza**: Es conocido por su enfoque en la seguridad y la protección de los fondos de sus usuarios. La plataforma implementa medidas de seguridad avanzadas, como autenticación de dos factores, almacenamiento en frío de criptomonedas y auditorías regulares de seguridad. Esto brinda tranquilidad a los usuarios que desean comprar y vender NFTs en un entorno seguro.

**Interfaz intuitiva y fácil de usar**: Binance, así como OpenSea, ofrece una interfaz amigable y fácil de navegar, lo que facilita a los usuarios encontrar, comprar y vender NFTs. La plataforma proporciona herramientas y características útiles, como filtros de búsqueda, clasificaciones y descripciones detalladas de los NFTs, para ayudar a los usuarios a tomar decisiones

informadas.

**Participación en eventos y lanzamientos exclusivos**: Organiza eventos y colabora con proyectos para lanzamientos exclusivos de NFTs. Esto brinda a los usuarios la oportunidad de acceder a NFTs exclusivos, ediciones limitadas y otras oportunidades de inversión o colección que pueden no estar disponibles en otros lugares.

## 8.2. Diseño del proyecto

El diseño desempeña un papel fundamental en el desarrollo de cualquier proyecto, y en el contexto de este Trabajo de Fin de Grado, es esencial abordar el diseño en diferentes aspectos clave. En este apartado, se explorarán tres componentes fundamentales del diseño: el diseño de la base de datos (BBDD), el diseño de la estructura del proyecto y el diseño estético de la página.

### 8.2.1. Diseño de la Base de Datos

El diseño de la base de datos es un proceso crítico en el desarrollo de aplicaciones basadas en bases de datos. Se centra en la organización y estructuración de la información de manera eficiente y coherente. Mediante un análisis exhaustivo de los requisitos y objetivos del proyecto, se definirán las tablas, los campos, las relaciones y las restricciones necesarias para garantizar la integridad y la eficacia de la base de datos.

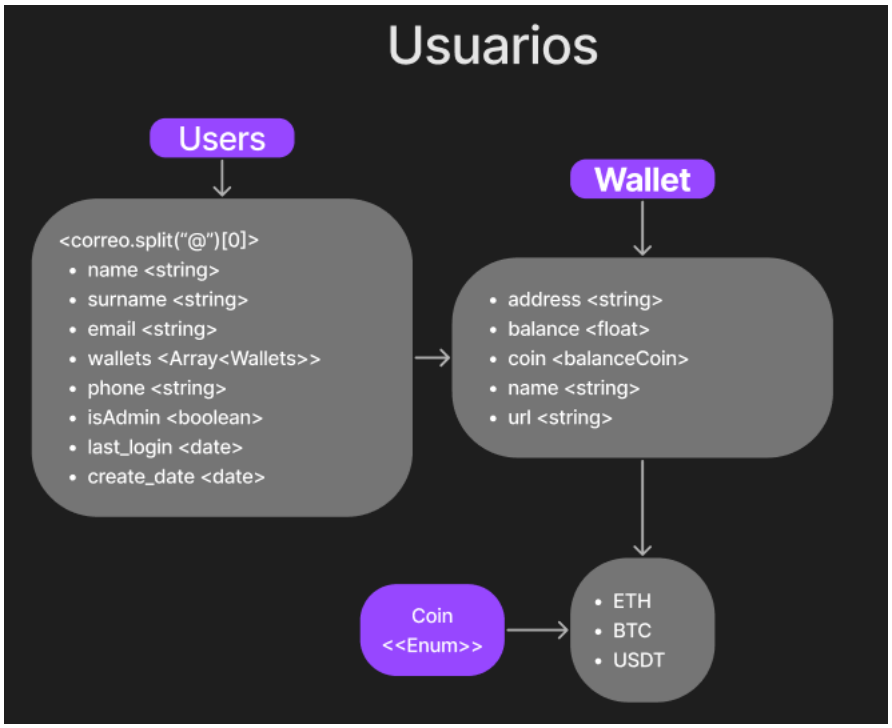
#### 8.2.1.1. Usuarios

Esta estructura de datos permite almacenar la información relevante de un usuario y sus carteras en el sistema. Proporciona los detalles necesarios para identificar y gestionar las cuentas de los usuarios, así como las carteras asociadas a ellos.

Un usuario tiene los siguientes atributos:

**Nombre**: El nombre del usuario, que representa su identificación personal.
**Apellido**: El apellido del usuario, que complementa su identificación personal.
**Correo**: La dirección de correo electrónico del usuario, que se para la identificación única en el sistema.
**Carteras**: Una colección de carteras asociadas al usuario. Cada cartera representa una cuenta de activos digitales asociados al usuario en el sistema.
**Teléfono**: El número de teléfono del usuario.
**Administrador**: Un indicador booleano que determina si el usuario tiene privilegios de administrador en el sistema.

Figura 8.1: Datos de Usuario

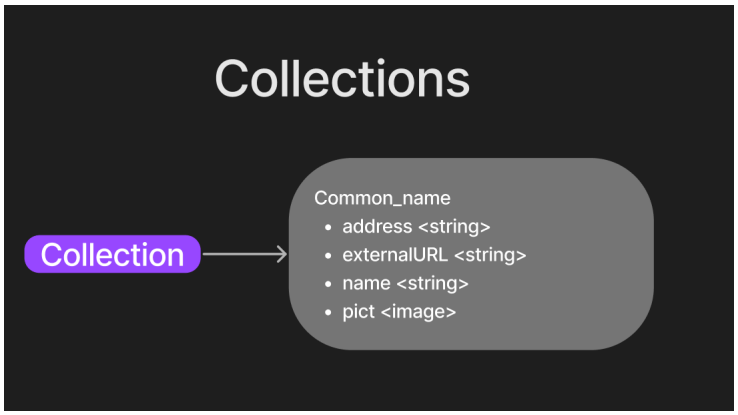



**Fuente:** Fichero FigJam personal de Figma

**Último inicio de sesión**: La fecha y hora del último inicio de sesión del usuario en el sistema.
**Fecha de creación**: La fecha y hora en la que se creó la cuenta del usuario en el sistema.

Cada cartera asociada al usuario contiene los siguientes atributos:
**Dirección**: La dirección de la cartera, que es una cadena única utilizada para identificar la cartera en el sistema de criptomonedas.
**Balance**: El saldo actual de la cartera, que indica la cantidad de activos digitales que se encuentran en la cartera en un momento dado.
**Moneda**: La moneda asociada a la cartera, que representa el tipo de activo digital que se almacena en la cartera (por ejemplo, Bitcoin, Ethereum, etc.).
**Nombre**: Un nombre descriptivo opcional para la cartera, que puede ser utilizado para identificar y diferenciar las carteras en el sistema. Esto pretende que el usuario se sienta más familiarizado con sus carteras, ya que las direcciones de las mismas son una ristra de caracteres.
**URL**: Una URL opcional que puede asociarse con la cartera, por ejemplo, para proporcionar enlaces a exploradores de bloques o interfaces adicionales para interactuar con la cartera.

Figura 8.2: Datos de Colección




**Fuente:** Fichero FigJam personal de Figma

#### 8.2.1.2. Colecciones

La estructura de datos de las colecciones permite organizar y gestionar la información relacionada con las distintas colecciones presentes en el sistema.

Una colección tiene los siguientes atributos:

**Nombre**: El nombre asignado a la colección, que permite al administrador identificarla fácilmente.
**Dirección**: La dirección única asignada a la colección en el sistema, que se utiliza para su identificación y acceso.
**URL de la página**: La URL de la página a la que pertenece la colección. Puede ser la página principal de la colección, una página de detalles o cualquier otra página relacionada.
**Nombre real**: El nombre completo o el nombre real de la colección. Esto puede ser útil para proporcionar información adicional sobre la colección, como el nombre del artista o el nombre de la marca.
**Imagen de marca**: La imagen de marca asociada a la colección. Esto puede ser un logotipo, una imagen representativa o cualquier otra imagen que identifique visualmente la colección.

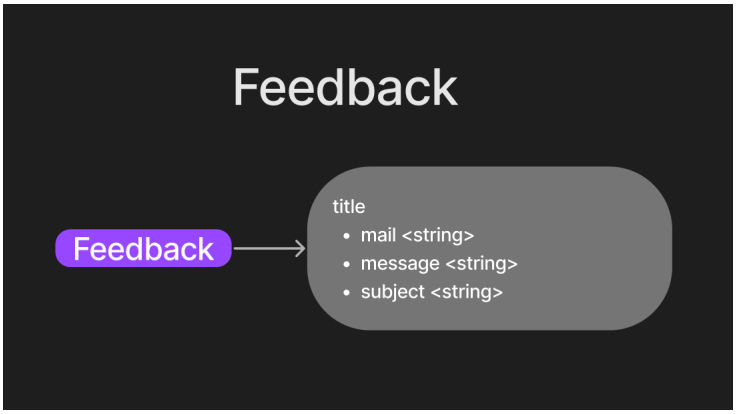
Almacenar esta información en la estructura de datos permite a los administradores, administrar de una manera efectiva, facilitando la búsqueda y la identificación de las mismas. También permite a los usuarios poder acceder y *redireccionar* entre ellas de una manera sencilla.

#### 8.2.1.3. Feedback

La estructura de datos de *Feedback* permite a los usuarios enviar sus comentarios y opiniones sobre la página, proporcionando una retroalimentación

valiosa para los administradores.

Figura 8.3: Datos de *Feedback*




**Fuente:** Fichero FigJam personal de Figma

**Título**: El título o encabezado del mensaje de *Feedback*, que proporciona una breve descripción del tema o contenido principal del mensaje.
**Correo**: La dirección de correo electrónico de la persona que envía el mensaje de *Feedback*. No es necesario que la persona esté registrada en la página para enviar *Feedback*.
**Asunto**: El asunto o tema específico del mensaje de *Feedback*, que ayuda a categorizar y organizar las retroalimentaciones recibidas.
**Mensaje**: El contenido del mensaje de *Feedback* en sí, que incluye la información detallada, comentarios, sugerencias o cualquier otra retroalimentación que la persona quiera compartir con los administradores.

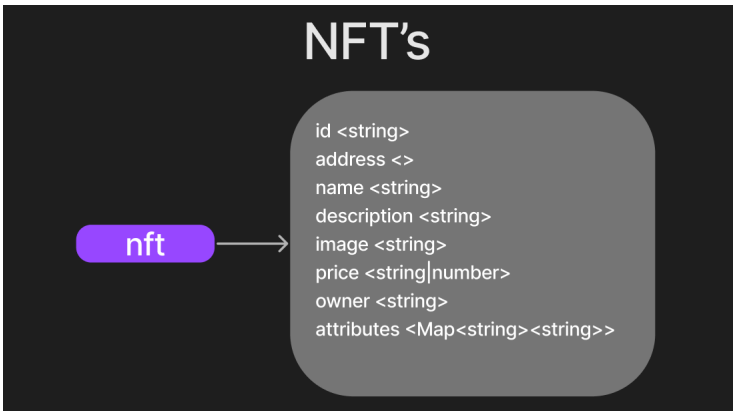
Almacenar esta información permite a los administradores recopilar, revisar y considerar las opiniones de los usuarios para mejorar continuamente la página y brindar una mejor experiencia a sus usuarios. La retroalimentación puede abordar aspectos como la *usabilidad*, el diseño, las funcionalidades o cualquier otro aspecto relacionado con la página, lo que permite a los administradores tomar decisiones informadas y realizar ajustes necesarios en base a las necesidades y sugerencias de los usuarios.

