

**INSTITUTO TECNOLÓGICO DE BUENOS AIRES – ITBA**  
**ESCUELA DE INGENIERÍA Y GESTIÓN**

# **Creación automática customizable de Smart Contract ERC-721 (NFT)**

**AUTOR/ES:** Petrikovich, Florencia (Leg. N° 58637)  
Hirsch, Gonzalo Miguel (Leg. N° 59089)

**TUTOR/ES:** Dra. Mon, Alicia Laura

**TRABAJO FINAL PRESENTADO PARA LA OBTENCIÓN DEL TÍTULO**  
**DE**  
**INGENIERO EN INFORMÁTICA**

**Lugar:** Instituto Tecnológico de Buenos Aires (ITBA)  
**Fecha:** Agosto 2022

# Índice

<b>1</b>	<b>Introducción</b>	<b>1</b>
<b>2</b>	<b>Estado del Arte</b>	<b>3</b>
2.1	Smart Contracts . . . . .	3
2.2	Non-Fungible Tokens . . . . .	4
2.3	Alternativas Existentes . . . . .	4
<b>3</b>	<b>Extensión y Alcance</b>	<b>6</b>
3.1	Limitaciones . . . . .	6
<b>4</b>	<b>Producto Desarrollado</b>	<b>8</b>
4.1	Especificación de Requerimientos del Proyecto . . . . .	8
4.2	Especificación de Requisitos Funcionales . . . . .	9
4.2.1	Detalle de Requisitos Funcionales . . . . .	9
4.3	Documentación de Interfaces . . . . .	11
4.3.1	Tipos de usuarios identificados . . . . .	11
4.3.1.1	Usuario Casual . . . . .	11
4.3.1.2	Usuario Experimentado . . . . .	11
4.3.2	Interacción . . . . .	12
4.3.3	Interfaz de Acceso . . . . .	12
4.3.4	Documento de Diseño . . . . .	13
4.3.4.1	Sistema de Diseño . . . . .	13
4.3.4.2	Creación de Componentes . . . . .	14
4.3.4.3	Interfaz - Primera Versión . . . . .	15
4.3.4.4	Interfaz - Versión Final . . . . .	17
4.4	Documento de Arquitectura . . . . .	18
4.5	Diseño de alto nivel . . . . .	19
4.6	Diseño de bajo nivel . . . . .	21
4.7	Infraestructura . . . . .	23

4.7.1	Flujos de Integración Continua . . . . .	26
4.8	Seguridad . . . . .	26
4.9	Analítica . . . . .	27
<b>5</b>	<b>Desarrollo</b>	<b>28</b>
<b>6</b>	<b>Resultados</b>	<b>30</b>
6.1	Tipos de Pruebas . . . . .	30
6.1.1	Pruebas de UX . . . . .	30
6.1.2	Unit testing . . . . .	30
6.1.3	Análisis Estático . . . . .	30
6.2	Pruebas de UX . . . . .	30
6.2.1	Casos de Prueba . . . . .	31
6.2.2	Resumen de Resultados . . . . .	33
6.2.3	Resultados de Pruebas . . . . .	33
6.2.3.1	Buscar información sobre NFTs . . . . .	33
6.2.3.2	Generar, publicar y verificar un contrato . . . . .	34
6.2.3.3	Buscar explicaciones de campos . . . . .	34
6.2.3.4	Guardar ID del contrato . . . . .	35
6.2.3.5	Interactuar con contrato y generar un token . . . . .	35
6.2.3.6	Buscar explicaciones sobre métodos y parámetros . . . . .	36
6.2.3.7	Visualizar tokens creados . . . . .	36
6.2.3.8	Verificar estado del servicio . . . . .	37
6.2.3.9	Volver a la aplicación y editar contrato . . . . .	37
6.2.4	Resultados de Formulario . . . . .	38
6.3	Unit Testing . . . . .	38
6.3.1	Resumen de Resultados . . . . .	39
6.3.2	Resultados . . . . .	39
6.3.2.1	Creación del contrato . . . . .	39
6.3.2.2	Manejo de Metadata . . . . .	40
6.3.2.3	Manejo de la entrada de los métodos del contrato . . . . .	41
6.4	Analísistico . . . . .	41
6.4.1	Resultados . . . . .	42

<b>7 Aplicación Desarrollada</b>	<b>43</b>
7.1 Selección de Opciones . . . . .	43
7.2 Crear un Contrato . . . . .	44
7.3 Editar un Contrato . . . . .	45
7.4 Definir Metadata . . . . .	46
7.5 Copiar o Descargar el Contrato . . . . .	47
7.6 Publicar un Contrato . . . . .	48
7.7 Verificar un Contrato . . . . .	49
7.8 Enviar un Recordatorio . . . . .	51
7.9 Interactuar con Contrato . . . . .	52
7.10 Ejecutar un Método . . . . .	53
7.11 Crear un Token . . . . .	55
7.12 Visualizar Tokens . . . . .	56
7.13 Detalles de un Token . . . . .	57
7.14 Buscar más Información . . . . .	57
7.15 Verificar Estado del Servicio . . . . .	58
7.16 Visualización de Analítica . . . . .	59
7.16.1 Google Analytics . . . . .	59
7.16.2 Google Search Console . . . . .	60
<b>8 Conclusiones y Trabajo Futuro</b>	<b>61</b>
<b>9 Referencias</b>	<b>63</b>

# 1 Introducción

Desde el inicio de los tiempos, un punto que siempre ha causado conflictos en la historia humana es la posesión de los recursos, es decir, quién es dueño de que. En la actualidad, la emisión de documentos que garantizan la propiedad de un recurso suele requerir de la intervención de un escribano para aportar su sello de validez a los mismos. Estos documentos incluyen títulos de propiedad de inmuebles o títulos de propiedad automotor, entre otros. Es decir, es necesario un tercero en toda clase de operaciones que involucren la propiedad de un recurso, principalmente en lo que respecta a recursos físicos. Se puede aplicar el mismo concepto a recursos no físicos, como la propiedad intelectual. Por otro lado, adquirir los servicios de un escribano agrega un plano de complejidad extra a operaciones de la índole mencionada. Sin mencionar que existe la posibilidad de la falsificación de dichos documentos. Los Non-Fungible Tokens (NFTs) surgieron alrededor de 2012 (Assia, 2012; Dormehl, 2021) como una idea de mejora para Bitcoin, mientras que a principios de 2018 se publicó el estándar ERC-721 en los *Ethereum Improvement Proposals* (Entriken & Shirley & Evans & Sachs, 2018) para solucionar los problemas mencionados anteriormente.

Los Non-Fungible Tokens (NFTs) constituyen una unidad de información almacenada en la blockchain, y eso les otorga una prueba de propiedad irrefutable debido a la naturaleza de la tecnología en sí. Se usan para representar la posesión de un recurso único, no intercambiable debido a que posee propiedades únicas. Es decir, uno puede cambiar 1 Ethereum por 1 Ethereum y sigue teniendo 1 Ethereum, pero no se puede intercambiar el título de dos propiedades ya que uno se quedaría con una propiedad diferente a la original. Estos NFT se pueden utilizar para representar la posesión del título de una propiedad o el título obtenido en una universidad y emitido por una entidad educativa, efectivamente eliminando la necesidad de entidades que aporten un sello de validez o verifiquen la veracidad de un documento de propiedad. Esto se debe a que, por la naturaleza de la tecnología, solo entidades autorizadas pueden emitir tokens dentro de un contrato y esos tokens no son falsificables. A nivel global se utiliza el estándar ERC-721 de Smart Contracts para los contratos que involucran NFTs y por eso se utilizó dicho estándar para este proyecto.

Uno de los principales problemas que se encuentran actualmente en torno a los NFTs es la poca facilidad para introducirse en el ecosistema como un actor que genera contratos. Es un nicho muy especializado donde es necesario poseer una gran cantidad de conocimientos expertos al respecto. Un problema del que los autores mismos de este trabajo han sido víctimas. Si bien existen aplicaciones que permiten la generación de NFTs, las mismas no se encuentran enfocadas en el contrato, suelen tener una experiencia del usuario que deja mucho que desear o simplemente el usuario necesita tener una gran cantidad de conocimientos al respecto.

El objetivo del presente trabajo es desarrollar una aplicación que permita la creación automática y customizable de contratos inteligentes ERC-721 a través de una interfaz web amigable para facilitar y acortar el proceso de desarrollo de dichos contratos, especialmente para principiantes. Se busca minimizar los requisitos necesarios para crear, publicar e interactuar con contratos inteligentes ERC-721, sin la necesidad de que los usuarios posean conocimientos de programación, billeteras virtuales o inclusive criptomonedas para financiar las operaciones.

Asimismo, la solución desarrollada se enfoca en educar al usuario sobre NFTs y los contratos inteligentes y sus propiedades, proveyendo una experiencia simple y educativa.

## 2 Estado del Arte

Es posible encontrar información sobre Non-Fungible Tokens (NFT) que data de 2012, pero el salto en su popularidad lo dieron a principios del año 2021 (Google Trends, 2020–2022, Julio 1–Agosto 1). Dicha popularidad generó que una gran cantidad de personas se encuentren interesadas en implementarlos. Incontables resultados aparecen al buscar cómo crear un NFT y se dividen en dos principales tipos. El primer tipo son guías que básicamente instruyen a uno a crear una billetera virtual (wallet) y una cuenta en OpenSea (<https://opensea.io/>) o Rarible (<https://rarible.com/>), los mercados digitales de NFTs más conocidos, para luego crear NFTs aislados, sin tener una idea sobre qué ocurre por detrás. El segundo tipo son guías que indican a uno como programar y publicar un Smart Contract para NFTs desde cero. La desventaja de este tipo de guías es que para poder seguir las es necesario tener una billetera virtual y un conocimiento considerable del entorno Web3, NFTs, programación y muchos otros elementos.

Una colección de NFTs o un Smart Contract ERC-721 sin necesariamente buscar vender los tokens, es una tarea complicada. Para poder hacerlo es necesario conocer mucho al respecto, haciendo que adentrarse al mundo de los NFTs del lado de los creadores, teniendo control sobre el contrato, sea extremadamente complicado para un usuario común.

### 2.1 Smart Contracts

Los Smart Contracts son programas que se ejecutan en determinadas direcciones (addresses) en la Blockchain. Esos contratos son un conjunto de datos y código, que representan el estado y funciones para modificar ese estado, respectivamente.

Hay estándares definidos para diferentes tipos de contratos que apuntan a modelar diferentes situaciones. Los estándares más comunes son ERC-20, ERC-777, ERC-1155 y ERC-721 (Ethereum, 2022). El estándar ERC-20 es el utilizado para crear tokens intercambiables, como tokens para votaciones o tokens como monedas, como por ejemplo el Ether. El estándar ERC-777 se utiliza para definir funcionalidades extra a contratos ERC-20 existentes. El estándar ERC-1155 se utiliza para realizar transacciones más eficientes y agrupar transacciones, se puede utilizar para tokens como monedas o para NFTs. Por último se encuentra el estándar ERC-721, el estándar principal para NFTs y se

utiliza para poder identificar a un recurso de manera inequívoca. Este estándar es el que se decidió usar para el proyecto, principalmente por su nivel de adopción en el mercado actual.

## 2.2 Non-Fungible Tokens

Los NFT son Tokens No Fungibles (Non-Fungible Tokens, NFT) y se usan para representar la posesión de un recurso único. Dichos recursos son no fungibles al no ser intercambiables, ya que poseen propiedades únicas. El recurso, o un URI apuntando al mismo, suele estar almacenado en el token en sí. Esa URI puede también apuntar a un conjunto de propiedades extra del recurso, conocidas como metadata, del cual el token representa propiedad. Esas propiedades y/o recursos se suelen almacenar en un *InterPlanetary File System* (IPFS), un sistema distribuido para almacenamiento y acceso de datos. Por ende, la URI suele apuntar a un recurso almacenado en un IPFS.

Solo puede haber un dueño para cada token a la vez y esos tokens se encuentran asegurados por la blockchain.

## 2.3 Alternativas Existentes

Las alternativas existentes se presentan en dos grupos. El primer agrupamiento responde a las soluciones existentes en el momento previo al comienzo del desarrollo de este proyecto que se integran con las siguientes alternativas, *Contracts Wizard by OpenZeppelin* (<https://wizard.openzeppelin.com/>) y *NFT Generator by OneMint* (<https://nft-generator.art/>).

*Contracts Wizard by OpenZeppelin* es una herramienta web gratuita e interactiva para la generación de Smart Contracts (ERC-721 y de otros tipos) que tiene la mayor cantidad de opciones que se aproximan a las propuestas para el proyecto. Permite generar un contrato y luego descargarlo. También permite abrir el contrato en Remix IDE (<https://remix-project.org/>), un editor online para el desarrollo de Smart Contracts en Ethereum. A través de este editor se puede publicar y operar con Smart Contracts. La desventaja de esta solución frente al proyecto desarrollado es que la herramienta se limita sólo a la generación, mientras que delega la responsabilidad de publicar y operar con el contrato al usuario, quién necesita ir a buscar información sobre cómo usar el Remix IDE.

*NFT Generator by OneMint* es una aplicación web enfocada en la parte artística y contenido de los NFTs, por lo que no ofrecen una gran personalización del mismo. No es completamente gratuita, aunque tiene una capa gratuita con límites muy estrictos, por ejemplo no poder guardar y continuar posteriormente. Permite la automatización de la generación del arte para los tokens a través de un editor visual. La desventaja de esta solución frente al proyecto desarrollado es que muchas de las funcionalidades interesantes que podría proveer en términos de los contratos no existen o son pagas. Además, no es muy intuitivo el editor que proveen.

En lo que respecta a alternativas que surgieron luego de comenzar con el proyecto, las principales son *Bueno.Art* (<https://www.bueno.art/>) y *Zero Code NFT* <https://www.zerocodenft.com/>.