#### 8.2.1.4. NFT

La estructura de datos de los NFTs permite almacenar la información necesaria para *renderizar* y acceder a las cartas de los NFTs. Esto facilita la visualización y el acceso a los NFTs, así como su seguimiento y gestión en un entorno de blockchain.

**ID del NFT**: Un identificador único que se asigna al NFT para su identificación y seguimiento en el sistema.
**Dirección del NFT**: La dirección única asociada al NFT en el sistema de

Figura 8.4: Datos de NFT




**Fuente:** Fichero FigJam personal de Figma

blockchain. Esta dirección permite localizar y acceder al NFT en la cadena de bloques correspondiente.
**Nombre del NFT**: El nombre del NFT, que proporciona una identificación descriptiva del activo digital único.
**Descripción del NFT**: Una descripción detallada que proporciona información adicional sobre el NFT, como su contexto, historia, detalles técnicos, etc.
**Imagen del NFT**: La imagen o representación visual del NFT, que puede ser una ilustración, una imagen digital, una obra de arte, etc.
**Precio base del NFT**: El precio inicial o base establecido para el NFT. Este precio puede utilizarse como referencia para las transacciones o para establecer un valor inicial en un mercado.
**Propietario del NFT**: La persona o entidad que posee actualmente el NFT. Esto puede incluir tanto usuarios individuales como cuentas de plataformas o contratos inteligentes.
**Atributos del NFT**: Los atributos específicos que caracterizan al NFT. Estos atributos pueden ser datos adicionales que proporcionan información sobre el NFT, como el color de fondo, los accesorios que se pueden encontrar dentro del NFT, marcas que aparezcan, gestos, etc.

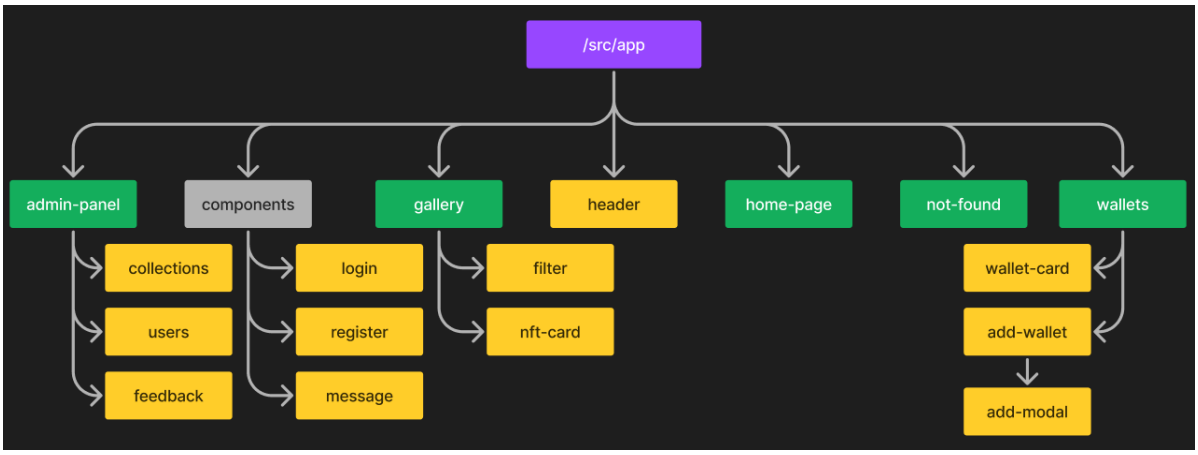
### 8.2.2. Diseño de la Estructura del Proyecto

El diseño de la estructura del proyecto se ocupa de la organización y la arquitectura general del proyecto. Se definen los módulos, componentes y su interacción, así como las tecnologías y herramientas que se utilizarán en el desarrollo. Este diseño estructural tiene como objetivo asegurar la modularidad, la escalabilidad y la mantenibilidad del proyecto, permitiendo un desarrollo eficiente y una fácil comprensión de su estructura global.

Como se especificó en el capítulo 6, se ha utilizado el Framework Angular [Sección 6.1]. Se explicó que Angular funciona por componentes, pero para

facilitar la compresión de la estructura, veremos en la figura 8.5 una diferenciación entre "páginaz çomponente".

Figura 8.5: Estructura del proyecto




**Fuente:** Fichero FigJam personal de Figma

Como podemos ver en la imagen 8.5, los cuadrados verdes serían las **páginas**, y los cuadrados **naranjas** serían los componentes.

Podemos ver que en la página **administrador** encontramos 3 componentes, que hacen referencia a los 3 usos que tienen los administradores, ver, y editar las colecciones que se pueden ver en la página de colecciones; ver los usuarios registrados; y ver el *feedback* que se haya podido enviar.

Existe un directorio llamado **components**, el cual contiene 3 componentes que se pueden usar en cualquier dirección de la página, estos son los modales de inicio de sesión y registro, y el modal de mensaje.

En la página de galería podemos ver que se hace uso de un componente filtro, el cual hace uso de traspaso de información de padre a hijo y de hijo a padre (esto se verá mejor en el apartado de desarrollo [Capítulo 7]), también podemos ver el componente de nft-card, que muestra visualmente la información de un NFT.

A continuación, encontraremos el componente del **header**, el cual se usa en el fichero base, solo se crea 1 vez.

La página del **home**, hace referencia a la *landing page* [1] de la aplicación

[1]Una *landing page* o página de destino es una página web diseñada para lograr un solo objetivo: convertir visitantes en clientes o *leads*. [21]

web, donde nuestro objetivo principal es que el usuario se registre o inicie sesión.

La página de **404 o página no encontrada** se hizo con la intención de verificar el correcto funcionamiento del *redireccionamiento* tanto dentro como fuera del dominio de la página. Una vez se entendió el funcionamiento del mismo se decidió dejar, ya que, por lo general, la gran mayoría de páginas web tienen una.

Por último, la página de las **carteras**, donde podemos encontrar 3 componentes, una carta con la información de la cartera, una carta con la capacidad para crear una nueva cartera, y el modal que genera la carta de añadir cartera.

### 8.2.3. Diseño Gráfico de la Página

Además, se abordará el gráfico estético de la página, que tiene un impacto significativo en la experiencia del usuario y la *usabilidad* de la aplicación. Aquí se considerarán aspectos como la elección de colores, tipografía, disposición de elementos y flujo de navegación. El diseño estético busca crear una interfaz atractiva, intuitiva y fácil de usar, brindando una experiencia agradable y satisfactoria para los usuarios.

# Capítulo 9

# Desarrollo

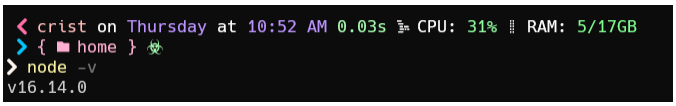
En estos subapartados se comentará la información básica utilizada para la ejecución del Trabajo de Fin de Grado.

## 9.1. Uso básico de Angular

En este apartado se comentará como empezar un proyecto en Angular de manera sencilla y cuales son los comandos básicos para crear los distintos Componentes que nos aporta este Framework, así como la ejecución del programa en local y la compilación de este.

Angular es un **Framework basado en NodeJS** [Sección 6.1], por lo que lo primero será tener NodeJS instalado en el sistema. Se puede descargar desde la web oficial de Node.js (https://nodejs.org), e instalarlo siguiendo las instrucciones dependiendo del sistema operativo. Para comprobar la correcta instalación del mismo podemos ejecutar la orden de la figura 9.1 dentro de una terminal, y debe devolverlos la versión que estamos utilizando.

Figura 9.1: Herramientas de Figma




**Fuente:** Terminal de mi sistema Windows

Una vez instalado NodeJS, se puede proceder a la instalación de Angular. Para ello, se abrirá una terminal y se ejecutará el siguiente comando:

```
npm install -g @angular/cli
```

Ya podremos hacer uso de los comandos que nos aporta Angular, a continuación veremos algunos de los más útiles: Usaré las acotaciones de los comandos:

- g - *Generate*, genera un elemento.
- c - *Component*, componente de Angular compuesto por ficheros HTML, SCSS y TypeScript.
- s - *Service*, servicio de Angular, un servicio se usa como ïnterfaz"que aporta funciones auxiliares a componentes.

Para crear un nuevo proyecto de Angular usaremos:

```
ng new nombre-del-proyecto
```

Tras este comando, tendremos que entrar en la carpeta para poder estar en un entorno de Angular:

```
cd nombre-del-proyecto
```

En las últimas versiones de Angular, no viene el directorio de Environments, crucial para incorporar las *Keys* (claves) de las API's:

```
ng g environments
```

Para crear un componente se usa el siguiente comando:

```
ng g c nombre_del_componente
```

Para crear un servicio se usa el siguiente comando:

```
ng g s nombre_del_servicio
```

Para poder ver el desarrollo de nuestro proyecto podemos usar:

```
ng serve # También se puede usar $ npm run start
```

Otra utilidad que nos aporta angular/node, es la capacidad de compilar el proyecto y devolvernos una carpeta *dist* donde encontramos el proyecto listo para desplegarlo.

```
ng build # También se puede usar $ npm run build
```

## 9.2. Uso básico de Firebase

En este apartado se comentarán los conocimientos adquiridos tanto en el transcurso de la universidad como por la iniciativa de aprender a usarlo de forma correcta.

Me centraré en explicar el funcionamiento de las que he utilizado:

### 9.2.1. Autenticación

Firebase Authentication nos permite tener ciertas funciones para poder registrar, iniciar sesión y cerrar sesión de los usuarios dentro de la aplicación web.

Registro:

```
import { getAuth, createUserWithEmailAndPassword } from "firebase/auth";
const auth = getAuth();
createUserWithEmailAndPassword(auth, email, password)
    .then((userCredential) => {
        // Signed in
        const user = userCredential.user;
    }).catch((error) => { const error = [error.code, error.message]; });
```

Inicio de sesión:

```
import { getAuth, signInWithEmailAndPassword } from "firebase/auth";
const auth = getAuth();
signInWithEmailAndPassword(auth, email, password)
    .then((userCredential) => {
        // Signed in
        const user = userCredential.user;
    }).catch((error) => { const error = [error.code, error.message]; });
```

Cierre de sesión:

```
import { getAuth, signOut } from "firebase/auth";
const auth = getAth();
signOut(auth).then(() => {
    // Sign-out successful.
}).catch((error) => { const error = [error.code, error.message]; });
```

### 9.2.2. Base de datos en la Nube

Cloud Firestore nos permite tener ciertas funciones para poder crear colecciones, documentos dentro de estas colecciones y atributos dentro de estos documentos. En resumen, nos facilita el control de una base de datos no relacional.
Creación y modificación de un documento:

```
import { doc, setDoc } from "firebase/firestore";
// Add a new document in collection "collections"
await setDoc(doc(db, "collections", "nombre_coleccion"), {
    address: "0xa49a0e5ef83cf89ac8aae182f22e6464b229e",
    externalURL: "https://rtfkt.com",
    name: "RTFKT"
    pict: "https://cdn.rtfkt.com/assets/logotype.png"
});
```

Obtención de un documento pasado por parámetros:

```
import { doc, getDoc } from "firebase/firestore";
const docRef = doc(db, "collections", "nombre_coleccion");
const docSnap = await getDoc(docRef);
if (docSnap.exists()) console.log("Document data:", docSnap.data());
else console.log("No such document!");
```

Obtención de más de un documento en función de algún parámetro:

```
import { collection, query, where, getDocs } from "firebase/firestore";
const q = query(collection(db, "users"), where("isAdmin", "==", true));
const querySnapshot = await getDocs(q);
querySnapshot.forEach((doc) => {
  // doc.data() is never undefined for query doc snapshots
  console.log(doc.id, " => ", doc.data());
});
```

## 9.3. Uso básico de Vercel

En este apartado se comentarán los conocimientos adquiridos de *motu propio* para poder desplegar una página web con Angular y una API en ExpressJS en Vercel.
La principal ventaja que nos aporta vercel es un despliegue inteligente, ya que la propia página verifica las características y ficheros de configuración existen en el proyecto, y hace una compilación del mismo para desplegarlo en la página, además de hacer el *host* de las páginas ya compiladas.

En el caso de desplegar el Back-End dio complicaciones. Se empezó optando por tener *front* y *back* en el mismo repositorio de Git[1] y desplegar ambos a la vez mediante un fichero de configuración. Esto resultó ser más una complicación que una ayuda, ya que las veces que funcionaba el *Front*, y se podía ver la página correctamente, el *Back* no funcionaba. O pasaba lo contrario, se podían hacer peticiones a las páginas de *Back*, pero el *Front* no llegaba a compilar bien.

Esto se debe al fichero de configuración de Vercel. La documentación que aparece sobre el mismo fue poca y solo en documentación oficial, por lo que se tuvo que realizar diversas iteraciones a prueba y error para conseguir que llegara a funcionar.

Tras un un número infraestimado de horas intentando que esto funcionara, se buscó una forma más simple, la cual era desplegar por separado *Front* y *Back*.

### 9.3.1. Desplegar el Front-End

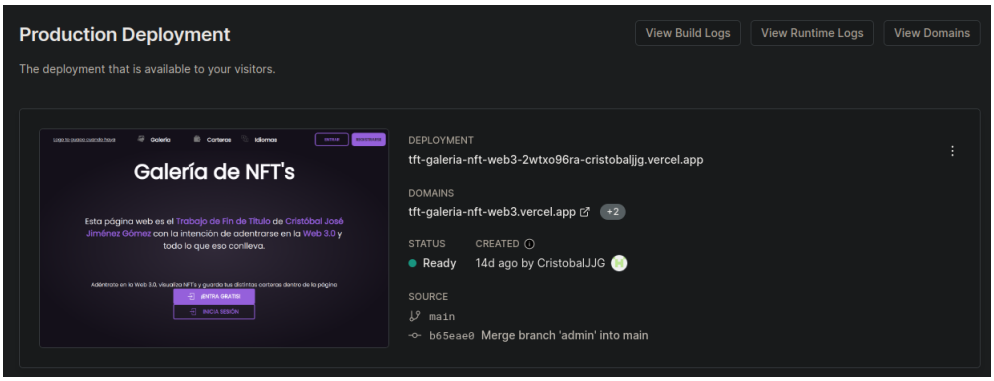
En el caso de desplegar el *Front-End* de la página, la estructura HTML, SCSS y TypeSCript, encontramos que con el simple hecho de colocar el repositorio nos detecta que es un proyecto de Angular. Nos permite cambiar las órdenes de compilar el proyecto, la carpeta con el proyecto compilado, o un cambio del directorio raíz.

### 9.3.2. Desplegar el Back-End

Se optó por realizar un nuevo repositorio solo para el *Back-End*, en el cual se ejecuta una API Rest, y se hacen llamadas desde el *Front* para poder recibir información.

[1]Un repositorio de Git es un almacenamiento virtual de tu proyecto. Te permite guardar versiones del código a las que puedes acceder cuando lo necesites. Lea más en www.atlassian.com Un repositorio en GitHub es un espacio en línea donde se almacena y comparte código.[22]

Figura 9.2: Despliegue de *Front* en Vercel




**Fuente:** Cuenta personal de Vercel

Y en este caso sí se pudo desarrollar un fichero de configuración de Vercel [Figura 9.5] para el correcto funcionamiento del servidor, así como la configuración de los CORS[23].

Figura 9.3: Fichero de configuración de Vercel

```
{
    "version": 2,
    "builds": [ { "src": "./index.js", "use": "@vercel/node" } ],
    "routes": [ {  "src": "/(.*)", "dest": "/" } ]
}
```

**Fuente:** Fichero de configuración de Back

Este archivo de configuración de Vercel [Figura 9.5] permite personalizar el proceso de compilación y enrutamiento de un proyecto en la plataforma de Vercel.

**builds**: Define la configuración de compilación para el proyecto. Se puede tener múltiples configuraciones de compilación en un solo proyecto.

- src: Especifica la ubicación del archivo principal del proyecto. En este caso, se está utilizando "*./index.js*" como archivo principal.

- use: Especifica el entorno de ejecución para el proyecto. Se usa, por tanto, "@vercel/node", que indica que el proyecto se ejecutará en un entorno de Node.js en Vercel.

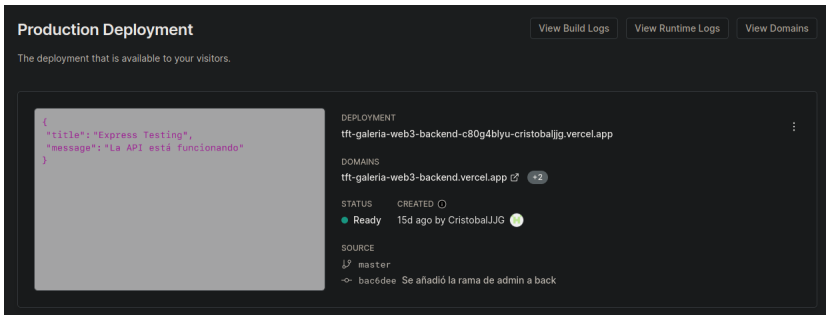
**routes**: Define las rutas y destinos para el enrutamiento personalizado en el proyecto.

- src: Especifica el patrón de ruta que se utilizará para redirigir las solicitu-

des. En este caso, se está utilizando /(.*) como patrón de ruta. Significa que cualquier ruta que coincida con cualquier cadena será redirigida.

- dest: Especifica la ruta de destino a la que se redirigirán las solicitudes que coincidan con el patrón de ruta. Aquí se utiliza "/" como destino,esto implica que todas las solicitudes se redirigirán a la raíz del proyecto.

Figura 9.4: Despliegue de Back en Vercel




**Fuente:** Fichero de configuración de Vercel ubicado en Back

## 9.4. Uso básico de Alchemy

Alchemy se enfoca más que Moralis en el concepto de los NFTs y las colecciones de los mismos, por eso se decidió utilizar esta para el desarrollo de la galería de NFTs.

Nos aporta un gran número de *EndPoints* con respecto a los NFTs:

- Recolección de metadatos en función de la dirección de la colección y del id del NFT.
- Recolección de los 100 primeros NFTs dada una colección.
- Recolección de precio base de un NFT.
- Recolección de NFTs obtenidos por un usuario dado.
- Recolección de los usuarios que tienen un NFT dando la dirección de la colección y el id del NFT.
- Dada una dirección y un NFT, validar si pertenece a esa cartera.

Figura 9.5: *Tutorial* de obtención de un NFT de Alchemy

```
import { Network, Alchemy } from "alchemy-sdk";
const settings = { apiKey: "demo", network: Network.ETH_MAINNET, };
const alchemy = new Alchemy(settings);
// Fetch metadata for a particular NFT:
console.log("fetching metadata for a Crypto Coven NFT...");
const response = await alchemy.nft.getNftMetadata(
  "0x5180db8F5c931aaE63c74266b211F580155ecac8", "1590"
);
// Print some commonly used fields:
console.log("NFT name: ", response.title);
console.log("token type: ", response.tokenType);
console.log("image url: ", response.rawMetadata.image);
```

**Fuente:** Alchemy Tutorial [https://docs.alchemy.com/reference/nft-api-quickstart]

## 9.5. Uso básico de Moralis

Como ya se mencionó en el apartado 6.3.3, Moralis tiene diversos tipos de APIs. En el caso de esta aplicación solo se ha hecho uso de la API de balances de una cartera.

Esta API se utiliza para obtener el **saldo total** de una dirección de billetera en una criptomoneda específica. Esto te permite mostrar el saldo actual de la billetera en tu aplicación y realizar cálculos relacionados. También puedes obtener saldos detallados de cada criptomoneda o token en particular.

Este sería el *tutorial* que nos aporta Moralis para el conocimiento y afianzamiento de esta API, la cual fue modificada para el Trabajo Final.

Figura 9.6: Tutorial de balances de Moralis

```
const Moralis = require("moralis").default;
const { EvmChain } = require("@moralisweb3/common-evm-utils");

const runApp = () => {
  await Moralis.start({
    apiKey: "YOUR_API_KEY",
    // ...and any other configuration
  });
  const address = '0xd8da6bf26964af9d7eed9e03e53415d37aa96045';
  const chain = EvmChain.ETHEREUM;
  const response = await Moralis.EvmApi.balance.getNativeBalance({
    address, chain,
  });
    console.log(response.toJSON());
}
runApp();
```

**Fuente:** Moralis Tutorial [https://docs.moralis.io/web3-data-api/evm/how-to-get-the-balance-of-a-wallet]

## 9.6. Uso básico de Figma

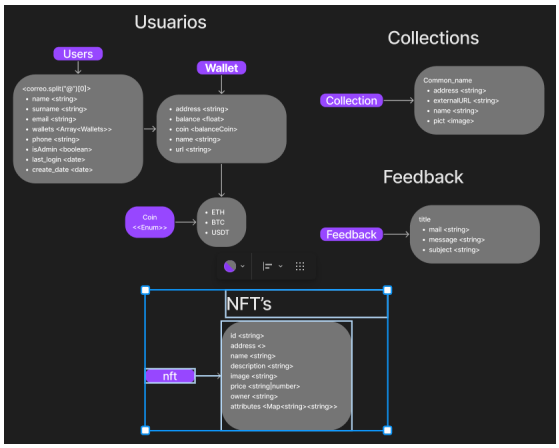
Como se especificó en el apartado 6, Figma tiene 2 tipos de componentes. Para el desarrollo de esquemas, tanto para este informe como para la correcta estructuración de la página web, se usó FigJam, mientras que para el diseño de la página se ha hecho uso de los ficheros de Figma.

### 9.6.1. FigJam

En este apartado se podrá encontrar los esquemas que se han desarrollado para el entendimiento de la estructura de datos [Figura 9.7], así como de la estructura a nivel de carpetas del proyecto [Figura 8.5].

Esta función ha aclarado los conceptos a la hora de crear las clases dentro del proyecto, ya que al poder ver los datos que tiene cada clase se hizo más sencillo la utilización de las mismas.