*Bueno.Art* es una alternativa con un concepto similar a *NFT Generator by OneMint*, las funcionalidades que ofrece son muy parecidas. No ofrece un editor tan visual como el de *NFT Generator by OneMint* aunque es más intuitivo.

*Zero Code NFT* es la alternativa más similar al presente proyecto. Presenta un enfoque orientado al contenido y al contrato, siendo una solución muy completa. Manejan diferentes redes e incluye publicación y verificación de los contratos. La principal diferencia con el proyecto desarrollado es que no ofrece las opciones de personalización del funcionamiento del contrato ERC-721 que se eligieron para este proyecto y tampoco permite la definición de campos de metadata fijos para los tokens.

No se encontraron soluciones existentes que estén enfocadas en el contrato en sí y que al mismo tiempo ofrezcan una solución en donde realmente no se necesita una configuración previa. No suelen ofrecer un manejo de la metadata integrado en la solución y tampoco una interacción general con el contrato.

## 3 Extensión y Alcance

Respecto a la extensión de la solución desarrollada en el presente proyecto, la principal responsabilidad del sistema es la generación de un Smart Contract siguiendo el estándar ERC-721 y utilizando las librerías de Open Zeppelin, un estándar en la industria, a partir de un formulario completado por el usuario. Dicho formulario, al buscar ser públicamente accesible para todo tipo de usuario, será accesible a través de una aplicación web pública. Dentro del mismo se encuentra la posibilidad de modificar funcionalidades y datos básicos, y al mismo tiempo definir una estructura para la metadata de los tokens que se generarán a partir del contrato mismo.

Por otro lado, el sistema permitiría la interacción con el contrato publicado en una blockchain. Eso implica que debe permitir publicar contratos en una blockchain, verificarlos para asegurarse que el contenido generado es el publicado en efecto y definir una forma de ejecutar diferentes métodos de los mismos. Dentro de la interacción se incluye el manejo de la metadata de los tokens, que por el enfoque del proyecto, debe ser administrado de manera automática y transparente por el sistema. Ese manejo implica el guardado de la metadata en servicios de IPFS, metadata siendo las propiedades e imágenes de cada token. Dado el objetivo del proyecto, es necesario que el desarrollo de las funcionalidades se haga teniendo en cuenta que se espera que su uso sea sin billeteras virtuales ni criptomonedas para pagar transacciones, desde el punto de vista del usuario.

Por último, el sistema podrá escanear la blockchain donde se encuentren los contratos para poder mostrar los tokens generados en cada uno, junto con su metadata. Eso implica tener una conexión a un nodo de la blockchain o algún servicio que provea eso, para poder realizar las búsquedas.

### 3.1 Limitaciones

El proyecto es de índole educativa y académica, por lo que se buscó minimizar los gastos y aprovechar capas gratuitas de los servicios utilizados.

La principal limitación consiste en la blockchain utilizada, dado que las transacciones en las blockchain presentan altos costos para su utilización. En este sentido, el desarrollo se

enfocó en utilizar las Test Nets Rinkeby y Ropsten, como respaldo en caso de que Rinkeby no se encuentre disponible. Al ser Test Nets, las operaciones se pagan con Ethereum de Test Net, que puede ser solicitado gratuitamente por diferentes medios. Las limitaciones de dichas Test Nets son la dificultad para conseguir el Ethereum necesario para operar y la inestabilidad de las mismas en ciertos momentos.

El backend del sistema se encuentra desarrollado en una API REST Serverless y en el caso del proveedor Amazon Web Services (AWS), las ejecuciones de APIs Serverless se encuentran limitadas en 29 segundos. En ciertos casos, las publicaciones de los contratos pueden fallar por exceder dicho límite, en cuyo caso el servicio responde con un error. La decisión de una API Serverless es una para balancear el potencial costo del servicio y poseer un servicio distribuido. Para tener un servicio sin límites de tiempo, es necesario una flota de servidores ejecutando el servicio sin descanso, para no tener un único punto de falla en el sistema. Se optó por un enfoque con límites, pero de escalamiento automático en función del uso y de bajo costo. Se mitigan los errores por exceso de tiempo a través de optimizaciones de código y librerías, de forma que no se encuentran errores a menos que sea un momento de congestión en la blockchain.

La base de datos a utilizar se encuentra limitada tanto en capacidad y RAM, ya que se utiliza una capa gratuita de la misma. Para el desarrollo del proyecto la capacidad es suficiente, pero si el sistema fuera a escalar, requerirá una capa superior del servicio.

Finalmente, como una limitación al funcionamiento, se destaca que al no tener una conexión dedicada con un nodo de IPFS, debido al elevado costo de la misma, pueden ocurrir fallas o grandes tiempos de respuesta al momento de querer recuperar la metadata de un token.

## 4 Producto Desarrollado

### 4.1 Especificación de Requerimientos del Proyecto

Se definieron los siguientes requerimientos del proyecto:

- Una aplicación web (construida con Vue.js, acelerado con Vite) para que los usuarios completen el formulario, visualicen el contrato creado, descarguen sus contratos, deployen el contrato e interactúen con el contrato deployado.
    - Su uso debe ser intuitivo, de forma que los usuarios puedan usarla con naturalidad.
    - Debe ser rápida, consiguiendo puntajes altos en las diferentes pruebas de velocidad.
    - Debe tener buen SEO y accesibilidad, para que sea más probable que aparezca en un resultado de Google y sea accesible para todos los usuarios.
    - La comunicación que realiza con el backend debe ser segura, mediante HTTPS.
  - Un backend (API) construido con TypeScript y la librería Serverless para poder desarrollar una API serverless en API Gateway y AWS Lambda.
    - El mismo se encarga de la generación de contratos on demand en base a templates y el establecimiento de conexión e interacción con los servicios externos, como Infura (<https://infura.io/>) para conectarse con la blockchain y un IPFS, en este caso Pinata (<https://www.pinata.cloud/>), para subir metadata.
    - La comunicación a servicios externos deberá realizarse mediante HTTPS.
    - La API debe implementar medidas de seguridad básicas como por ejemplo filtros de XSS, CORS, no permitir sniffers y otras.
    - Contará con una base de datos noSQL para guardar información relacionada a los contratos.
- \* La comunicación con la base de datos debe ser exclusivamente con la API.

## 4.2 Especificación de Requisitos Funcionales

A continuación se presenta el catálogo de requisitos funcionales:

ID	Nombre
REQ1	Creación de un contrato
REQ2	Configuración y generación de un contrato
REQ3	Compilación de un contrato
REQ4	Publicación (deploy) de un contrato
REQ5	Comunicación/interacción con un contrato
REQ6	Verificación de un contrato
REQ7	Copia y descarga de un contrato
REQ8	Envío de recordatorio de un contrato
REQ9	Edición de un contrato en progreso/ya publicado
REQ10	Visualización de tokens y metadata de un contrato
REQ11	Verificación de estado del servicio
REQ12	Información y ayuda para el usuario
REQ13	Generación de analítica para administradores

**Tabla 4.1:** Catálogo de requisitos funcionales.

### 4.2.1 Detalle de Requisitos Funcionales

A continuación se describen los diferentes requisitos funcionales identificados para el sistema:

- **REQ 1 - Creación de un contrato:** El sistema debe ofrecer al usuario la posibilidad de iniciar la creación de un contrato, otorgándole un contrato vacío para que el mismo comience la configuración. Debe incluir el ID interno del contrato dentro de la información que le retorna y/o muestra al usuario.
- **REQ 2 - Configuración y generación de un contrato:** El sistema debe facilitarle al usuario un formulario para que configure el contrato. El formulario debería contener los datos básicos del contrato (nombre y símbolo), extensiones para activar (Mintable, Auto Increment IDs, Pausable, Burnable, Enumerable, Limited Supply, URI Storage y Unique Storage) y campos de metadata (inclusión de una imagen, nombre de los campos, tipo de los campos y tipo de campo numérico, si aplica). Con los datos previamente mencionados, el sistema debe ser capaz de generar un contrato ERC-721 válido.
- **REQ 3 - Compilación de un contrato:** El sistema debe tomar el contrato

mencionado anteriormente y compilarlo utilizando el compilador de Solidity <sup>1</sup> para generar el bytecode para utilizar en la blockchain.

- **REQ 4 - Publicación (deploy) de un contrato:** El sistema debe poder comunicarse con un nodo de la blockchain (Testnet) para poder realizar una transacción de publicación de contrato utilizando el bytecode generado durante la compilación del mismo. Debe retornar la dirección de publicación del contrato.
- **REQ 5 - Comunicación/interacción con un contrato:** El sistema debe poder comunicarse con un nodo de la blockchain (Testnet) para poder realizar la ejecución de los métodos que el contrato ofrece. Los parámetros de entrada de cada método serán dinámicos, por lo que la forma de ejecución es unificada para todos. Debe retornar un valor o un hash de transacción luego de la ejecución del método, cuando corresponda.
- **REQ 6 - Verificación de un contrato:** El sistema debe poder comunicarse con servicios externos para poder realizar la verificación de un contrato ya publicado en una blockchain. Lo hará mediante Etherscan (<https://etherscan.io/>), un explorador de bloques y visualizador de analítica de Ethereum.
- **REQ 7 - Copia y descarga de un contrato:** El sistema debe ofrecer la posibilidad de copiar el código del contrato creado o descargar un archivo con el código del contrato.
- **REQ 8 - Envío de recordatorio de un contrato:** El sistema debe ofrecer la posibilidad al usuario de enviarse a sí mismo un recordatorio del ID del contrato vía email.
- **REQ 9 - Edición de un contrato en progreso/ya publicado:** El sistema debe ofrecer la posibilidad de continuar con la edición de un contrato que se encuentra en progreso o con uno que ya haya sido publicado. Esto debería ser mediante el ID del contrato mencionado anteriormente.
- **REQ 10 - Visualización de tokens y metadata de un contrato:** El sistema debe poder listar todos los tokens creados con la versión publicada de un contrato, y debe poder visualizar la metadata (datos e imagen) de cada uno. También debe

---

<sup>1</sup>Lenguaje de programación utilizado para implementar Smart Contracts.

incluir hipervínculos para visualizar al token y su metadata en servicios externos.

- **REQ 11 - Verificación de estado del servicio:** El sistema debe ofrecer al usuario una página para poder verificar el estado del servicio. El estado incluye si el backend funciona o no, a que blockchain (Testnet) se están publicando los contratos y el balance de la billetera con la que opera.
- **REQ 12 - Información y ayuda para el usuario:** El sistema debe ofrecer al usuario información general sobre todo lo que él mismo realiza con la aplicación y ayuda durante el uso del mismo.
- **REQ 13 - Generación de analítica para administradores:** El sistema debe ser capaz de generar datos de analítica para los administradores en términos de acceso, posicionamiento y uso del sistema en sí.

## 4.3 Documentación de Interfaces

### 4.3.1 Tipos de usuarios identificados

#### 4.3.1.1 Usuario Casual

Este usuario es uno que no utiliza la aplicación con regularidad, y tampoco tiene experiencia en el entorno de los contratos inteligentes. Por ende, necesita que la interfaz sea amigable y que las opciones contengan pequeñas explicaciones. Se espera que este tipo de usuario explore las diferentes opciones de creación de contratos que la aplicación web le ofrezca ya que no sabe lo que busca, aprendiendo así en el proceso. Se considera que este tipo de usuario es el más común para la aplicación.

#### 4.3.1.2 Usuario Experimentado

Este usuario es uno que tiene experiencia en el desarrollo de contratos ERC-721. Se espera que utilice la aplicación si quiere agilizar el proceso de creación y testeo de un contrato sencillo. La aplicación podría reemplazar a este tipo de usuario, pero no es el objetivo, ya que probablemente en un ambiente profesional se busca poder customizar aspectos muy específicos del contrato que la aplicación no ofrece.

### 4.3.2 Interacción

Los usuarios tienen 3 opciones principales, crear/modificar un contrato, interactuar con un contrato ya existente o ver los tokens creados con el contrato. Al entrar a la aplicación, se encuentran con estas 3 opciones.

**Crear/modificar:** Si el usuario elige la opción de crear un contrato, se le presentan las opciones de crear uno nuevo, o seguir la creación a partir del identificador proporcionado. En ambos casos se dirige a la pantalla de creación, con los campos prepopulados en caso de que se elija editar. El usuario luego va a interactuar con el formulario para la generación, y una vez generado se le ofrecen las opciones para descargar, deployar y verificar, además de otras opciones e hipervínculos para visualizar el contrato en aplicaciones de terceros.

**Interactuar:** Si se elige la opción de interactuar con un contrato ya existente, el usuario puede, con un identificador del contrato, buscar al contrato que generó e interactuar con el mismo llamando a los métodos del contrato en sí.

**Visualizar:** Si se elige la opción de visualizar los tokens generados con un contrato, se puede, con un identificador del contrato, buscar al contrato que generó y ver todos los tokens generados.

### 4.3.3 Interfaz de Acceso

Las principales características de las interfaces es que deben ser intuitivas y amigables, se espera que usuarios no experimentados usen la aplicación, por lo que debe ser lo suficientemente simple para que esos usuarios puedan entenderla y navegar con facilidad. Esto se puede lograr de varias maneras, pero lo principal que se tiene en cuenta es no presentar muchas opciones a los usuarios al mismo tiempo, de forma que se los puede guiar a medida que se va usando. Además, esto se alcanza mediante una simple navegación que solo presente las opciones más importantes, y botones y acciones llamativas para los usuarios.

Sin embargo, se espera que el usuario tenga dudas respecto a que hacer, y por esto es que todas las opciones de formularios tendrán ayudas fáciles de encontrar con explicaciones concisas y claras. Para facilitar la búsqueda de la información dentro de las explicaciones, se van a resaltar las palabras o conceptos clave para que llamen la atención del usuario.

Teniendo en cuenta que los usuarios casuales se pueden sentir abrumados por el proceso de creación, interacción y visualización de los tokens, se agregaron múltiples vías de acceso a cada una de las páginas internas de la aplicación, permitiendo que los usuarios “creen” su propio camino hacia las funcionalidades que desean usar.