Figura 9.7: Fichero de FigJam



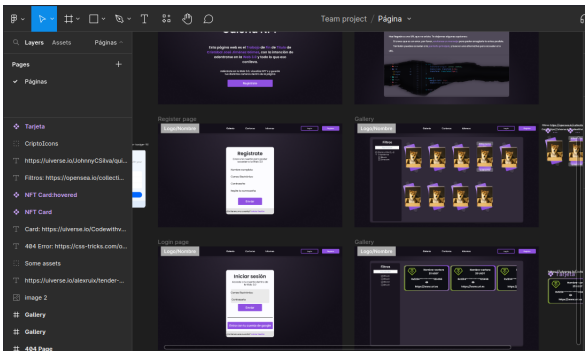

**Fuente:**Proyecto de FigJam para el TFT

### 9.6.2. Ficheros de Figma

Por otra parte, el fichero de Figma muestra los distintos diseños de los componentes y páginas [Figura 9.8]. Estas se han usado como bocetos para luego desarrollar con mayor fluidez.

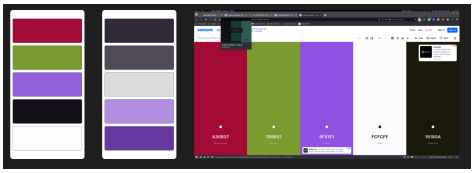
Uno de los primeros pasos en cuanto al diseño estético de la página, fue la selección de colores. Al principio se eligieron distintas tonalidades, pero por simplicidad se acabó eligiendo las tonalidades violetas que se puede ver en la figura 9.9, así como una gama de blancos y negros acorde con estos colores. Además, se eligieron un color verde para las posibles acciones acertadas del usuario, así como un color rojo para los posibles errores.

Figura 9.8: Fichero de Figma



**Fuente:** Proyecto de Figma para el TFT

Figura 9.9: Paleta de colores seleccionada



**Fuente:** Proyecto de Figma para el TFT

Con respecto a la página principal de la aplicación, la galería, se puede ver que el diseño de la misma [Figura 9.10] no dista mucho de la página de galería final [Figura 9.11]

Se pueden ver algunas diferencias:

- Las cartas de los NFTs se han simplificado debido al *feedback* de algunos compañeros y usuarios que han probado la página:
  - Se vio que eran demasiadas transiciones innecesarias.
  - Al tener los componentes minimizados, no había una captación directa de la información, ya que para ver el nombre y el precio del NFT se tenía que pasar el ratón por encima del mismo.
  - Para la versión móvil hubiera sido un recurso pésimo, debido a que en estos dispositivos ”no existe”la capacidad de hacer *hover* sobre un elemento.
- Se agregó el título de la Colección de NFTs, para que el usuario sepa en todo momento en cual está.

- Algunas de las ayudas visuales que podemos aportar a los usuarios pueden ser iconos, cambios de grosor o de tamaño en en las letras.

  - A la hora de estilizar el apartado de filtrado de NFTs, haciendo que las letras de las categorías tengan un mayor grosor.

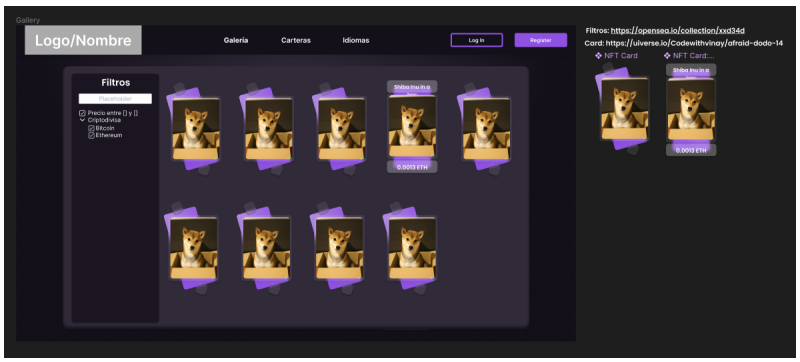
  - Los elementos del desplegable tienen un menor grosor, y están tabulados para entender que no forman parte del mismo componente.

  - Además, y por *feedback* de usuarios, se han separado el último elemento de la lista de desplegables con la categoría siguiente, para hacer ver a los usuarios que no son la misma categoría.

- Se agregó en la parte superior de la página un acceso para poder elegir la colección de NFTs a visualizar.

- Otro de los mayores cambios que se pueden ver es la separación que se puede encontrar entre las distintas cartas de NFTs. Revisando otras páginas, en ninguna había tanta separación entre dichas cartas, por lo que se decidió que la distancia lateral fuera mínima, y que la separación entre las distintas filas fuera mayor para no ocasionar fatiga visual al usuario.

Figura 9.10: Diseño de la página de galería en Figma

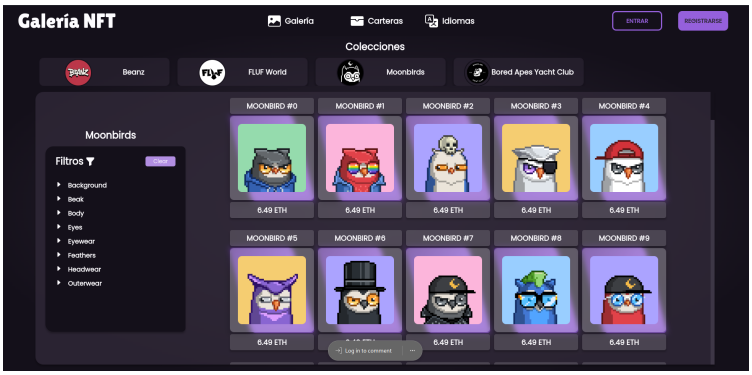



**Fuente:** Proyecto de Figma para el TFT

Otras de las funcionalidades por las que se ha decidido realizar esta página es por las carteras. En este caso se ve que hay algunas diferencias entre el diseño [Figura 9.12] y el desarrollo final [Figura 9.13]

Se pueden ver algunas diferencias:

Figura 9.11: Página de galería desarrollada y desplegada en Vercel




**Fuente:** Página desarrollada para el TFT [https://tft-galeria-nft-web3.vercel.app/gallery]

- El apartado de filtros no se vio necesario. Esto es debido a que la cantidad de carteras que una persona podría querer guardar es relativo, pero probablemente no llegue a más de 15, por lo que realmente no es tan necesario un filtro en este apartado.

- Otro de los cambios más grandes que se pueden ver es la forma de insertar una nueva cartera, ya que esto no se visualizó a la hora de hacer el diseño.

  - Se decidió crear una tarjeta, como la tarjeta de las carteras, pero con la capacidad de crear un modal para que el usuario inserte los datos.

  - Este contiene un icono, como antes se mencionó, para que el usuario entienda sin leer el texto, para qué sirve esta tarjeta.

  - También contiene el texto que designa la acción de tal tarjeta, para diferenciarla de las otras.

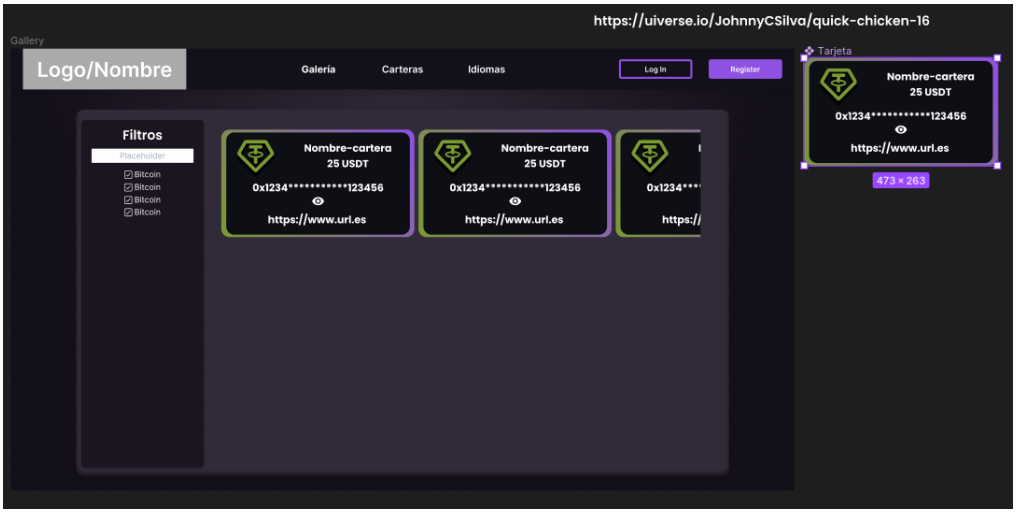
- Otra de las cosas que no se pensó a la hora de hacer el diseño es cómo editar y cómo eliminar una cartera. Para seguir aprendiendo distintas formas de ejecutar acciones en la pagína, se decidió utilizar el clic derecho del ratón.

  - Esto genera un menú contextual, donde podemos editar y eliminar la cartera, también acompañado de iconos.

  - Sólo se genera al hacer clic derecho sobre las tarjetas, ya que la acción corresponde con la propia cartera, no con el conjunto.

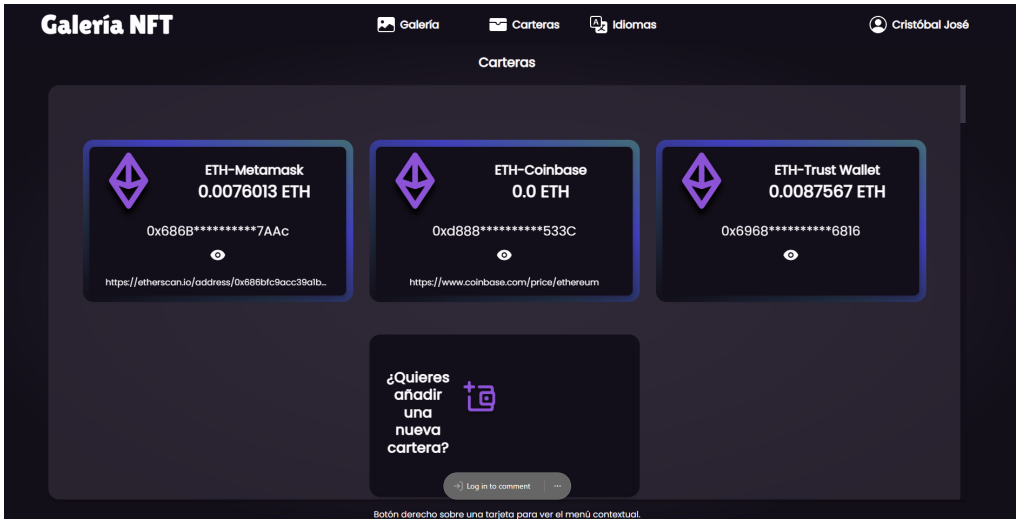
- Como esta función podría no ser obvia para el usuario, se insertó un pequeño mensaje como *footer* donde se pone "Botón derecho sobre una tarjeta para ver el menú contextual."