Para soportar a usuarios más experimentados que pueden llegar a utilizar la aplicación, se agregarán atajos u opciones directas para que una persona que sabe lo que quiere hacer pueda rápidamente llegar a la pantalla que desea. Esto se logra mediante atajos con los logos de servicios conocidos, esperando que un usuario avanzado los reconozca y a través de íconos especiales para facilitar el movimiento entre las opciones de edición, interacción y visualización sin necesidad de volver a escribir el ID del contrato.

Se espera que la mayor parte del uso sea a través de computadoras de escritorio, por lo que si bien se soportan los celulares, el mejor soporte será para computadoras. De todas maneras, dadas las tecnologías utilizadas, el esfuerzo requerido para adaptar el diseño a dispositivos móviles es pequeño, por lo que se podría llegar a adaptar.

Deben ser claras las opciones que tienen los usuarios para interactuar con la aplicación, tanto para crear, editar o interactuar con contratos.

#### 4.3.4 Documento de Diseño

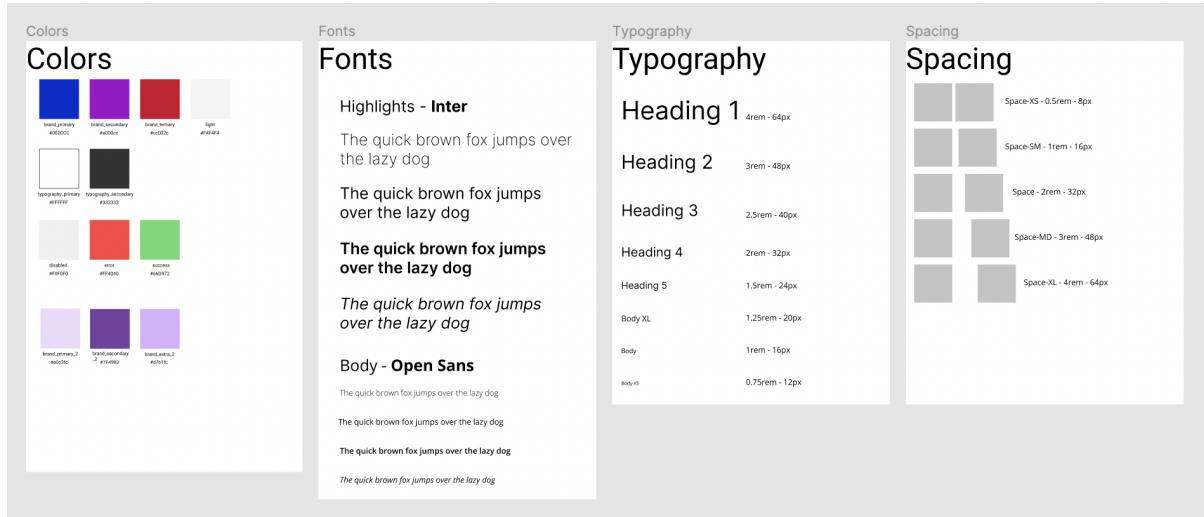
Utilizando la herramienta online *Figma* (<https://www.figma.com/>) se diseñó un prototipo de la interfaz que permite colaborar en el mismo y hacer pruebas con diferentes usuarios. Al ser un prototipo interactivo, permite que hacer pruebas con usuarios sea más simple.

El proceso de diseño fue uno iterativo, en donde se pasaron por varias etapas de diseño con múltiples prototipos. En primera instancia se armó un sistema de diseño, luego se construyeron componentes, y después de eso se construyeron una primera y segunda versión del prototipo. A continuación se detallan las etapas recién mencionadas.

##### 4.3.4.1 Sistema de Diseño

Para poder facilitar y unificar el diseño del prototipo, se generó un sistema de diseño (*Design System*) para tener acceso rápido a los valores para utilizar durante el diseño y

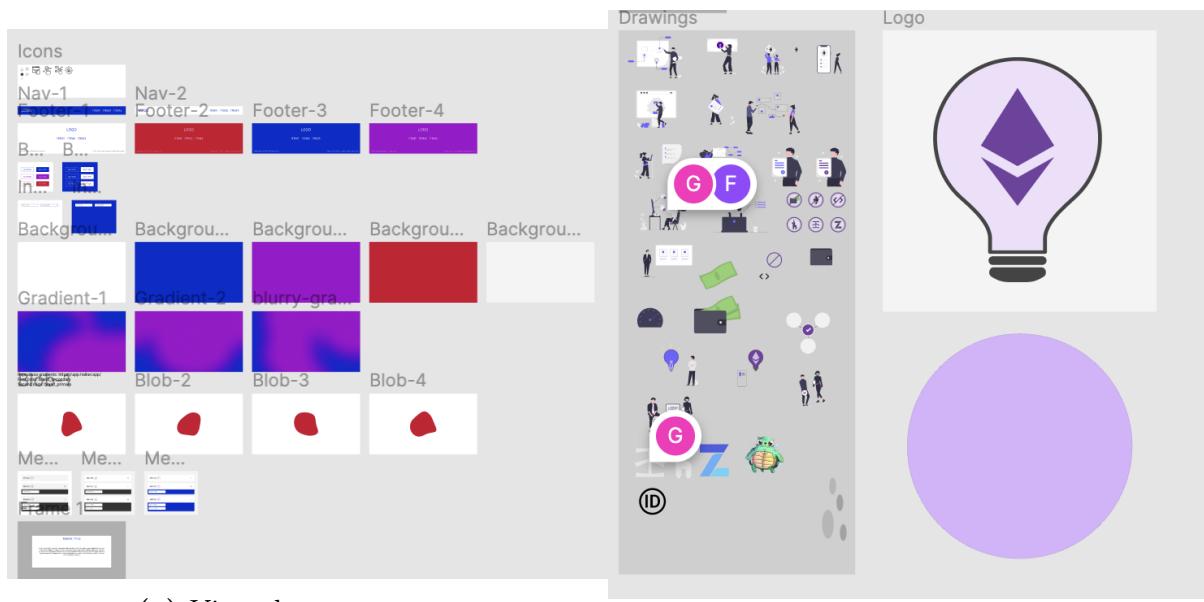
programación de la interfaz. Cómo se puede ver en la Figura 4.1, el sistema de diseño incluye los colores, tipografías, tamaño de fuentes y espaciamiento, mostrando todas las posibles valores que la respectiva propiedad puede tomar.



**Figura 4.1:** Sistema de diseño para el prototipo y aplicación.

#### 4.3.4.2 Creación de Componentes

Una vez creado el sistema de diseño, se construyeron diferentes variantes de componentes para poder reutilizar dichos componentes durante el diseño de los prototipos y poder analizar variantes de los mismos para elegir las mejores. La Figura 4.2a incluye una vista general de los componentes utilizados y la Figura 4.2b muestra los diferentes íconos generados y utilizados.



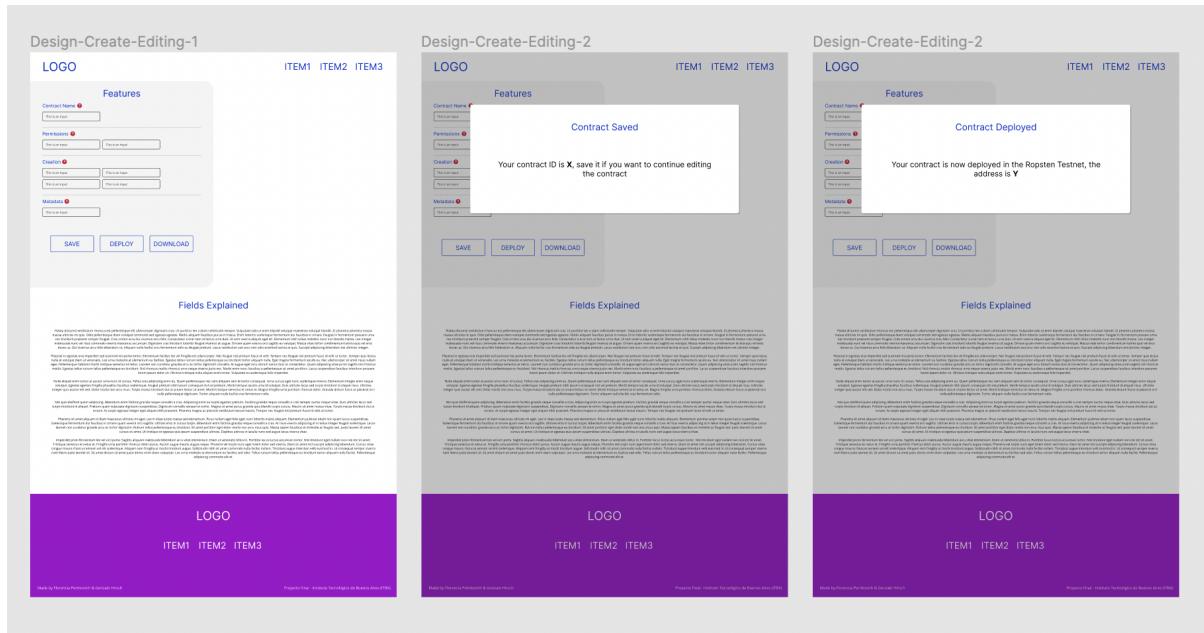
(a) Vista de componentes.

(b) Vista de íconos y dibujos.

**Figura 4.2:** Conjunto de componentes de diseño desarrollados.

#### 4.3.4.3 Interfaz - Primera Versión

Luego de generar una librería interna de componentes, se procedió con el diseño de una primera versión del prototipo para la interfaz. Este primer prototipo incluyó el diseño de la página principal, interfaz de edición e interacción del contrato y páginas de flujo internas. A continuación se incluyen los diseños de la primera versión del prototipo que fueron mantenidos para la versión final.

**Figura 4.3:** Diseño de la interfaz de edición.

En la Figura 4.3 se puede ver la estructura de la interfaz de edición. A la izquierda se ve el diseño del formulario con los botones de acción, mientras que a la derecha hay un espacio en blanco para el código autogenerado. También se incluye la vista del modal de publicación del contrato y un texto tentativo para el mismo. En la parte inferior del diseño se incluye la explicación para los campos y extensiones a elegir.

A continuación en la Figura 4.4 se puede ver el diseño de la página de interacción. Si bien el diseño previsto en la Figura 4.4 no fue el final dado que se eliminó la sección derecha para resultados, se mantuvo el diseño en general. Posee los acordeones esquematizados, cada uno representando un método y los posibles parámetros que el mismo puede requerir. El diseño final separa los métodos de lectura, escritura y creación de tokens para facilitar la navegación entre los mismos. Similarmente a la Figura 4.3, en la parte inferior incluye las explicaciones de los métodos y parámetros que se utilizan, para ayudar al usuario a poder interactuar sin necesidad de buscar fuera de la aplicación las explicaciones.

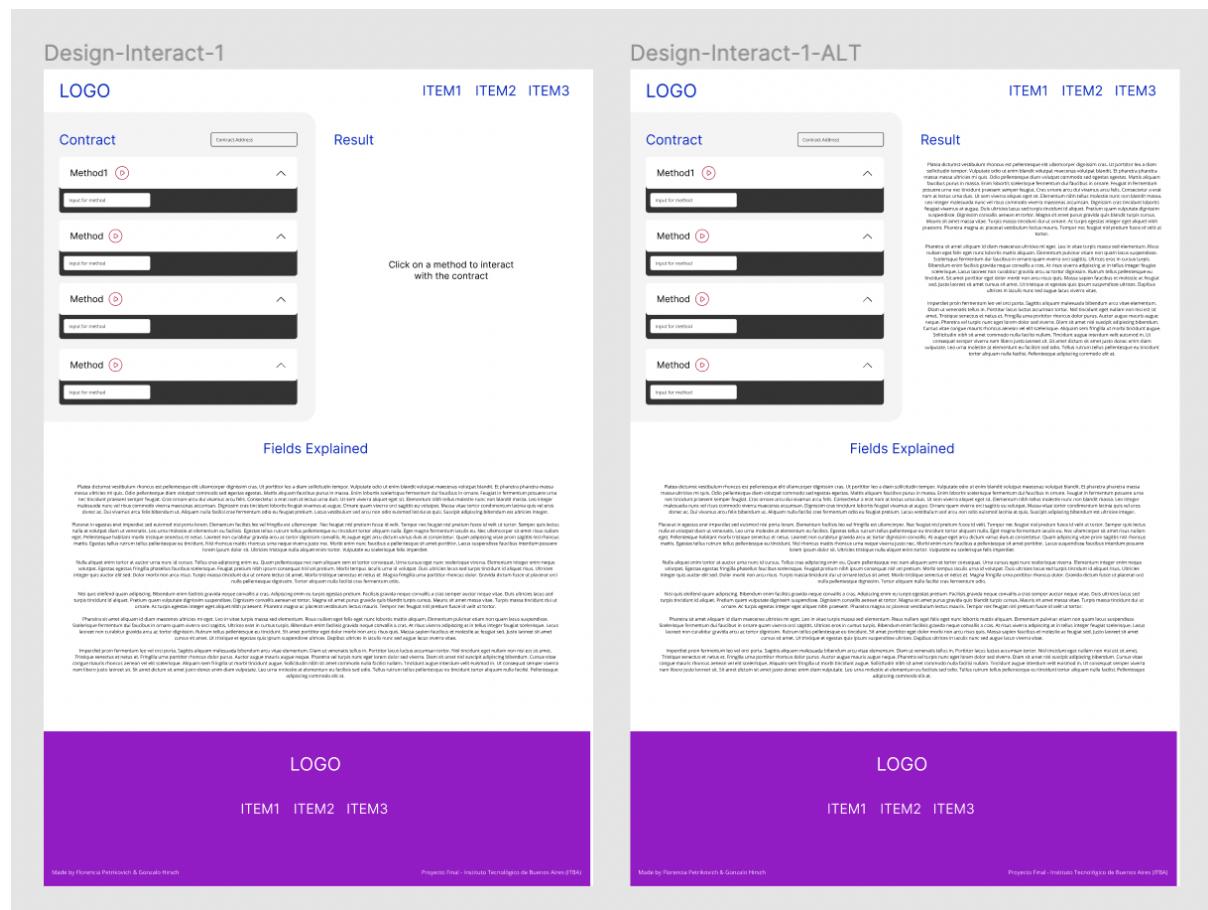


Figura 4.4: Diseño de la interfaz de interacción.

#### 4.3.4.4 Interfaz - Versión Final

Luego de iterar sobre la primera versión del prototipo y de algunos componentes, se diseñó la versión final de la interfaz web de la aplicación. Como se puede ver en la Figura 4.5, se tomaron muchos de los aspectos de la primera versión de la interfaz en la versión final. Específicamente, aspectos del diseño de la interfaz de edición e interacción del contrato y el diseño de múltiples componentes.

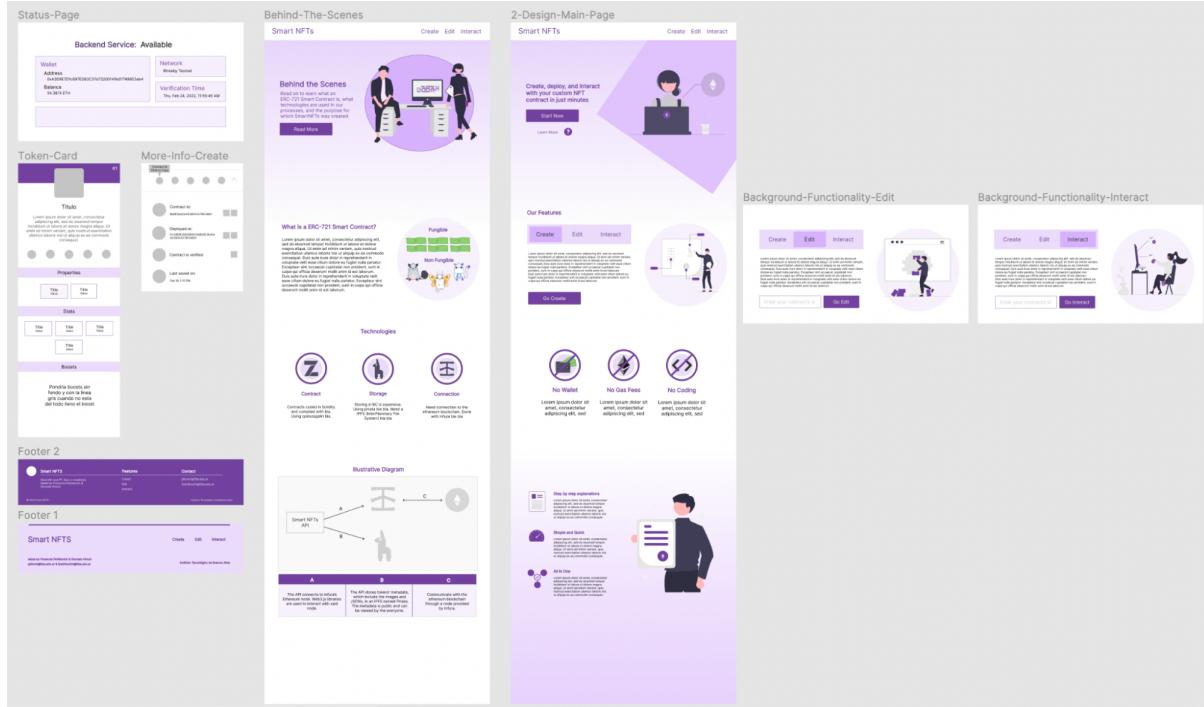
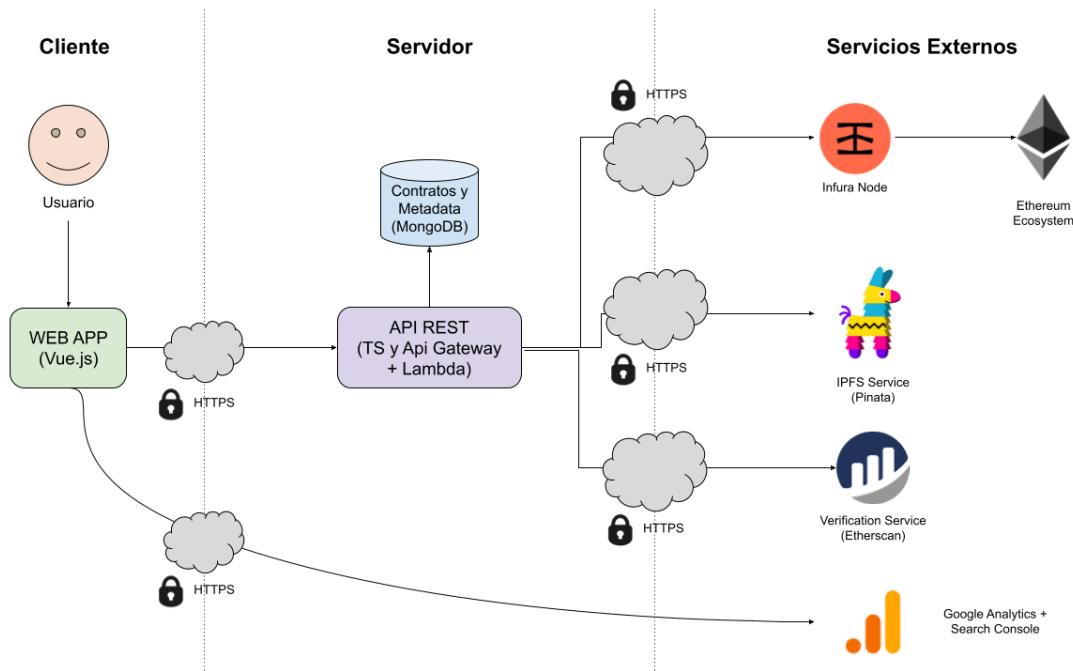


Figura 4.5: Versión final de la interfaz de la aplicación.

## 4.4 Documento de Arquitectura



**Figura 4.6:** Documento de arquitectura dividido en capas.

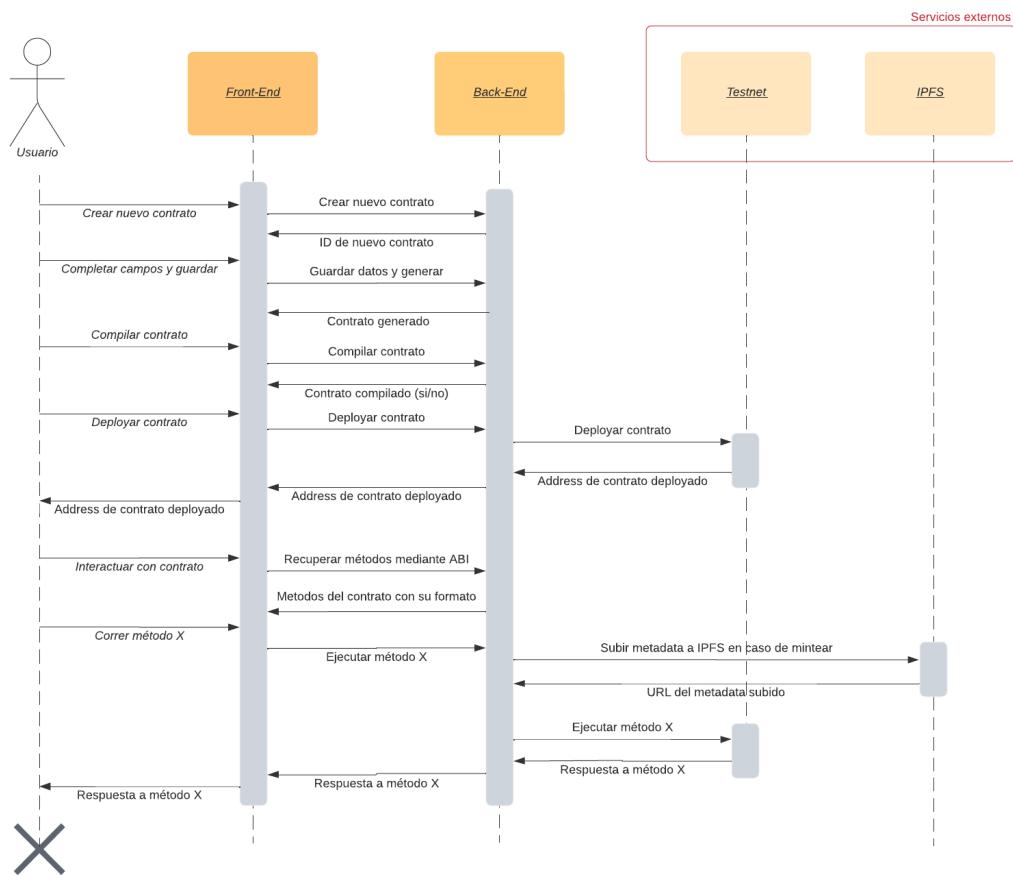
La Figura 4.6 explica la arquitectura establecida dividida en 3 secciones: cliente, servidor, y servicios externos. Por el lado del cliente, utiliza una aplicación web servida desde el servidor, renderizada en el browser del cliente. Interactúa realizando llamadas a la API REST en el servidor. Toda la comunicación se hace mediante HTTPS por seguridad.

La API REST se encuentra en el servidor y se comunica con la base de datos, una base NoSQL orientada a documentos (MongoDB). Esta base de datos guardaría las selecciones para la creación de los contratos. La API también se encarga de la comunicación con los servicios externos. Para la conexión con la Ethereum blockchain, se usará *Infura*, la cual ofrece infraestructura como servicio para abstraer al programador de las complejidades que conlleva mantener una comunicación estable y sincronizada con la blockchain. Por otro lado, la API se comunicará con un servicio IPFS, llamado *Pinata*, para subir la metadata del usuario al momento de generar tokens y recibir una URI asociada que se guardará en los tokens del contrato. Para la verificación de contratos, se comunicará con el servicio de verificación de Etherscan mediante su API, a través de este servicio se pueden verificar contratos publicados en las diferentes Testnets.

En lo que respecta a la analítica, la aplicación web se comunica directamente con los

servidores de Google Analytics (<https://analytics.google.com/>) y Google Search Console (<https://search.google.com/search-console/about>) para informar los diferentes eventos y acciones que ejecuta el usuario.

## 4.5 Diseño de alto nivel



**Figura 4.7:** Diagrama UML de secuencia para el uso de la aplicación.

La Figura 4.7 define una posible secuencia de interacciones que se espera que la aplicación tenga: Crear un contrato, compilarlo, deployarlo y luego interactuar con el mismo. Específicamente el diagrama de la Figura 4.7 incluye a los requisitos funcionales *REQ1*, *REQ2*, *REQ3*, *REQ4* y *REQ5*. La verificación del contrato es opcional, y por ende, no se agregó por simplicidad. Si se fuera a verificar el contrato, el backend se comunicaría con el servicio externo de Etherscan una vez terminada la compilación del contrato.

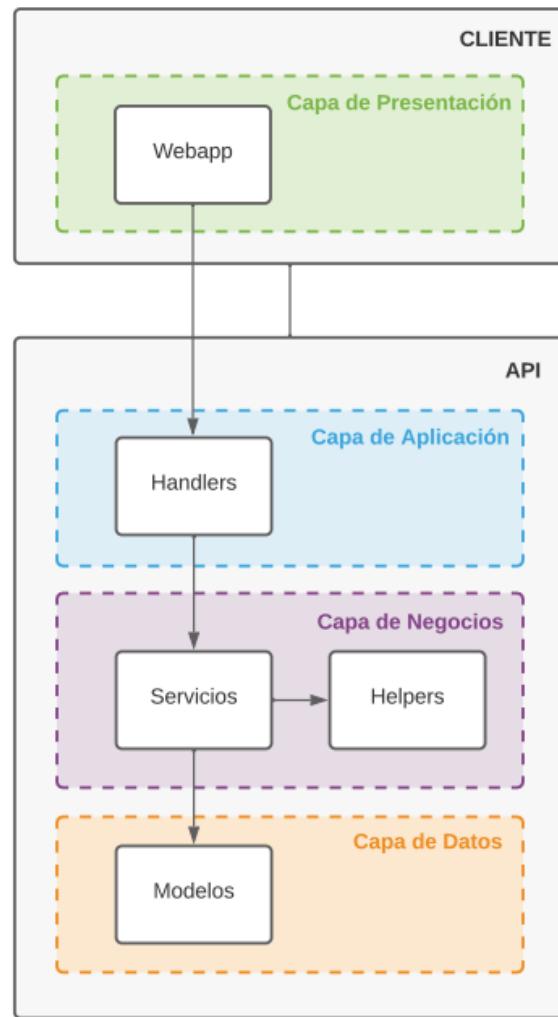
Para poder llevar a cabo con éxito ese caso de uso, se identificó al usuario como actor que inicia la interacción, y 4 diferentes sistemas que interactúan entre sí. Dentro de los

servicios que se consideran internos, se encuentran la aplicación web (frontend) y la api (backend). Como servicios externos se encuentran la testnet (o mainnet en un futuro) que se use y el servicio de IPFS que se vaya a utilizar para el guardado de metadata.

Como se puede observar en el diagrama, los sistemas internos se encargan por su cuenta de la creación y el almacenamiento del contrato en base a la customización establecida por el usuario. Una vez creado y compilado el contrato, utilizan los servicios externos para deployar e interactuar con el mismo en la blockchain. El servicio IPFS se utiliza, en particular, cuando se quiere generar (conocido como *mintear*) un token nuevo. El servicio se encarga del almacenamiento de la metadata especificada por el usuario y el retorno de una URL asociada a lo almacenado para que luego se pueda interactuar con el contrato en la blockchain donde se almacena la URL.