Figura 9.12: Diseño de la página de carteras en Figma




**Fuente:**Proyecto de Figma para el TFT

Figura 9.13: Página de carteras desarrollada y desplegada en Vercel



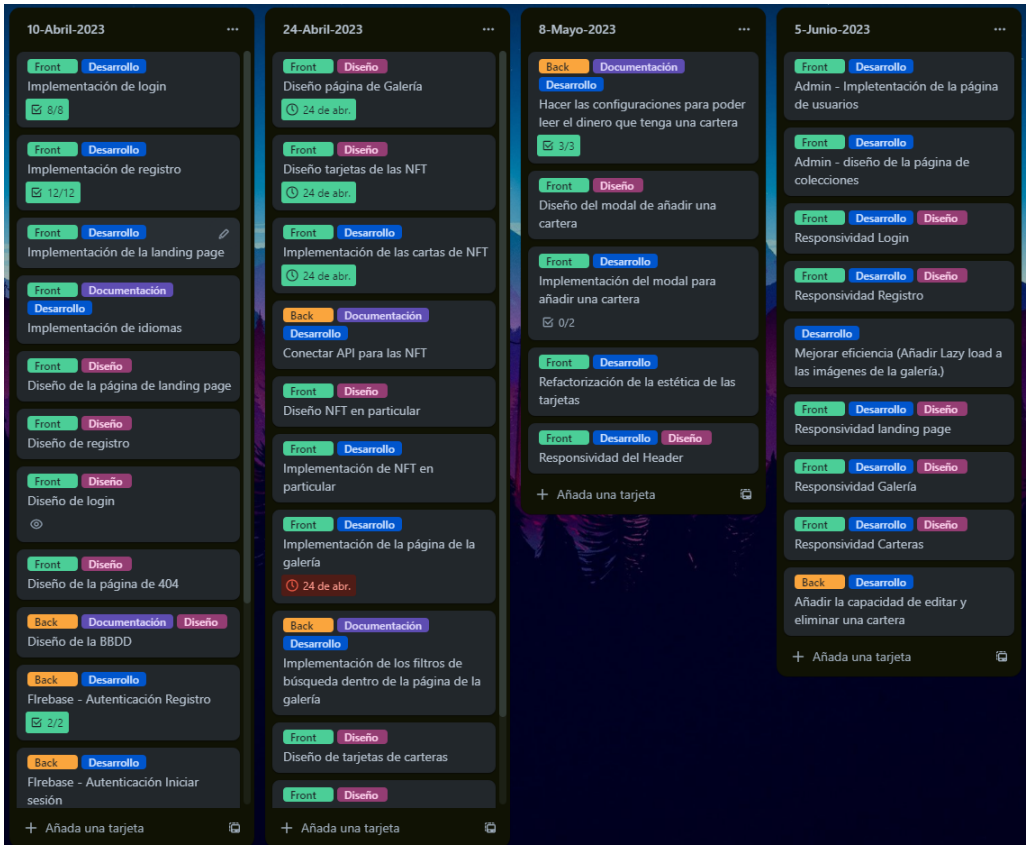

**Fuente:**Página desarrollada para el TFT

## 9.7. Desarrollo y explicación

Debido al uso de Scrum, la parte de desarrollo se ha decidido explicar en función de los distintos *sprints*. No obstante, solo se explicará el final de cada apartado.

Las historias de usuario y tareas han quedado en el final del proyecto tal y como se ve en la figura 9.14 haciendo referencia cada calle al final de un sprint.

Figura 9.14: Tablero de Trello al final del proyecto




**Fuente:** Cuenta personal de Trello

### 9.7.1. Sprint 0

Este al ser el primer Sprint del Trabajo, se hizo una estructuración de la arquitectura del proyecto, análisis de otras páginas [apartado 8.1], así como los esquemas [apartado 8.2] necesarios para estructurar el trabajo de una forma adecuada.

### 9.7.2. Sprint 1

En este Sprint se empezó con el diseño y desarrollo de algunos componentes, como son el *header*, la *landing page* y las páginas de Inicio de Sesión y

Registro.

#### 9.7.2.1. Header

Algo a destacar sobre la programación de esta aplicación web, es la capacidad de Angular de ejecutar una SPA (*Single Page Application*), gracias a esto, el *header* solo se *renderiza* una única vez al inicio de la página [Figura 9.16], y se mantiene constante, lo que cambia en las páginas es el contenido que se encuentra inmediatamente debajo del *Header*.

Figura 9.15: Simplificación del funcionamiento de index.html

```
<body>
    <app-header></app-header>
    <router-outlet></router-outlet>
</body>
```

**Fuente:** Trabajo desarrollado

El *Header* tiene una estructura típica, donde a la izquierda se encuentra el ”logotipo”de la página, en el centro encontramos una barra donde podremos navegar por las distintas páginas, seleccionar el idioma, y a la derecha encontraremos los botones de inicio de sesión y registro, o en caso de ya haber iniciado sesión, encontraremos nuestro nombre de usuario.

Como se mencionó en el apartado 9.5, todos los textos vienen acompañados de iconos con la finalidad de ofrecer al usuario un mayor *confort* dentro de la página.

Figura 9.16: Simplificación del código del *header*

```
<header class="structure">
    <!-- Izquierda - Logo -->
    <a routerLink="/home">Galería NFT</a>

    <!-- Centro - Botones de la cabecera -->
    <nav class="buttons">
        <!-- Botón galería -->
        <a routerLink="/gallery"> <div> <i class="gallery"></i>Galería </div> </a>
        <!-- Botón cartera -->
        <a routerLink="/wallets"> <div> <i class="wallets"></i>Carteras </div> </a>
        <!-- Dropdown idioma -->
        <div> <a> <i class="languages"></i>Idiomas </a> </div>
    </nav>

    <!-- Derecha - botones o usuario -->
    <div class="right">
        <div *ngIf="user!=null" class="dropdown profile">
            <div> <i class="User-icon"></i> <p>{{user.getName()}}</p> </div>
        </div>

        <div *ngIf="user==null">
            <button (click)='openLogin()'> Inicia Sesión </button>
            <button (click)='openRegister()'> Regístrate </button>
        </div>
    </div>
</header>
```

**Fuente:** Simplificación del funcionamiento de index.html

#### 9.7.2.2. Landing Page

Una *landing page*, también conocida como página de aterrizaje, es una página web diseñada específicamente para captar la atención de los visitantes y convertirlos en clientes potenciales o realizar una acción específica y tiene como objetivo principal guiar a los visitantes hacia una acción deseada.

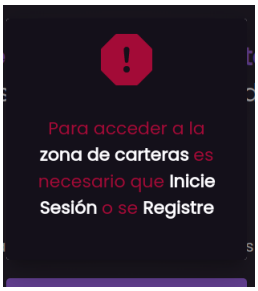
En este caso, la función principal de esta sería que el usuario sea consciente de lo que es la página, y lo que se puede realizar en la misma, esto se pretende gracias a la explicación que se muestra en pantalla. Además, los botones están diseñados con la intención de destacar la capacidad del usuario para registrarse, así como intentar dejar claro también que se puede iniciar sesión. Es la denominada *llamada a la acción* o *call to action*.

Otra de las cosas que se pueden realizar desde esta página es ir a la galería y, en caso de estar registrados, también se podrá acceder a la pestaña de carteras. En el caso de no estar registrado e intentar entrar a la pestaña de carteras, nos devolverá un diálogo [Figura 9.33] indicando al usuario que para entrar a este apartado debe iniciar sesión o registrarse.

Figura 9.17: Landing page


**Fuente:** https://tft-galeria-nft-web3.vercel.app/home

Figura 9.18: Diálogo donde no permite al usuario acceder al apartado de carteras




**Fuente:** https://tft-galeria-nft-web3.vercel.app/home

#### 9.7.2.3. Modal de Inicio de Sesión y Registro

En un inicio, se pensó en realizar tanto la pantalla de Inicio de Sesión, como la página de Registro como páginas separadas.

En el Sprint 3 (Apartado 9.7.4), se rectifico esto debido al diseño elegido y a la facilidad que nos aporta Angular Material, y se realizaron modales de Inicio de Sesión y de Registro para facilitar a la hora de la programación, ya que no se tendría que realizar una nueva página solo para estos casos. Simplemente se desarrolló un servicio ”modal”[Figura 9.26], el cual usa la librería de Material Dialogs. Este servicio nos aporta una interfaz para poder generar modales con cualquier componente que creemos.

Tanto en el Inicio de Sesión [Figura 9.19] como en el Registro [Figura

9.20], se hace referencia a una de las reglas de Oro de Ben Schneiderman [24]. **Ofrezca un manejo simple de errores**, por lo que se decidió realizar un manejo de errores de forma que el usuario vea al instante si los datos que pretende introducir son correctos o no. Además, se bloquea el botón de enviar para notificar al usuario de que faltan datos a rellenar.

Figura 9.19: Modal de Login


**Fuente:** https://tft-galeria-nft-web3.vercel.app/home

Figura 9.20: Modal de Registro


**Fuente:** https://tft-galeria-nft-web3.vercel.app/home

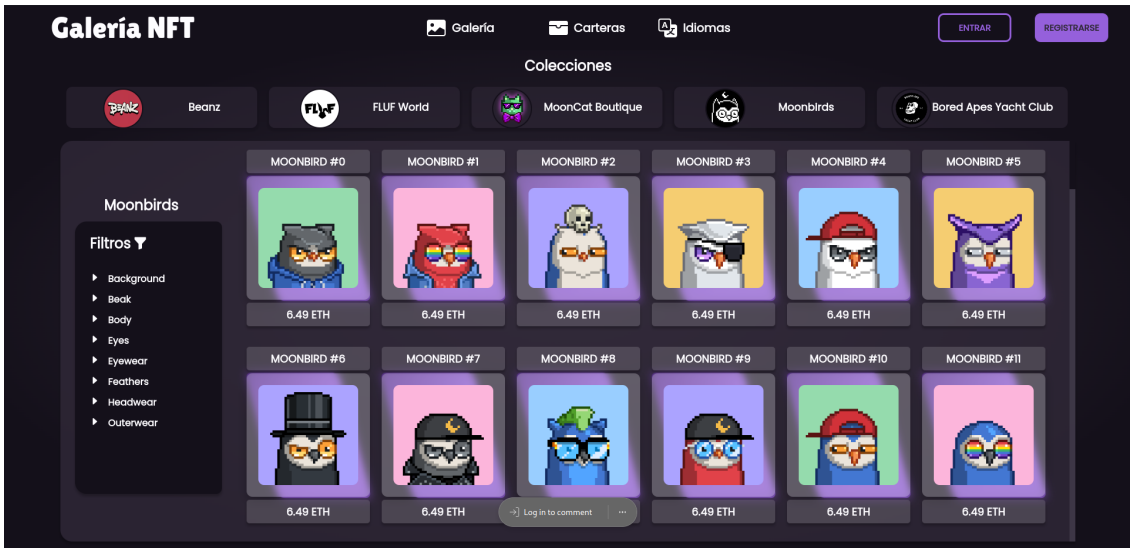
### 9.7.3. Sprint 2

Este Sprint se basó, en su gran mayoría, en entender el funcionamiento de la API de Alchemy, diseñar y desarrollar la página de la galería, diseñar y desarrollar las cartas de los distintos NFTs y realizar los filtros para la búsqueda de los mismos. Y como se terminó antes de tiempo, se decidió empezar a diseñar la página de carteras y a desarrollar la estructura de esta.

#### 9.7.3.1. Galería

En la página de la galería se encuentran distintas funcionalidades, entre ellas, podemos cambiar de colección *clicando* en las tarjetas que se encuentran en la parte superior (Esta parte no se había gestionado desde un inicio, y surgió el problema de no poder cambiar de colección el cuál se solucionó una vez las funcionalidades básicas ya estaban desarrolladas [Apartado 9.7.6]).

Figura 9.21: Landing page




**Fuente:** https://tft-galeria-nft-web3.vercel.app/gallery

Otra de las funcionalidades es filtrar los distintos NFTs. Los filtros vienen dados por los distintos NFTs, de forma que la lista de filtros se va actualizando a medida que se van obteniendo los NFTs desde la API [Figura 9.22]. Este apartado tuvo muchos fallos desde un inicio con demasiados filtros. Al acabar las funcionalidades se continuó con la mejora de los filtros para un correcto funcionamiento [Apartado 9.7.6].