Cabe mencionar que la conexión a la testnet no es fácil de establecer y mantener con el hardware a nuestra disposición. Por esta razón, la conexión se hará mediante un nodo de Ethereum manejado por Infura, quienes ofrecen infraestructura como servicio. Para facilitar la comprensión del diagrama, se estableció la comunicación con la blockchain como “Testnet” en vez de “Nodo de Infura”.

## 4.6 Diseño de bajo nivel



**Figura 4.8:** Separación entre el cliente y la API, con las capas establecidas.

La Figura 4.8 muestra el esquema de separación, especialmente para la API, en donde se utilizó un modelo de desarrollo separado por capas.

El objetivo principal de dicho modelo de desarrollo es el desacoplamiento y la abstracción de las partes que componen un sistema complejo de software. Esta metodología facilita el mantenimiento y desarrollo de la aplicación al suponer reusabilidad y fácil intercambio. A la vez, mejoras en una capa impactan en todo el sistema.

Se tiene un total de 4 capas donde cada una provee servicios a la capa superior.

La primera capa consiste en la **Capa de Presentación**, compuesta por el Webapp. Mediante este GUI, el cliente se puede conectar con la API. La API consiste en la Capa

de Aplicación, Capa de Negocios y la Capa de Datos.

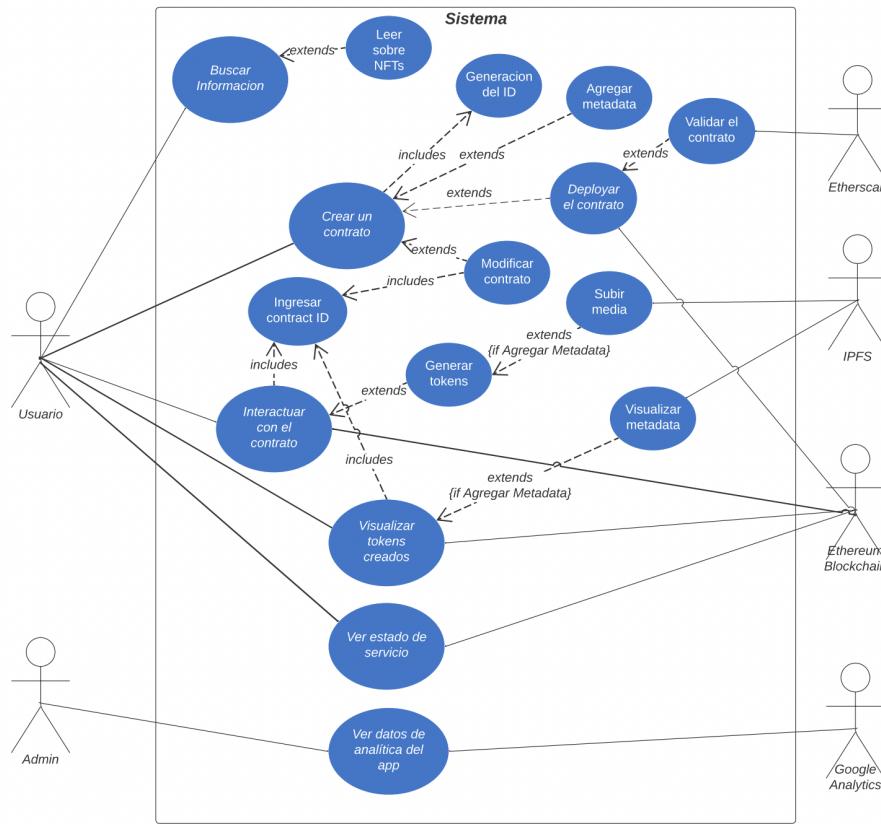
La **Capa de Aplicación** es la capa compuesta por handlers que se ocupan de procesar las diferentes llamadas que puede recibir la API. Cada handler corresponde a una función lambda que puede ser invocada.

La **Capa de Negocios** está compuesta por diferentes servicios, que contienen la lógica para las diferentes acciones que se pueden realizar con recursos. En el caso de creación de contratos, por ejemplo, se tendrá el *ContractCreationService*, con la lógica para generar los contratos. En esta capa se hace uso de los Helpers, funciones que pueden ser reutilizables en diferentes servicios o que pueden minimizar las funciones auxiliares en los servicios.

La **Capa de Datos** contiene los Modelos, que son las abstracciones de los modelos de datos guardados en la base de datos. Contiene lógica para interactuar con la base de datos en sí, para poder guardar información sobre los contratos y la estructura de la metadata.

Esta arquitectura en capas permite en el futuro poder migrar con mayor facilidad entre tecnologías o hasta bases de datos, sin tener que alterar a las capas relacionadas.

A continuación se muestra un diagrama de Casos de Uso para visualizar cómo los diferentes actores y sistemas externos pueden interactuar con el sistema y sus diferentes capas.



**Figura 4.9:** Diagrama de Casos de Uso para el sistema.

## 4.7 Infraestructura

La infraestructura de Smart NFTs se encuentra montada en la nube. Se utilizó *Amazon Web Services* (AWS) como proveedor de la nube. En la Figura 4.10 se puede observar la infraestructura utilizada por el sistema completo; a continuación se describen los principales componentes del mismo.

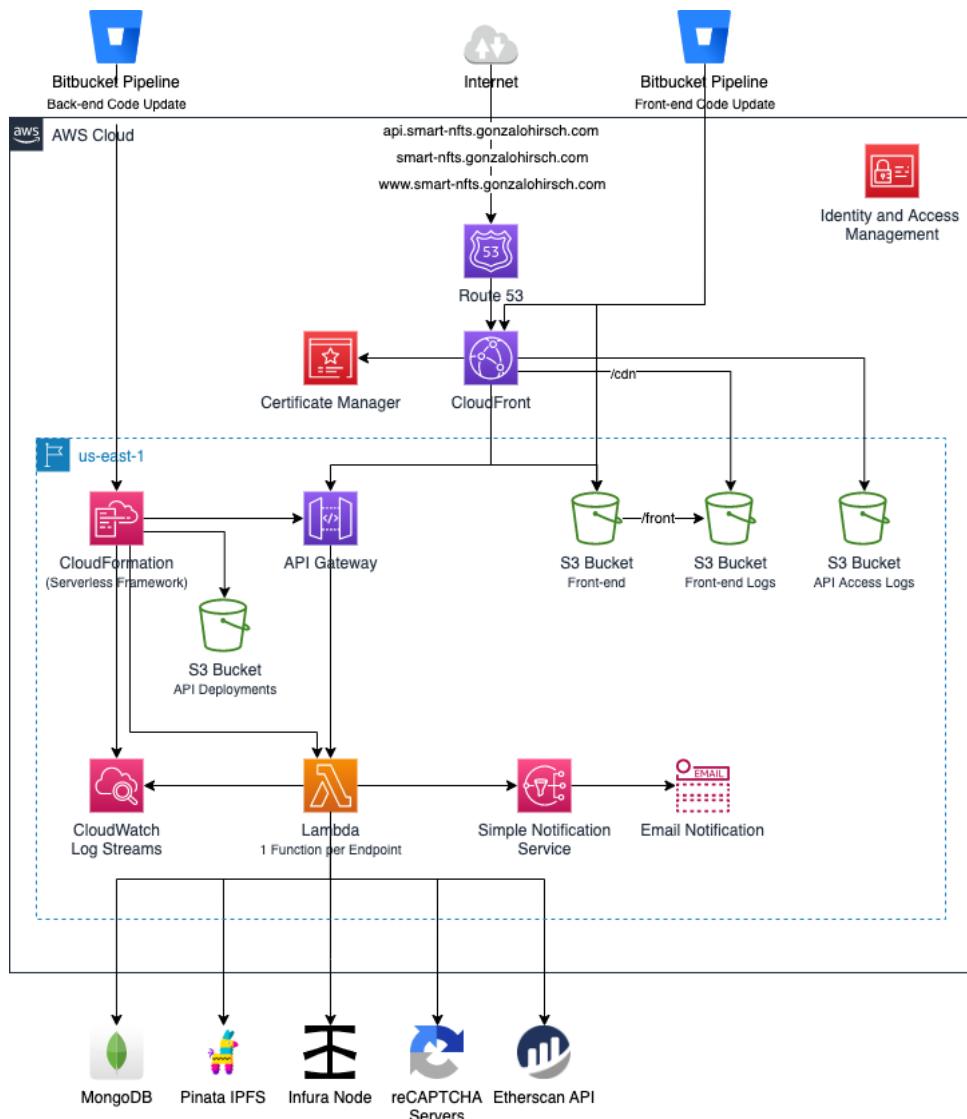


Figura 4.10: Diagrama de la infraestructura del sistema montado en AWS.

La infraestructura se encuentra dividida en dos principales partes, la infraestructura de front-end y de back-end. En ambos casos, front-end y back-end, se utilizaron servidores DNS (Route 53) para la resolución de nombres, certificados SSL/TLS generados vía el *Certificate Manager* de AWS para proveer una conexión mediante HTTPS y *Cloudfront* como CDN (Content Delivery Network) para la distribución de contenido y aceleración de la API respectivamente. La resolución de nombres se hizo con un subdominio de *gonzalohirsch.com*, un dominio perteneciente a uno de los miembros del equipo, para poder tener resolución DNS y no tener que pagar por un nombre.

Para montar la infraestructura del front-end, dado que se utilizó una aplicación web que no requiere de servidores, se optó por un Bucket de *S3* (Simple Storage Service). El

bucket permite guardar el contenido de la aplicación web de manera económica y en un medio de alta durabilidad, sin necesidad de provisionar servidores.

Por el lado del back-end, se optó por desarrollar una API Serverless. Las dos principales ventajas de este enfoque son que no hay servidores, por lo que se paga por uso y cada función es escalable por su cuenta. Esto permite que si hay una parte del proceso que se utiliza más, escala automáticamente para suplir la demanda y en caso de que no se use, no se incurre en costos. La principal desventaja de este enfoque, ya mencionada anteriormente, es que tiene un límite de ejecución de 29 segundos. Se utiliza *API Gateway* para construir una API REST, en donde cada ruta es una función Lambda y esas funciones reportan logs y métricas a *CloudWatch*. Se utiliza *Simple Notification Service* (SNS) en el envío de emails para el flujo en donde el usuario envía un recordatorio del email; por cuestiones de seguridad de datos y anonimidad, esas direcciones no se persisten en el sistema.

Las funciones Lambda también se comunican con MongoDB para acceder a la base de datos ya que se optó por una base de datos noSQL, orientada a documentos, principalmente por la flexibilidad del esquema y el hecho que muchos componentes de los contratos (como el ABI) están en formato JSON. Para esto se utilizó un cluster provisionado por MongoDB Atlas [19], que ofrece una capa gratuita. También se comunican con otros servicios externos como:

- Pinata IPFS, para el guardado y recupero de la metadata de los tokens.
- Infura, para la conexión, ejecución y escaneo de la blockchain.
- reCAPTCHA (<https://developers.google.com/recaptcha/docs/v3>), para verificar los tokens que el front-end envía para los desafíos reCAPTCHA.
- Etherscan, para verificar contratos y validar balances.

En el caso de ambos, front-end y back-end, se crearon Buckets de S3 para el guardado de logs de acceso. En el caso del front-end, la distribución de CloudFront y el Bucket de front-end generan logs que se guardan en un mismo Bucket. En el caso del back-end, CloudFront también genera logs de acceso para la API, que luego se guardan en un Bucket. Estos registros permiten hacer análisis y entender posibles errores.

### 4.7.1 Flujos de Integración Continua

Se implementaron flujos de automatización de publicación de código para ambos front-end y back-end a través de *Bitbucket Pipelines*, incluidos en el repositorio provisto. Estos flujos hacen uso de usuarios especiales provistos mediante el servicio *Identity and Access Management* (IAM) de AWS.

En el caso del front-end, se realizan operaciones vía la AWS CLI en donde se genera el código del front-end, se actualiza el contenido del Bucket de S3 que lo contiene y se invalida el caché de la distribución de CloudFront que lo distribuye.

En el caso del back-end se utiliza *CloudFormation*, un servicio de IaC (Infrastructure as Code), a través del framework *Serverless*. Esto genera un flujo en donde al construir la aplicación se ejecuta un Stack de CloudFormation, que a su vez genera las funciones Lambda necesarias, nuevas rutas en API Gateway y Log Streams en CloudWatch. Al mismo tiempo se guarda una copia de las últimas 5 publicaciones de código en un Bucket de S3, en caso de necesitar volver a una versión anterior.

## 4.8 Seguridad

El back-end maneja operaciones y datos sensibles, como la billetera virtual para publicar e interactuar con contratos, envío de emails y operaciones costosas con la blockchain. Dado esto, se buscó limitar el acceso a la API ya que el sistema se basa en no tener usuarios. Para esto se implementaron 2 medidas principales.

La primera medida es el uso de Google reCAPTCHA v3 para prevenir el abuso por manos de código automatizado. Es requerido un token especial, que solo puede ser generado desde el dominio donde se encuentra el front-end, que es luego validado por el back-end. Se utiliza un reCAPTCHA invisible que se envía al back-end y luego se verifica con los servidores de reCAPTCHA.

La segunda medida es el uso de *Cross-Origin Resource Sharing* (CORS) en la API para limitar los posibles dominios que utilizan el contenido a través de la web.

Estas medidas, en conjunto con otras como encabezados de seguridad en las respuestas de la API a través de CloudFront y requerir encabezados especiales, permiten limitar el uso

no autorizado de la API.

## 4.9 Analítica

Parte de los requerimientos del sistema consiste en generar analítica para los administradores<sup>2</sup>. Para esto se implementaron tres enfoques para obtener analítica.

El primer enfoque es a través del uso de Buckets de S3 para el guardado de logs de acceso para tanto el back-end como el front-end. Si bien estos datos no se utilizan actualmente para la generación de métricas u otros resúmenes, se genera el contenido crudo para poder realizar analítica en el futuro.

El segundo enfoque es la implementación de Google Analytics, un servicio de reportes web ofrecido por Google que permite ver la cantidad de usuarios activos en tiempo real y métricas de adquisición, entre otras. Permite entender quién es el usuario que utiliza la aplicación, y cómo lo hace, para poder mejorarla y hacerla más atractiva.

El último enfoque está dado por la implementación de Google Search Console, un servicio de Google para permitir indexar sitios o aplicaciones web en Google. A través de esto, se hace pública la aplicación y es posible de encontrar en Google, por ejemplo, al buscar “*smart nfts itba*”<sup>3</sup>. Además permite entender el posicionamiento en Google y los términos que utilizan los usuarios para poder encontrarlo.

---

<sup>2</sup>Ver REQ-13.

<sup>3</sup>“*smart nfts*” parece ser un término competitivo en Google, por lo que no aparece el sitio en la primera página del buscador, suele aparecer en la tercera o cuarta.

## 5 Desarrollo

Para llevar a cabo el desarrollo y puesta en producción del sistema completo se utilizaron un conjunto de metodologías ágiles. Se aplicó Scrum para la administración del proyecto, en tanto que se hizo uso de un tablero inspirado en Kanban para la visualización y separación de tareas.

La aplicación de Scrum permitió desarrollar ciclos de iteración (*sprints*) de 2 semanas de duración. De esta manera, al comienzo de un *sprint* se definen tareas pendientes para realizar en el *Sprint Backlog*, utilizando como base las tareas pendientes del *Product Backlog*. Dichas tareas son las que cada miembro del equipo trabajó durante las siguientes 2 semanas.

Para el seguimiento de las tareas, se utilizó un tablero inspirado en Kanban a través de la herramienta web Trello (<https://trello.com/>), utilizando las siguientes columnas, TODO, WORK IN PROGRESS, BLOCKED y DONE. Dicho tablero se integró con Slack, herramienta de mensajería, para el envío de notificaciones a los miembros del equipo cuando una tarea se movía en relación a la columna BLOCKED.

La columna TODO se utiliza como *Product Backlog*, en donde se listan todos los requisitos del producto y las posibles tareas para realizarse. En caso de ser asignada a un *sprint* se marca a la misma con el nombre o ícono del integrante que debe completarla, efectivamente siendo en ese momento parte del *Sprint Backlog*. Las tareas en esta instancia se marcan con la categoría FRONT o BACK, que representan tareas relacionadas al front-end y back-end respectivamente. También se utilizaron categorías para definir tareas que representan errores o son de alta importancia.

La columna WORK IN PROGRESS representa una tarea que se encuentra siendo llevada a cabo por un miembro del equipo.

La columna BLOCKED se utiliza en casos especiales, cuando un miembro del equipo se encuentra bloqueado en una tarea, sea por un error que no puede resolver, falta de información o cualquier otra causa. Como fue mencionado antes, se utilizó una integración con Slack para poder informar a todo el equipo sobre esto, en caso de que alguien quiera ayudar en esto.

Por último, la columna DONE se utiliza para marcar los requisitos que fueron completados por un miembro del equipo y fueron publicados en la versión de producción.

Como fue mencionado anteriormente, las publicaciones de código se hacen automáticamente utilizando los flujos provistos por *Bitbucket Pipelines*.

## 6 Resultados

En esta sección, se analizarán los resultados obtenidos de las distintas pruebas llevadas a cabo. El objetivo principal de las pruebas fue encontrar fallas en la experiencia de los usuarios que utilizan la aplicación y errores de programación que puedan surgir a futuro. Para esto se realizaron tres diferentes pruebas descritas a continuación.

### 6.1 Tipos de Pruebas

#### 6.1.1 Pruebas de UX

Estas pruebas involucran a diferentes usuarios reales interactuando con la aplicación en escenarios planteados por el equipo, para poder observar las decisiones que toman frente al uso que le pueden dar. En general las observaciones que se hicieron a partir de las mismas fueron mejoras de UX y correcciones en el flujo de la aplicación y diferentes pantallas.

#### 6.1.2 Unit testing

Estas pruebas involucran escribir tests unitarios en el backend para probar las diferentes funciones que se utilizan en esa base de código. Para esto se utilizó la librería de testing Jest (<https://jestjs.io/>).

#### 6.1.3 Análisis Estático

Se utilizó Sonarqube (<https://www.sonarqube.org/>) para realizar un análisis estático del backend para poder encontrar errores y potenciales mejoras que simplifican el mantenimiento del código. Este analizador es Open Source (en su versión gratuita) y los cambios para poder instalarlo y correrlo en el backend son mínimos.

## 6.2 Pruebas de UX

Durante 2 semanas se realizaron pruebas de UX con usuarios reales, en sí se realizaron pruebas con un total de 8 usuarios. Estos usuarios podían clasificarse dentro de dos tipos, específicamente los tipos definidos previamente en la sección de “Tipos de usuarios

identificados”.

Las pruebas realizadas involucran una serie de preguntas sobre la información disponible en la aplicación y un conjunto de acciones para llevar a cabo dentro de la aplicación. Estas preguntas y tareas se les planteaban a los usuarios mientras eran observados.

Luego de realizar la interacción con la aplicación, debían completar un formulario para relevar datos extra tanto de su experiencia, como de ellos mismos, de forma que se pudiera realizar un perfil más comprensivo sobre el tipo de usuario que utilizó la aplicación.

Previo a comenzar las pruebas, se esperaba una gran independencia y decisión en las acciones que los usuarios experimentados realizaban, sin embargo, se consideró que era muy probable que los usuarios casuales tuvieran dificultades para realizar algunas de las acciones o tareas, principalmente por una falta de conocimiento sobre Smart Contracts y NFTs. No obstante, se creyó que dado el diseño y contenido de la aplicación, la curva de aprendizaje iba a ser empinada, por lo que el aprendizaje iba a ser veloz. Debido a esto último, se diseñaron las pruebas de manera que eventualmente se requería que realicen acciones similares a las propuestas inicialmente, para poder evaluar el nivel de aprendizaje del usuario en torno al uso de la aplicación.

### 6.2.1 Casos de Prueba

Se identificaron los siguientes casos de prueba:

- Un usuario busca información sobre qué es un NFT.
  - Requerimientos relacionados: *REQ12*.
- Un usuario genera, publica y verifica un contrato.
  - Requerimientos relacionados: *REQ1*, *REQ2*, *REQ3*, *REQ4* y *REQ6*.  
(segmentado en múltiples acciones que se requieren de los usuarios).
- Un usuario interactúa con el contrato y genera un token.
  - Requerimientos relacionados: *REQ12*.
- Un usuario busca explicaciones de los campos que completa.
  - Requerimientos relacionados: *REQ5*.

- Un usuario busca explicaciones sobre los métodos y parámetros al interactuar.
  - Requerimientos relacionados: *REQ12*.
- Un usuario quiere ver los tokens creados y la información de los mismos.
  - Requerimientos relacionados: *REQ10*.
- Un usuario busca si el servicio de la aplicación está funcionando.
  - Requerimientos relacionados: *REQ11*.
- Un usuario quiere guardar el ID del contrato.
  - Requerimientos relacionados: *REQ8*.
- Un usuario vuelve a la aplicación y agrega funcionalidad a un contrato ya existente.
  - Requerimientos relacionados: *REQ9*.
- Un usuario ve el contrato, token o metadata creados en servicios externos.
  - Requerimientos relacionados: *REQ10*.

Se consideró que los *REQ7* y *REQ13* no poseían casos de pruebas necesarios para las pruebas de UX.

Para poder realizar pruebas sobre los casos de prueba definidos, se crearon flujos de prueba para que los usuarios naveguen. Esos flujos contienen búsqueda de información y acciones para realizar.

Algunos ejemplos de los flujos son:

- Buscar información sobre NFTs en la página.
- Crear un contrato en donde se pueden crear tokens, destruirlos y agregarles una imagen.
- Agregar 2 propiedades a los tokens del contrato creado, una numérica porcentual y otra textual.
- Publicar y verificar el contrato creado.
- Identificar a la dirección dueña del contrato (mediante la interacción con el mismo) y crear un token.

- Visualizar el token y sus propiedades tanto en Smart NFTs como en servicios externos.
- Editar el contrato luego de haber interactuado con el mismo y agregar funcionalidades para que no se pueda repetir la metadata del token y que se pueda deshabilitar la creación de los mismos.
- Verificar el estado del servicio.

### 6.2.2 Resumen de Resultados

En general, las pruebas demostraron que habían potenciales mejoras en lo que respecta a la UX y el contenido de la aplicación. Se notó que los usuarios casuales tenían dificultades inicialmente para realizar acciones con la aplicación, principalmente por no poseer conocimientos sobre NFTs y Smart Contracts. Sin embargo, exactamente como se creyó previo al comienzo, la aplicación posee una curva de aprendizaje empinada, por lo que usuarios sin experiencia, luego de unos pocos minutos de usarla podían empezar a realizar las acciones solos.

Los usuarios avanzados utilizaban las explicaciones menos que los casuales, por ende, se recibieron varias críticas constructivas de los usuarios casuales de cómo mejorar las explicaciones, ya sea por falta de detalles o claridad o errores gramaticales.

### 6.2.3 Resultados de Pruebas

#### 6.2.3.1 Buscar información sobre NFTs

Para este caso de prueba se le solicitó a los usuarios que naveguen e indiquen en donde (dentro de la aplicación) buscarían información para saber más sobre NFTs, suponiendo que no supieran lo que eran.

Inicialmente los usuarios tenían dificultades encontrando las páginas con información extra ya que los hipervínculos no eran muy claros. Por ejemplo, el hipervínculo para aprender más sobre NFTs decía solamente Learn More, lo cual no es específico a qué información se está refiriendo. Además, en muchas instancias se utilizaba el nombre del estándar, ERC-721, en vez del nombre más común, NFT, por lo que los usuarios tenían dificultades al realizar esta tarea ya que no sabían que hacía referencia a lo mismo.

En base a lo observado, se modificaron los textos de los hipervínculos y se reemplazaron instancias de “ERC-721” por “NFT” para una mayor claridad. Esto resultó en otros grupos de usuarios encontrando sin dificultades las páginas con explicaciones y las explicaciones útiles en sí para el caso de prueba.

Además se observó cómo los diferentes usuarios alcanzaban las páginas de información mediante los diferentes hipervínculos que se agregaron en toda la página, reforzando las decisiones del equipo sobre su posicionamiento.

#### **6.2.3.2 Generar, publicar y verificar un contrato**

Para este caso de prueba se le solicitó a los usuarios generar un contrato con ciertas funcionalidades específicas, y luego publicar y verificar al mismo.

Como era de esperar, los usuarios casuales mostraron dificultades al tratar de asociar la funcionalidad requerida y la extensión a utilizar. En estos casos pudieron hacer uso de los botones de ayuda para las extensiones. Las mayores dificultades se presentaron en la lectura de las explicaciones o encontrar explicaciones para las acciones de publicar y verificar, que no tenían botones de ayuda. Si bien hubieron dificultades para entender el 100% del lenguaje utilizado (por ejemplo, “bytecode” y “compilar”), la esencia de la explicación se entendía y en la mayoría de casos podían realizar las acciones.

En base a lo observado, se agregaron botones de ayuda en lugares estratégicos para publicar y verificar y se modificaron las explicaciones para poder enfatizar palabras clave para llamar la atención de los usuarios al momento de leerlas. Esto resultó en usuarios posteriores utilizando los nuevos botones para ayuda y potencialmente entendiendo mejor las explicaciones.

#### **6.2.3.3 Buscar explicaciones de campos**

Para este caso, se observó si los usuarios necesitaban explicaciones o no, y como las alcanzaban.

En general los usuarios entendieron por su cuenta que hacer click en los botones de ayuda los llevaba a las explicaciones en sí. Luego de haber experimentado ese comportamiento, volvían a usar los botones en otras circunstancias o simplemente navegaban hacia la explicación en sí si ya conocían la localización de la misma. En algunos casos se observó

que no lograban encontrar lo que buscaban ya que faltaban algunas palabras clave resaltadas.

En base a lo observado, se enfatizaron nuevas palabras clave y se modificaron levemente las explicaciones para mejorarlas. También se agregaron botones de ayuda extra para publicar y verificar, lo cual fue mencionado en el ítem anterior.

#### 6.2.3.4 Guardar ID del contrato

Para este caso, se le planteaba al usuario la necesidad de guardar el ID del contrato para su uso en el futuro, incitandolos a descubrir los medios que podían emplear para realizar eso.

Inicialmente se observó una dificultad al tratar de comprender cuál era el ID del contrato y encontrar como copiarlo (sin necesidad de ir a la URL que contiene el ID). Además, tenían problemas para encontrar el botón para enviar un mail con el mismo.



**Figura 6.1:** Conjunto de capturas de pantalla mostrando algunos de los cambios/adiciones implementadas.

En base a lo observado, se agregaron íconos hechos a medida para que se represente al ID de manera más intuitiva. Por otro lado, se agregó texto e íconos flotantes para recordarle a los usuarios que se guarden el ID y que pueden copiarlo o enviarlo por mail. Los íconos actualizados generaron que los usuarios confundan menos el ID del contrato con el address del mismo (se expandirá sobre el tema en los siguientes ítems) y que encuentren los botones con más facilidad. El ícono flotante resultó en una mayor cantidad de usuarios encontrando el botón para enviar el mail.

#### 6.2.3.5 Interactuar con contrato y generar un token

Para este caso de prueba, se le plantea al usuario partir de la edición del contrato e ir a la interacción con el mismo.

Se observó que diferentes usuarios llegaban a la interfaz de interacción de diferentes maneras, que es lo esperado y refuerza la decisión de poner varias alternativas para llegar.