Figura 9.22: Fichero de configuración de Vercel

```
async getNFTs() {
    await this.nft.getNFTs(this.address).then(data => {
      for (let item of data) {
        this.push_data_in_NFT(item);

        if (item.attributes != undefined)
          for (let att of item.attributes)
            this.push_attribute(att);
      }
    })
  }
```

**Fuente:** Código reducido para mayor comprensión

### 9.7.4. Sprint 3

En este Sprint conllevó la mayor parte del proyecto, ya que al seguir estudiando y no disponer de tiempo para desarrollar el conjunto de tareas, se adaptaron las 2 semanas de los *sprints* habituales a 1 mes. En este, se terminó la función de crear carteras, pero al pedir *feedback* se encontró un problema de accesibilidad el cual hizo necesaria una *refactorización* en el desarrollo de la pantalla y de las cartas de las carteras.

En adición, se encontró el problema de que sólo se era capaz de pedir a la API de Moralis las carteras que residen en la red Ethereum, por lo que cualquier otra red nos devolvía números incorrectos, ya que la red podía existir, pero no pertenecía al usuario.

Además, y debido a diversos fallos con enlaces, se decidió hacer una página de *Error 404: página no encontrada*, la cual serviría también como acceso para enviar mensajes de *feedback* a los administradores de la página.

#### 9.7.4.1. Carteras

Esta es una funcionalidad que no he llegado a encontrar en internet, la cual veo bastante útil para el caso que se ve en la figura 9.13, ya que se pueden introducir diversas carteras de distintas páginas para tenerlas todas juntas.

Aquí podemos encontrar que la estética final de las carteras [Figura 9.25] nos muestra:

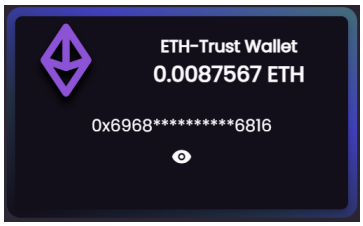
- En la esquina izquierda superior el icono de la Criptodivisa
- En la esquina derecha superior el nombre que se le ha puesto a la cartera

y la cantidad de criptomonedas que se tiene en la respectiva cartera.

- En el centro se observa la dirección de la cartera, la cual está minimizada, y *clicando* el ojo que se encuentra debajo se puede observar la dirección completa.

- En caso de haber insertado una URL para poder ir a la cuenta donde se ubica la cartera, se verá esta dirección para poder clicar y ser *redireccionado*. En caso de no existir, esta parte se omite de la tarjeta.

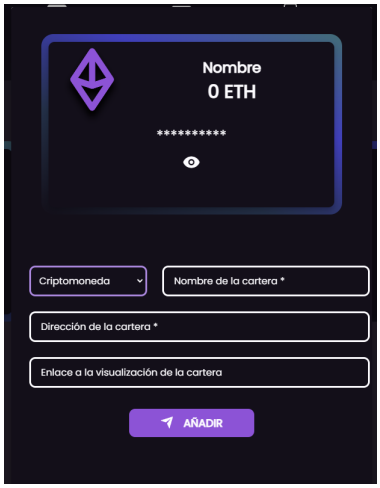
Figura 9.23: Componente de cartera




**Fuente:** Página desarrollada para el TFT

Se realizó un modal [Figura 9.24] para poder añadir y editar estas tarjetas. Este proporciona una *previsualización* de la tarjeta, para ver cómo quedará, así como un formulario para rellenar los datos de la cartera que se quiere añadir/editar.

Figura 9.24: Modal añadir/editar tarjeta




**Fuente:** Página desarrollada para el TFT

Se realizaron las funciones pertinentes para cada acción (añadir, editar y eliminar carteras), pero, al igual que para la elección de colecciones (Apartado

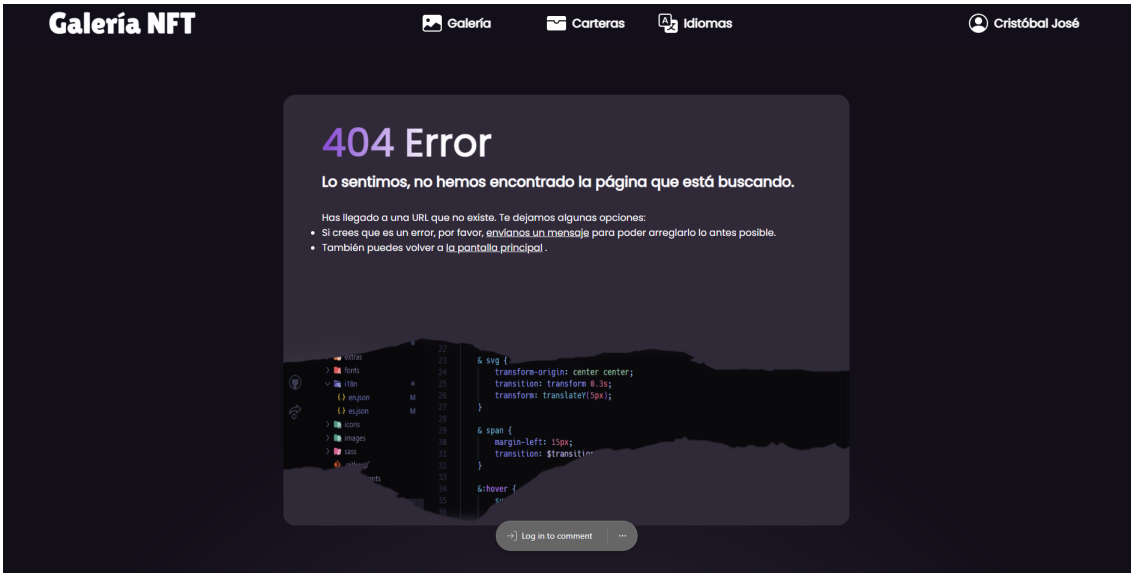
9.7.3.1) no se hizo una interfaz para aplicarlas. Como se verá en el Apartado 9.7.6, también se desarrolló un menú contextual para estas acciones.

#### 9.7.4.2. 404: Página no encontrada

Esta página, como su nombre dice, se utiliza para redirigir en caso de que la URL a la que se está intentando acceder no exista. Esta página se creó debido a un problema en el apartado de los NFTs, ya que al intentar *clicar* sobre ellos no *redireccionaba* a su respectiva página, sino que nos redirigía a la *landing page*, para que no pareciera una falla, de desarrolló esta página en caso de no encontrar una URL válida.

A parte de existir con la intención antes nombrada, se hizo por si debería existir una URL la cual no es accesible en la actualidad, y poder enviar desde aquí un mensaje a los administradores de la página con un mensaje de *feedback*.

Figura 9.25: Página de Error 404: Página no encontrada




**Fuente:** Página desarrollada para el TFT

#### 9.7.4.3. Servicio de Diálogos modales

Se encontró la necesidad de reducir el número de URLs de la página, y en vez de tener una página para el Registro del usuario, otra para el Inicio de Sesión del usuario y otra para la creación de tarjetas, se pensó que una mejor manera de desarrollar esto era haciendo diálogos modales.

Angular Material [Apartado 6.2] nos aporta una librería que se pudo utilizar para esta funcionalidad, la cual se usa como interfaz para crear modales [Figura 9.26] a partir de un componente pasado como parámetro. Además, y para hacer cada modal único, también se pueden pasar los distintos parámetros como el alto, el ancho, y algunos datos que pueden servir de ayuda a la hora de generar el modal.

Figura 9.26: Servicio de Modales

```
@Injectable({ providedIn: 'root' })
export class ModalService {
  constructor(protected dialog: MatDialog) { }
  openDialog(component: any, heigh: string, width: string, data: any) {
    this.dialog.open(component, { height: heigh, width: width, data });
  }
  closeDialog() { this.dialog.closeAll(); }
}
```

**Fuente:** Código reducido para mayor comprensión

En el caso de añadir una cartera, se reutilizó el componente para que pudiera crear y modificar una cartera [Figura 9.27], para crear, se asume que los datos están vacíos, y por lo tanto crea una instancia de cartera nueva. Para editar una cartera, se pasa por parámetros la cartera que se desea modificar, y el modal recoge los datos, se clona la cartera, para que no se cambien los datos originales, y se actualizan los datos en Firebase

Figura 9.27: Obtención de datos en el modal de Añadir/Editar cartera

```
  constructor(private modal: ModalService, private db: FirestoreService,
          @Inject(MAT_DIALOG_DATA) public data: any) {
  if (data.wallet) {
    let w: Wallet = data.wallet;
    //Nombre anterior - identificador en BBDD
    this.nameBeforeChange = w.getName();
    this.future_wallet = w;
  } else this.future_wallet = new Wallet("Nombre", "", 0, "URL", "ETH");

  //Se pasa por parámetros si es para editar o no
  if (data.edit) this.editing = true;
}
```

**Fuente:** Código reducido para mayor comprensión

### 9.7.5. Sprint 4

Este Sprint pretendía ser el último del proyecto, ya que teniendo todas las pantallas y funcionalidades mínimas, el proyecto estaría terminado.

En este se realizaron los diseños y desarrollos de las pantallas de de administrador, para poder ver los usuarios registrados, y poder modificar las colecciones que se podrán ver en la galería. También se empezó con la *responsividad* de alguna de las pantallas en los tamaños más grandes. Así como aplicar el *feedback* recibido de las personas que probaron la página, mejorando en esta la experiencia de usuario y la eficiencia de la misma.

#### 9.7.5.1. Pantallas de Administrador

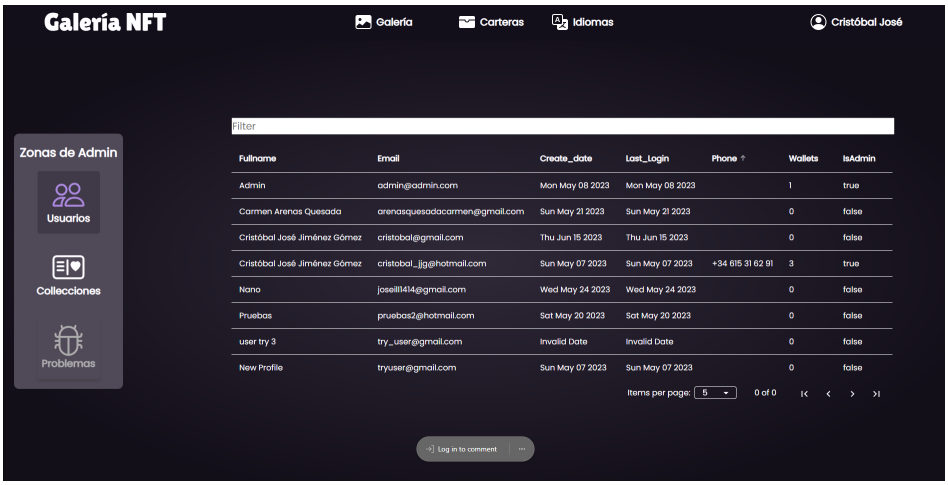
Para la zona de administradores se decidió tener los 3 conceptos básicos que se han mencionado a lo largo de todo el informe:

- Los usuarios
- Las colecciones de NFTs
- Los mensajes de *Feeback*

Se decidió realizar un apartado donde ver la información básica de los usuarios que se han registrado en la página:

- El nombre completo del usuario
- El correo del usuario
- Cuándo se creó la cuenta
- La última vez que inició sesión
- Su número de móvil
- El número de carteras que ha insertado
- Y ver si es administrador

Figura 9.28: Pantalla Administrador sección de Usuarios

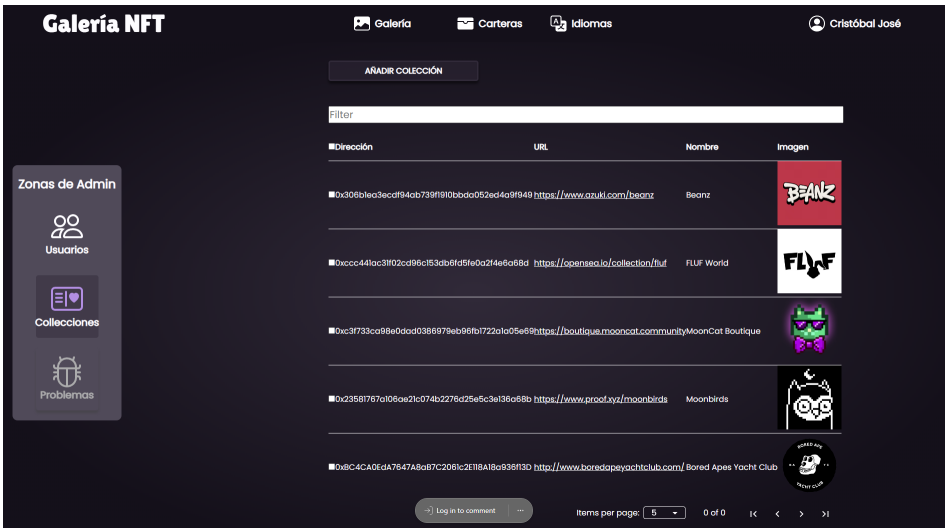



**Fuente:** Página desarrollada para el TFT

En el apartado de las colecciones podemos ver dos funcionalidades básicas, las cuales son ver las colecciones que se encuentran disponibles y añadir colecciones para que puedan ser vistas en la página de colecciones.

- Un apartado para seleccionar varias colecciones
- La dirección de la colección
- La URL a la página oficial
- El nombre de la colección
- La imagen que se verá de la colección

Figura 9.29: Pantalla Administrador sección de Colecciones



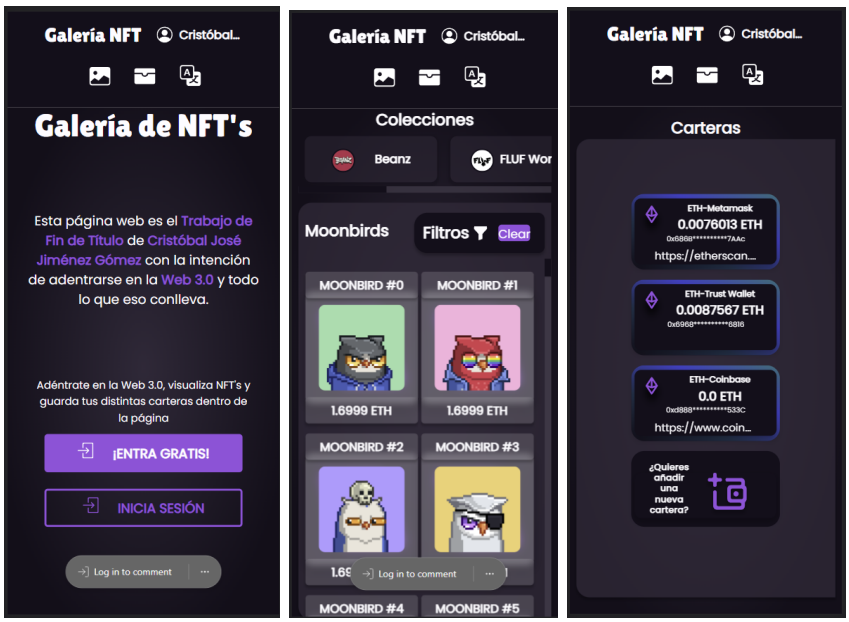

**Fuente:** Página desarrollada para el TFT

### 9.7.5.2. *Responsividad* en las pantallas

Se decidió hacer la página responsiva para aumentar la accesibilidad de la misma, haciendo que esta se pueda ver de forma correcta en diversos dispositivos como son móviles, *tablets*, ordenadores y televisores de mayor resolución.

En este apartado se mostrarán algunos ejemplos:

Figura 9.30: Páginas en versión móvil (Samsung Galaxy S20 Ultra)




**Fuente:** https://tft-galeria-nft-web3.vercel.app/home

### 9.7.6. *PostSprints*

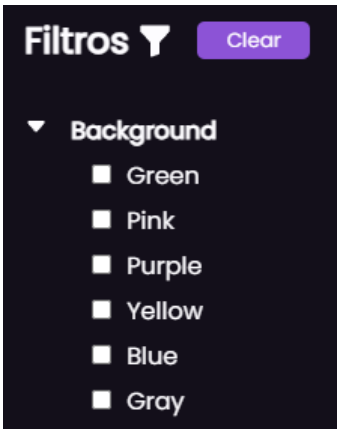
#### 9.7.6.1. Iconos

Uno de los mensajes que más repetían los usuarios que hacían uso de la página, era la necesidad de tener iconos en las acciones principales. Esto se tomó en cuenta, además de seleccionar iconos rellenos, ya que en un principio había una mezcla de iconos tanto rellenos como *outlined*.

Figura 9.31: Iconos en la cabecera


**Fuente:** https://tft-galeria-nft-web3.vercel.app/home

Figura 9.32: Iconos en el filtro de la galería




**Fuente:** https://tft-galeria-nft-web3.vercel.app/home

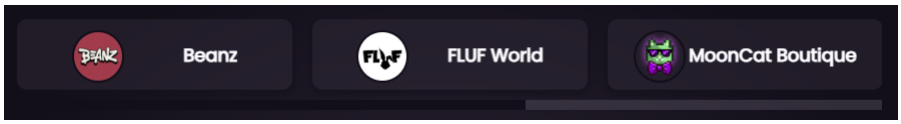
Figura 9.33: Iconos en el apartado de administradores


**Fuente:** https://tft-galeria-nft-web3.vercel.app/home

#### 9.7.6.2. Cambiar entre colecciones

Como se comentó en el apartado 9.7.3.1, esta funcionalidad no estuvo premeditada, por lo que se decidió crear una barra de navegación [Figura 9.34] en la parte superior del contenedor con todas las colecciones que se encuentran en la base de datos, y ahí poder cambiar entre las distintas colecciones.

Figura 9.34: Barra donde se ubican todas las colecciones para poder navegar




**Fuente:** https://tft-galeria-nft-web3.vercel.app/home

#### 9.7.6.3. Mejora de filtros

El apartado de filtros tenía un error, el cual consistía en que si elegíamos 2 o más cualidades del mismo atributo nos devolvía que no existía ningún NFT. Y es verdad, ya que no puede existir un NFT que tenga un fondo ver y un fondo azul, por ejemplo. Esto se entendió, se rectificó y se añadió la funcionalidad de limpiar los filtros, esto hace que todos los *chackbox* se desmarquen, y se limpien las listas de filtros.

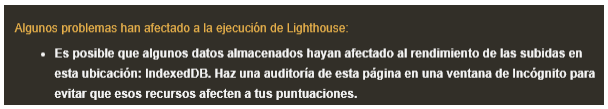
# Capítulo 10

# Resultados y Evaluación

## 10.1. Evaluación y Resultados de Lighthouse

Se realizarán las pruebas con la *landing page* y con la página de la galería. Estas pruebas se harán en un navegador en modo incógnito, tal y como nos dice Lighthouse si intentamos ejecutarlo con un navegador normal.

Figura 10.1: Aviso por parte de Lighthouse




**Fuente:** Test de Lighthouse

Figura 10.2: Resultados de Ligthouse para la página de Home Nueva

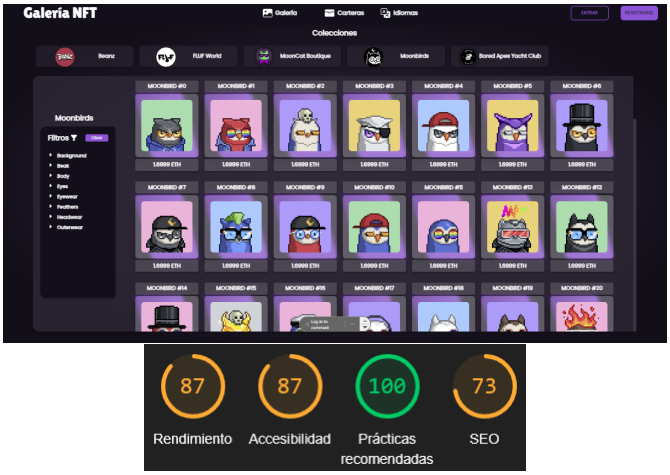

**Fuente:** Test de Lighthouse

En este caso, se observa que ha habido un pequeño **decremento en cuan-**

**to a la accesibilidad** [Figura 10.2]. Podemos observar en este caso, que la página tiene un gran rendimiento, una alta accesibilidad y se siguen las prácticas recomendadas. Con respecto al posicionamiento SEO, los mensajes que muestra el test son errores más orientados a sistemas que a desarrollo. De la información que ofrecen estos informes podemos concluir que los porcentajes son más que aceptables.

### 10.1.1. Galería

Figura 10.3: Resultados de Ligthouse para la página de Galería Nueva




**Fuente:** Test de Lighthouse

Para esta página se priorizó la mejora del rendimiento [Figura 10.2], este se consiguió precargando las imágenes de las colecciones y realizando un *Lazy Load* a las imágenes de los NFTs, de esta forma las imágenes solo se cargan si se muestran en pantalla.

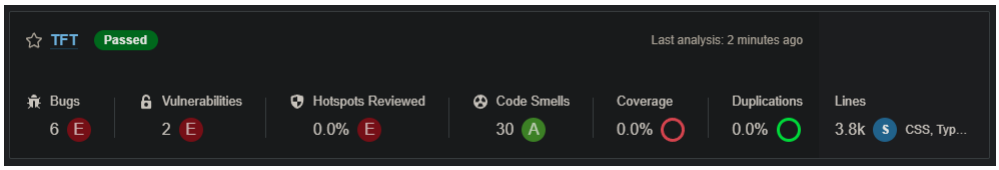
De esta forma, se aumentó en gran medida la eficiencia tanto de carga de imágenes como de obtención de datos al filtrar elementos.

## 10.2. Evaluación y Resultados de SonarQube

Durante el sprint continuo final no se ejecutaron los tests de Sonarqube, debido a que no se hacia un *merge* a la rama principal. En la figura 10.4 se ve lo que nos devuelve al ejecutar el test en esta rama.

A continuación, se harán los cambios pertinentes para que todos los apartados den una nota de A (no obstante, el test está aprobado), y si se pueden mejorar los *Code Smells* ya existentes, se arreglarán.

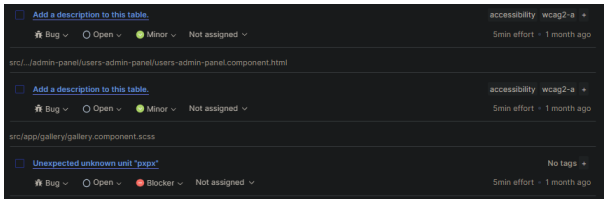
Figura 10.4: Resultados de SonarQube en la última subida




**Fuente:** Test de SonarQube

Se puede comprobar en la figura 10.5, que son *bugs* sencillos de solucionar, por lo que no tendrán mucha complicación.