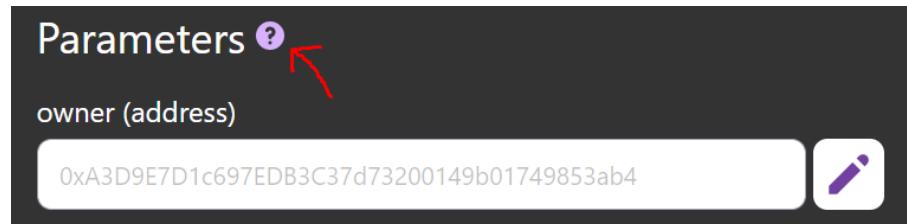
Una vez encontrada la página de interacción y luego de haber ejecutado el método indicado, les resultaba a los usuarios confuso o molesto que no se expandieran automáticamente los acordeones con los resultados de la ejecución, lo cual significa que el resultado no es fácilmente visible.

En base a lo observado, se agregó la funcionalidad que abre el acordeón en presencia de un resultado y que mueve al usuario automáticamente a la posición correcta para ver el resultado luego de la ejecución. Esto ayudó a disminuir la confusión de los usuarios al usar eso.

#### 6.2.3.6 Buscar explicaciones sobre métodos y parámetros

Para este caso de prueba, se observó cómo reaccionan los usuarios al necesitar información sobre los métodos y los parámetros de los mismos.

Inicialmente los usuarios casuales tenían algunas dificultades para entender los parámetros y los tipos, y no encontraban explicaciones para eso. Eso llevaba a que las interacciones posteriores tengan una mayor dificultad al poseer menos información.



**Figura 6.2:** Cambios en la pantalla de interacción.

En base a lo observado, se agregaron explicaciones y textos extra para facilitar la comprensión, junto a botones de ayuda en donde corresponde. Esto llevó a que posteriores usuarios puedan interactuar con más facilidad y puedan comprender mejor los tipos de parámetros.

#### 6.2.3.7 Visualizar tokens creados

Para este caso de prueba, se le solicita al usuario que muestre el/los tokens que creó previamente con el contrato en cuestión.

Inicialmente algunos usuarios tenían dificultades para encontrar la página de visualización, pero se cree que en muchos de los casos se debió a la manera de realizar la pregunta, potencialmente insinuando que era parte de la página de interacción, por lo que se desestima esta observación inicial. En general los usuarios no tuvieron dificultades visualizando los tokens y su información relacionada. Por esto es que no hubo cambios que surgieron a partir de este caso de prueba.

Se utilizaron ambos caminos existentes para llegar a la página de visualización, lo cual refuerza nuestra decisión de hacer la página accesible desde Interact mediante un ícono representativo.

#### **6.2.3.8 Verificar estado del servicio**

Para este caso de prueba, se le solicita al usuario que verifique el estado del servicio de la aplicación.

Se observó que los usuarios que más exploraron inicialmente la aplicación lo encontraron con más facilidad y que en sí llegaron con todas las diferentes maneras que la aplicación ofrece para alcanzar esa página. En base a esto, no hubo cambios necesarios a la experiencia del usuario.

#### **6.2.3.9 Volver a la aplicación y editar contrato**

Para este caso de prueba, se le solicita al usuario que vuelva a editar un contrato y agregue funcionalidades extras.

Se observó que todos los tipos de usuarios llegaron al editor del contrato a través de múltiples métodos, reforzando los posicionamientos de hipervínculos a lo largo de la página. Cabe mencionar que la mayoría experimentaba cierta confusión al momento de ingresar el contract ID del contrato existente, dado que confundían el mismo con el address del contrato deployado. Al recibir el mensaje de que dicho contrato no existía, notaban por su cuenta el error y lo corregían, ingresando el ID del contrato correcto. El ícono flotante de ID ayudó a disminuir estos casos. A pesar de esto, se considera una equivocación simple dado que no tiene consecuencias y se espera que solo se realice la primera vez que se utiliza la funcionalidad.

En términos de agregar las funcionalidades, como era esperado, los usuarios casuales

tuvieron una dificultad mucho menor al momento de agregarlas, por lo que demuestra que ya con un poco de uso, los mismos no necesitan ayuda para realizar las acciones. No hubo cambios que surgieron a partir de estas observaciones.

#### 6.2.4 Resultados de Formulario

Dentro de lo relacionado a la experiencia al utilizar la aplicación, la mayoría de los usuarios (principalmente los usuarios casuales) declararon que en algún momento se sintieron perdidos o confundidos, y eso llevó a que lean las explicaciones. Todos los usuarios leyeron alguna clase de explicación de las presentes en la aplicación, y a su vez todos dijeron que las explicaciones sirvieron para aclarar el tema.

En términos de la impresión de la aplicación, todos los usuarios se encontraron muy satisfechos con el diseño, funcionalidades y experiencia al utilizarla.

En lo que respecta a la opinión sobre la misma, la mayoría dijeron que la utilizarían, y los que no declararon eso, mencionaron que no conocen del tema lo suficiente como para poder definir si la van a usar o no. La gran mayoría recomendaría la página a un amigo/a, mencionando dentro de las respuestas que es atractiva la facilidad, simpleza y claridad para realizar todo lo que ofrece la aplicación. Lo cual indica que cumplimos uno de los objetivos principales planteados al realizar el desarrollo, cubrir la necesidad de una solución de esta índole, mediante una interfaz simple y fácil de utilizar.

Por último, mediante preguntas sobre conocimientos del entorno crypto y otros temas, se pudo definir con mayor precisión el tipo de usuario en cuestión, ayudando a determinar el nivel de interacción esperado para cada uno.

### 6.3 Unit Testing

Para testear lógica compleja del backend, se crearon unit tests utilizando la librería Jest. Unit testing se enfoca en probar componentes pequeños o una parte de la funcionalidad del código base. Su objetivo es validar que la unidad se comporta como se espera en condiciones aisladas. En este caso, se deseaba analizar el comportamiento de lógica compleja que no involucre servicios de terceros, aislandolo del mismo. Con este tipo de pruebas, se podría verificar que la aplicación se está comportando correctamente al generar

y validar datos que van a ser mandados a los servicios y recibidos de los mismos.

Se eligió el uso de la librería Jest por su configuración y manejo simple, su compatibilidad con TypeScript, y su uso popular en conjunto con el Serverless Framework. Además es la librería recomendada por el framework en sí.

Los casos que se encararon son aquellos esenciales para el funcionamiento de la aplicación y que solo dependen de nuestra API, es decir, no involucra los servicios externos. Estos casos son la *creación del código del contrato en base a los inputs del usuario, el manejo del ingreso de la metadata en comparación a la declarada al crear el contrato y la estructuración de los argumentos para mandar a la blockchain*. Los casos que involucran servicios externos no se consideraron dado que el objetivo de unit testing es poner a prueba partes pequeñas de la aplicación en aislamiento completo y no se permite interacciones con dependencias externas.

En los casos testeados, se puso a prueba la respuesta de los métodos ante inputs válidos e inválidos, verificando que el mensaje de error sea adecuado en el segundo caso.

### 6.3.1 Resumen de Resultados

Los unit tests expusieron errores de distintas severidades que permitieron mejorar el backend. Se detectaron faltas de validaciones, manejos incorrectos de errores y, en un caso, una necesidad de preprocesamiento extra por falta de código esencial.

### 6.3.2 Resultados

#### 6.3.2.1 Creación del contrato

En los unit tests de esta sección, se verificó que la creación de un contrato (1) sea la correcta ante parámetros válidos y (2) ejecute un error específico ante el ingreso inválido de argumentos. La creación del contrato cuenta con los siguientes parámetros:

1. Nombre del contrato.
2. Símbolo del contrato.
3. Lista de extensiones.

**6.3.2.1.1 Argumentos válidos** Se notó con los test que ante argumentos válidos, el contrato se generaba correctamente. Se probaron varios casos con combinaciones distintas de extensiones, nombres y símbolos válidos.

**6.3.2.1.2 Argumentos inválidos** Por un lado, se analizó el ingreso incorrecto de los parámetros mandatorios: nombre y símbolo de contrato. Particularmente, se verificó que se ejecute el error adecuado para cada caso, los cuales son falta de alguno de los argumentos, el uso de caracteres inválidos o una cantidad inválida de caracteres.

Estos tests nos permitieron encontrar que dichas validaciones sobre los parámetros nombre y símbolo se realizaban en el lugar incorrecto, el cual era al momento de guardar el contrato en la base de datos y no al momento de creación. Cambiar el posicionamiento de las validaciones al momento de la creación permitió mejorar el procesamiento al evitar la creación del contrato.

Por último, se probó la creación del contrato con combinaciones inválidas de las extensiones. Estas combinaciones son aquellas que incluyen una extensión hija sin incluir al padre de las cuales dependen.

Con esto se observó una falta total de validaciones para la prevención de estos casos. Agregado a esto, se notó un manejo inadecuado de ciertos errores que se usaban en distintas situaciones. Se aplicaron las mejoras sobre el manejo de los errores y se agregaron las validaciones faltantes debido a los resultados de estos tests.

### 6.3.2.2 Manejo de Metadata

Con dichos tests, se verificó que la metadata que ingrese el usuario al momento de crear su token corresponda con aquella que definió al momento de crear el contrato, y de no ser el caso, que el error adecuado sea ejecutado e informativo sobre lo ocurrido.

**6.3.2.2.1 Ingreso valido de metadata** Se realizó un testeо con el ingreso de metadata válida que se conformaba a la definición de la misma, resultando en una ejecución exitosa y limpia ante un ingreso adecuado.

**6.3.2.2.2 Ingreso inválido de metadata** Para el ingreso inválido, se analizó la falta de los atributos mandatorios y el tipo inadecuado de los mismos. Estos atributos pueden

ser los agregados por defecto (`name`, `description`) o aquellos definidos por el usuario en la creación del contrato. Ambos tipos deben estar presentes y ser del tipo correcto para que se consideren válidos.

Se nota que ante ambos errores, falta o tipo incorrecto, el mismo error se estaba ejecutando, lo cual no era informativo sobre que particularmente estaba causando el error. Para encarar este problema, se crearon errores distintos e informativos, separando la validación de los errores en dos casos disjuntos.

### 6.3.2.3 Manejo de la entrada de los métodos del contrato

Por último, se puso a prueba el manejo de los datos de entrada para la ejecución de un método del contrato. Las entradas se deben mandar con un orden específico al interactuar con un método en la blockchain, pero el usuario puede no necesariamente ingresarlos en el orden correcto.

**6.3.2.3.1 Entradas válidas** Al testear este método con una entrada válida, en primera instancia, parecía estar correcto. Sin embargo, al cambiar el orden de la entrada, lo cual sigue considerándose una entrada válida, el método fallaba. Esto se debía a que no se estaba realizando el manejo adecuado de las entradas antes de mandarlas al método en la blockchain. Al notar esto, se agregó un preprocesamiento que ordena las entradas en base a la definición del método, la cual se encuentra almacenada en la base de datos en la ABI del contrato.

**6.3.2.3.2 Entradas inválidos** Ante ingresos inválidos de las entradas, ya sea por falta de alguna o el uso incorrecto de tipo, el método ejecutó los errores adecuados y no se necesitaron realizar más cambios.

## 6.4 Análisis Estático

Se realizaron múltiples análisis para iterar sobre las posibles correcciones que el analizador pudo encontrar. Además de informar sobre los errores y potenciales mejoras, informa también como realizar las correcciones e implementar las mejoras, lo que facilita y acelera la corrección del código.

### 6.4.1 Resultados

En lo que respecta a errores (bugs) en el código, Sonarqube encontró unos pequeños errores que no fueron detectados previamente. Estos errores estaban principalmente relacionados a no tener en cuenta que posiblemente algún parámetro sea nulo o de tipos en TypeScript; los mismos fueron rápidamente corregidos.

En términos de code smells (estilo o características del código que el analizador encuentra según las reglas que tiene definidas y clasifica como negativas), el analizador encontró una gran cantidad, principalmente debido a la poca experiencia que se tiene programando en TypeScript. Lo que encontró eran recomendaciones de tipos faltantes en las declaraciones, unificación del estilo del código, complejidad cognitiva de las funciones y potenciales refactorizaciones y no utilizar constantes cuando la variable no se modifica, entre varios otros. Estos cambios en general son pequeños, por lo que fueron un conjunto de ítems que se pudieron resolver velozmente.

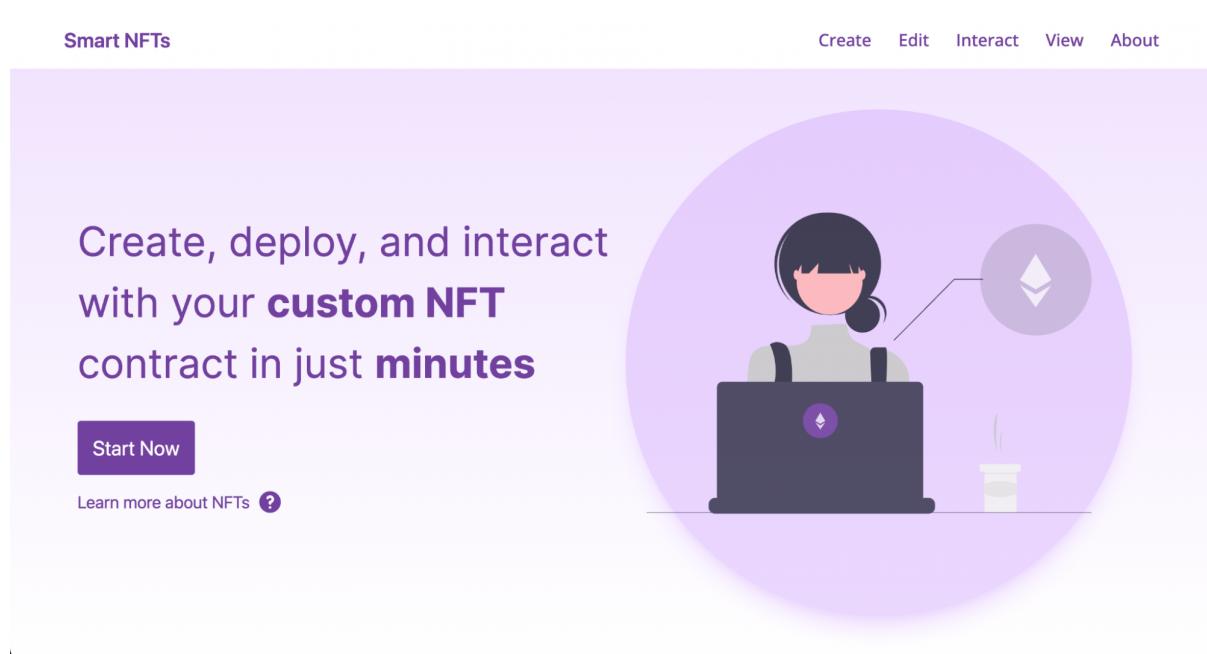
Utilizar el analizador ayudó a encontrar y resolver problemas, mejorando la mantenibilidad del código y reduciendo los errores que podrían surgir durante la ejecución.