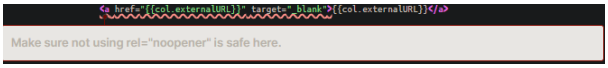
Figura 10.5: Apartado de *Bugs* antes de hacer la subida de la rama




**Fuente:** Test de SonarQube

En el caso de fallas de seguridad [Figura 10.6] nos dice textualmente que el fallo se debe a no colocar una URL exacta. Si la página a la que hace referencia contiene código malicioso, podría usar la puerta trasera (window.opener) de su sitio web para secuestrar el navegador de su usuario y, por ejemplo, redirigirlo a un sitio web de *phishing* peligroso que roba su información o instala *malware* en su computadora[25].

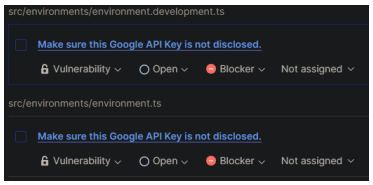
Figura 10.6: Apartado de *Security Hotspot* antes de hacer la subida de la rama




**Fuente:** Test de SonarQube

En el apartado de seguridad [Figura 10.7] siempre se encuentra que no es recomendable tener las *keys* de las APIs en el código, no obstante, están puestas dentro de las variables de entorno, las cuales se usan en Angular para esa misma funcionalidad. Por lo que este apartado podemos asumir que realmente está correcto.

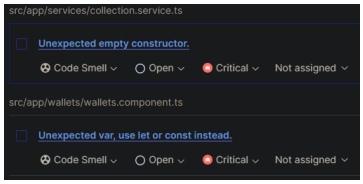
Figura 10.7: Apartado de seguridad antes de hacer la subida de la rama




**Fuente:** Test de SonarQube

Si se revisa el apartado de *Code Smells* se observan 2 problemas calificados como críticos [Figura 10.8]. El resto de *Code Smells* son pequeñas fallas de código, duplicados de atributos en CSS, uso de *var* en vez de *let*, fallos que no afectan a las funcionalidades en sí, pero a futuro puede ocasionar una gran deuda técnica.

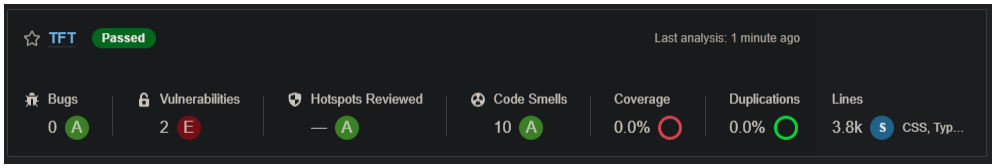
Figura 10.8: Apartado de seguridad antes de hacer la subida de la rama




**Fuente:** Test de SonarQube

En la figura 10.9 se puede ver que se han eliminado los *bugs* y los fallos de *Security Hotspot* y se redujeron el número de *Code Smells*. En este caso, y viendo que el test está aprobado, y se tiene una mejora en cuanto al código, sería un buen momento para realizar una subida del proyecto actual, además de combinarlo con la rama principal.

Figura 10.9: Resultados de SonarQube en la última subida




**Fuente:** Test de SonarQube

# Capítulo 11

# Trabajo futuro y Conclusiones

## 11.1. Trabajo Futuro

Como líneas a desarrollar en futuras iteraciones, se plantean varias que resultan de interés, ya que aportan funcionalidades que facilitan la gestión de la información:

Para la mejora de la gestión por parte de la aplicación.

- Diseñar y desarrollar los formularios que facilitarán la comunicación entre usuarios y administradores para el envío y gestión de sugerencias.
- Ofrecer la posibilidad de que cada usuario pueda gestionar su perfil.
- Formatear los mensajes que ofrecen información al usuario, cambiando así los actualmente proporcionados por angular material.
- Completar el proyecto i18n.
- Diseñar y desarrollar un panel de administración para gestionar servicios como los distintos perfiles de usuarios.

En aras de aumentar las funcionalidades del servicio de carteras.

- Facilitar el acceso a la información, tanto en detalle como agrupada, de cada una de las carteras de los usuarios haciendo uso de menús.

- Mostrar información de cambio de criptodivisas.

Un aspecto interesante de este tipo de aplicaciones es el análisis de información de uso como: accesos, tiempos, o comportamiento de los usuarios. Aunque la aplicación puede ofrecer un panel de control que muestre estadísticas del tráfico, en el estado actual del proyecto el uso de Google Analytics es suficiente para obtener informes de seguimiento y analizar esta información.

## 11.2. Conclusiones

El principal objetivo de este Trabajo de Fin de Grado es el diseño y desarrollo de una galería de NFTs en el marco de la denominada Web3. Se han alcanzado los diversos objetivos propuestos: los usuarios registrados podrán acceder a distintas colecciones de NFTs, y añadir carteras que podrán gestionar desde un solo sitio; por otro lado, los usuarios administradores podrán gestionar colecciones y realizar consultas sobre los usuarios registrados.

A nivel técnico, el trabajo me ha servido de ayuda, no solo para avanzar en el área de desarrollo web, también en la búsqueda de estrategias para la optimización de los filtros, para mejorar el rendimiento de las aplicaciones, o conocer cómo hacer un uso adecuado de las librerías que ofrece Angular Material.

A nivel personal, he conseguido desarrollar una herramienta desde la que, de forma resumida, es posible acceder a las colecciones de NFT's que a los usuarios les resulten más atractivas. La herramienta también facilita la gestión de las distintas carteras de criptomonedas seleccionadas por los usuarios, y acceder a sus páginas. Esta última característica hace que la aplicación funcione como una capa superior de acceso a carteras que se encuentran alojadas en distintos sitios web, de forma similar al funcionamiento de una interfaz.

El desarrollo de este proyecto, desde su planteamiento inicial hasta la consecución de los objetivos inicialmente trazados, haciendo uso de las tecnologías estudiadas para el mismo y de metodologías ágiles, dentro de los plazos y con los recursos inicialmente establecidos, es el resumen de lo que ha supuesto este Trabajo Fin de Grado. Lo puedo definir como la materialización de cómo aplicar las competencias adquiridas a lo largo de estos últimos años académicos en las asignaturas cursadas en la titulación.

# Bibliografía


[1] IBM. ¿tecnología blockchain? https://www.ibm.com/es-es/topics/blockchain. [Online; accedido a 30/April/2023].

[2] BBC News Mundo. Qué son los nft y por qué están valorados en millones de dólares. https://www.bbc.com/mundo/noticias-56502251, March 2021. [Online; accedido a 01/May/2023].

[3] Moralis. What is a web3 wallet? – web3 wallets explained. https://moralis.io/what-is-a-web3-wallet-web3-wallets-explained/?ref=morioh.com&utm_source=morioh.com, January 2022. [Online; accedido a 01/May/2023].

[4] Open Bootcamp. ¿qué es un framework y qué tipos hay? https://open-bootcamp.com/aprender-programar/que-es-un-framework. [Online; accedido a 01/May/2023].

[5] Amazon Web Service. ¿qué es una api? https://aws.amazon.com/es/what-is/api/. [Online; accedido a 28/06/2023].

[6] CERN. A short history of the web. https://home.cern/science/computing/birth-web/short-history-web. [Online; accedido a 02/May/2023].

[7] Heather Hall. Web 2.0 explained: Everything you need to know. https://history-computer.com/web-2-0/, November 2022. [Online; accedido a 02/May/2023].

[8] Sean Yang and Max Li. Web3.0 data infrastructure: Challenges and opportunities. https://ieeexplore.ieee.org/document/10110018, April 2023. [Online; accedido a 02/May/2023].

[9] Sebastián Heredia Querro. Smart contracts: Qué son, para qué sirven y para qué no servirán? https://deliverypdf.ssrn.com/delivery.php?ID=400020126087064111116088082069127010096068026065069063076084016102113025
EXT=pdf&INDEX=TRUE, 08 2021. [Paper; accedido a 28/06/2023].

[10] Nelson Rodriguez. 35+ ejemplos de web 3.0 y como blockchain está cambiando la web. https://101blockchains.com/es/web-3-0/, 12 2018. [Online; accedido a 12/05/2023].

[11] Binance. ¿qué es el staking? https://academy.binance.com/es/articles/what-is-staking, 04 2023. [Online; accedido a 13/07/2023].

[12] stakingcrypto. Best deal for staking basic attention token. https://stakingcrypto.io/stake/BAT/basic-attention-token. [Online; accedido a 13/07/2023].

[13] Blogdecentralab. Qué es alchemy. https://decentralab.tech/blog/que-es-alchemy/, 03 2023. [Online; accedido a 13/05/2023].

[14] developers.google.com/web. Lighthouse. https://chrome.google.com/webstore/detail/lighthouse/blipmdconlkpinefehnmjammfjpmpbjk?hl=es. [Online; accedido a 13/05/2023].

[15] SonarQube. Sonarqube documentation. https://docs.sonarqube.org/latest/. [Online; accedido a 13/05/2023].

[16] Desarrollo web. Code smell. https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/code-smell/, 11 2022. [Online; accedido a 13/05/2023].

[17] Team Asana. Qué es la deuda técnica y cómo saldarla (con ejemplos). https://asana.com/es/resources/technical-debt, 07 2022. [Online; accedido a 13/05/2023].

[18] Atlassian. Learn git with bitbucket cloud. https://www.atlassian.com/git/tutorials/learn-git-with-bitbucket-cloud. [Online; accedido a 13/07/2023].

[19] Ken Schwaber andJeff Sutherland. *The Scrum Guide The Definitive Guide to Scrum: The Rules of the Game*. 2020.

[20] MAX REHKOPF. Historias de usuario con ejemplos y plantilla. https://www.atlassian.com/es/agile/project-management/user-stories. [Online; accedido a 28/06/2023].

[21] Danae Salinas. ¿qué es una landing page? la guía completa. https://es.wix.com/blog/2021/05/que-es-una-landing-page, 01 2022. [Online; accedido a 27/05/2023].

[22] Kelly. ¿qué es un repositorio en github? https://abrirarchivos.info/tema/que-es-un-repositorio-en-github/#:~:text=Un%20repositorio%20de%20Git%20es%20un%20almacenamiento%20virtual,en%20l%C3%ADnea%20donde%20se%20almacena%20y%20comparte%20c%C3%B3digo., 2023. [Online; accedido a 20/05/2023].

[23] Developer.Mozilla. Intercambio de recursos de origen cruzado (cors). https://developer.mozilla.org/es/docs/Web/HTTP/CORS, 04 2023. [Online; accedido a 23/05/2023].

[24] Interactemos. 1.1 : Las ocho reglas de oro de shneiderman lo ayudarán a diseñar mejores interfaces. https://www.interactemos.com/lessons/1-1-las-ocho-reglas-de-oro-de-shneiderman-lo-ayudaran-a-disenar-mejores-interfaces 06 2023. [Online; accedido a 25/06/2023].

[25] Chrome Developers. Links to cross-origin destinations are unsafe. https://developer.chrome.com/docs/lighthouse/best-practices/external-anchors-use-rel-noopener/, 08 2019. [Online; accedido a 28/06/2023].