## 7 Aplicación Desarrollada

El sistema se encuentra internacionalizada con idiomas Inglés y Español; el idioma a utilizar se define según el idioma predefinido en el buscador.

Al entrar a la aplicación web en la URL <https://smart-nfts.gonzalohirsch.com/>, los usuarios se encuentran con la pantalla de inicio, como se puede ver en la Figura 7.1. Ahí se encuentran con las siguientes opciones:

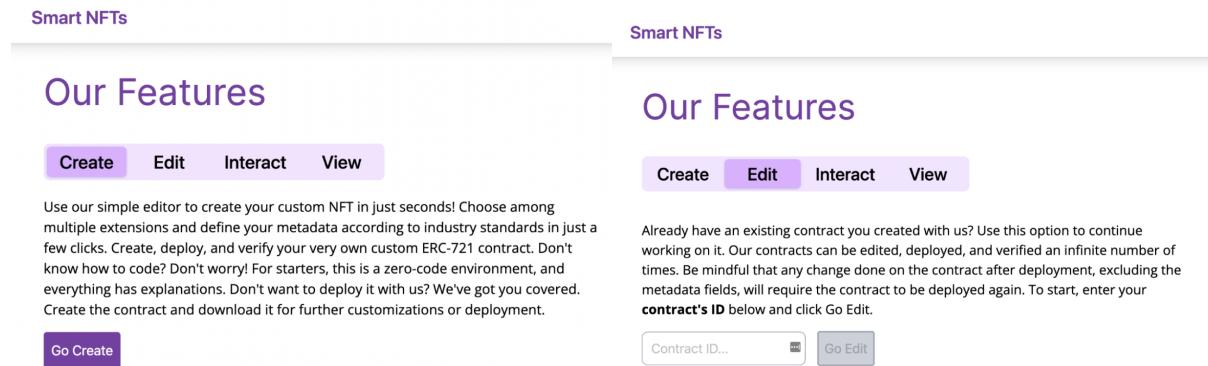
- Comenzar con el proceso de creación de un contrato (“*Start Now*” o “*Create*”).
- Aprender más sobre NFTs (“*Learn more about NFTs (?)*” o “*About*”).
- Editar un contrato (“*Edit*”).
- Interactuar con un contrato (“*Interact*”).
- Visualizar los tokens creados con un contrato (“*View*”).



**Figura 7.1:** Página principal de Smart NFTs.

### 7.1 Selección de Opciones

Al seleccionar “*Start Now*” la página ofrece las opciones para crear, editar, interactuar o visualizar los tokens de un contrato, como se puede ver en la Figura 7.2.



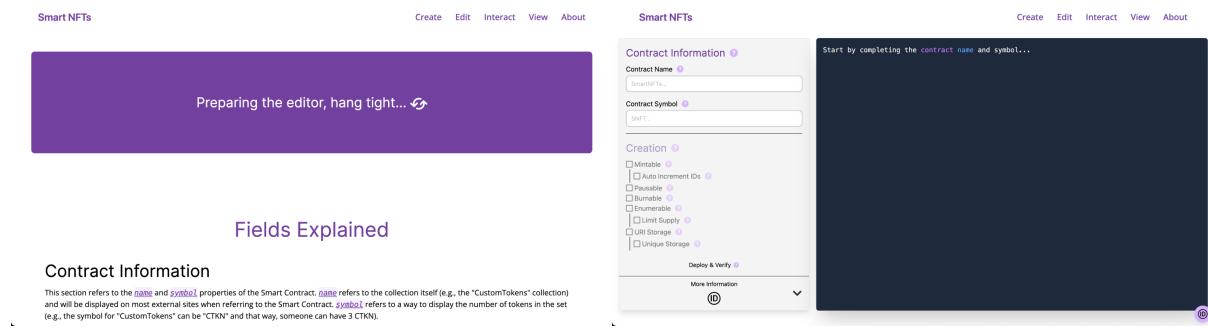
**Figura 7.2:** Conjunto de opciones para operar con los proyectos de contratos.

Si se desea crear un contrato, la opción de “*Create*” permite comenzar un proyecto nuevo. Si ya se posee un contrato en progreso, la opción de “*Edit*” permite que el usuario ingrese el ID del contrato para continuar editando. Las opciones de “*Interact*” y “*View*” sirven para interactuar con un contrato publicado o visualizar los tokens de un contrato publicado, respectivamente.

Las opciones que poseen un campo para completar el ID del contrato permiten hacer click en el respectivo botón acompañante una vez que se completa el campo del ID.

## 7.2 Crear un Contrato

Al seleccionar la opción “*Go Create*” de la Figura 7.2 se comenzará a cargar el editor y se mostrará el editor listo para ser utilizado, como puede verse en la Figura 7.3.



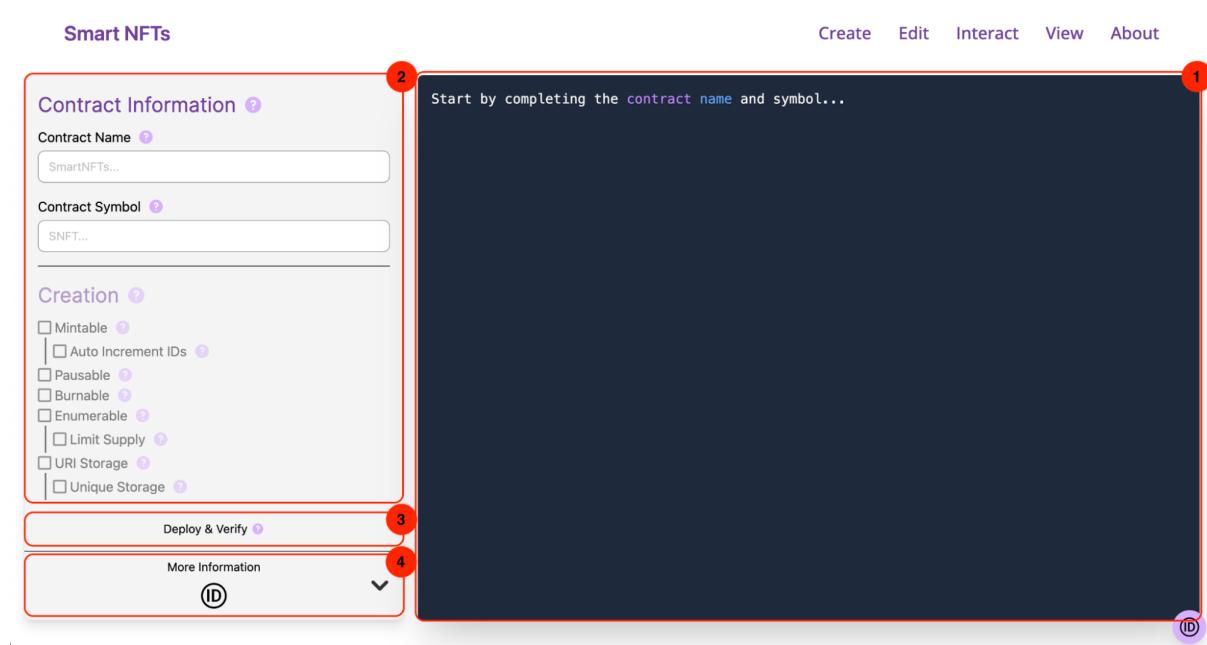
**Figura 7.3:** Pantalla de carga del editor de contrato.

El editor posee 4 secciones principales, señaladas en la Figura 7.4.

1. Panel de previsualización del contrato, contiene el código autogenerado en tiempo

real.

2. Formulario para definición del contenido del contrato. Contiene campos de edición de la información básica, selección de extensiones y definición de metadata (no visible en la versión básica a menos que se seleccione la extensión “*URI Storage*”).
3. Sección de acciones de contrato, posee las opciones de publicación o validación de un contrato.
4. Sección de información extra de contrato, posee información como fecha de edición, dirección de publicación y envío de recordatorios, entre otras.



**Figura 7.4:** Editor de contrato, inicialmente vacío, con secciones señaladas.

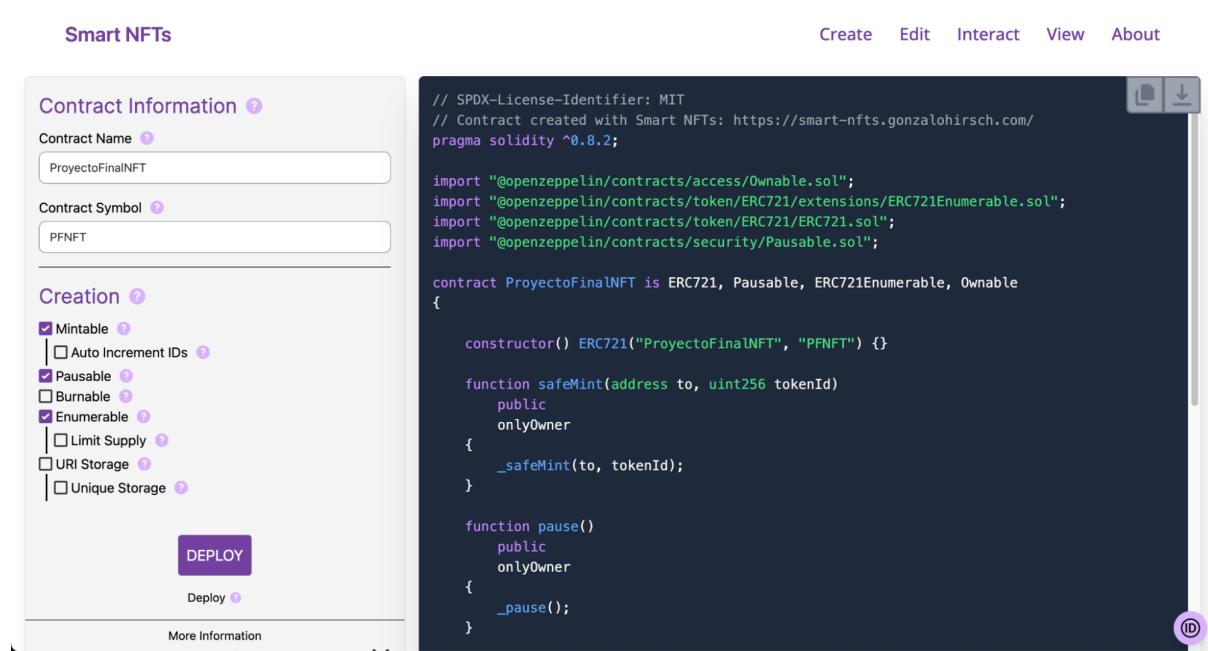
El contrato en sí ya se encuentra creado al entrar al editor, ya se provee el ID con el que se puede recuperar al mismo.

## 7.3 Editar un Contrato

Para llegar a la edición de un contrato se pueden utilizar los métodos y opciones descritas anteriormente y visibles en la Figura 7.1 y Figura 7.2. Dichos métodos son mediante la barra de navegación o las pestañas de selección, respectivamente.

La edición se comienza completando el “*Contract Name*” y “*Contract Symbol*”, como instruye la previsualización del contrato en la Figura 7.4. Una vez completo eso se

habilita la selección de extensiones, como se puede ver en la Figura 7.5, donde se seleccionaron extensiones a modo ilustrativo.



The screenshot shows the Smart NFTs web interface. At the top, there are navigation links: Create, Edit, Interact, View, and About. Below the header, the title "Smart NFTs" is displayed. On the left, there's a sidebar with "Contract Information" and "Creation" sections. In the "Creation" section, several checkboxes are checked: Mintable, Pausable, and Enumerable. Under Enumerable, "Limit Supply" is also checked. There are also options for Auto Increment IDs, Burnable, URI Storage, and Unique Storage. A large purple "DEPLOY" button is prominent. On the right side, the Solidity code for the contract is shown:

```
// SPDX-License-Identifier: MIT
// Contract created with Smart NFTs: https://smart-nfts.gonzalohirsch.com/
pragma solidity ^0.8.2;

import "@openzeppelin/contracts/access/Ownable.sol";
import "@openzeppelin/contracts/token/ERC721/extensions/ERC721Enumerable.sol";
import "@openzeppelin/contracts/token/ERC721/ERC721.sol";
import "@openzeppelin/contracts/security/Pausable.sol";

contract ProyectoFinalNFT is ERC721, Pausable, ERC721Enumerable, Ownable
{
    constructor() ERC721("ProyectoFinalNFT", "PFNFT") {}

    function safeMint(address to, uint256 tokenId)
        public
        onlyOwner
    {
        _safeMint(to, tokenId);
    }

    function pause()
        public
        onlyOwner
    {
        _pause();
    }
}
```

**Figura 7.5:** Configuración ejemplo de un contrato con extensiones habilitadas.

Una vez habilitadas las extensiones se pueden combinar las selecciones. Es necesario notar que las extensiones que se encuentran anidadas dentro de otra necesitan a la extensión en el nivel superior; al seleccionar una extensión anidada, la extensión padre se selecciona de manera automática. En caso de necesitar ayuda sobre algún campo o extensión, se pueden utilizar los íconos de signo de pregunta para buscar la información extra.

## 7.4 Definir Metadata

Es necesario seleccionar la extensión “*URI Storage*” para activar el panel de definición de metadata, como se puede ver en la Figura 7.6.

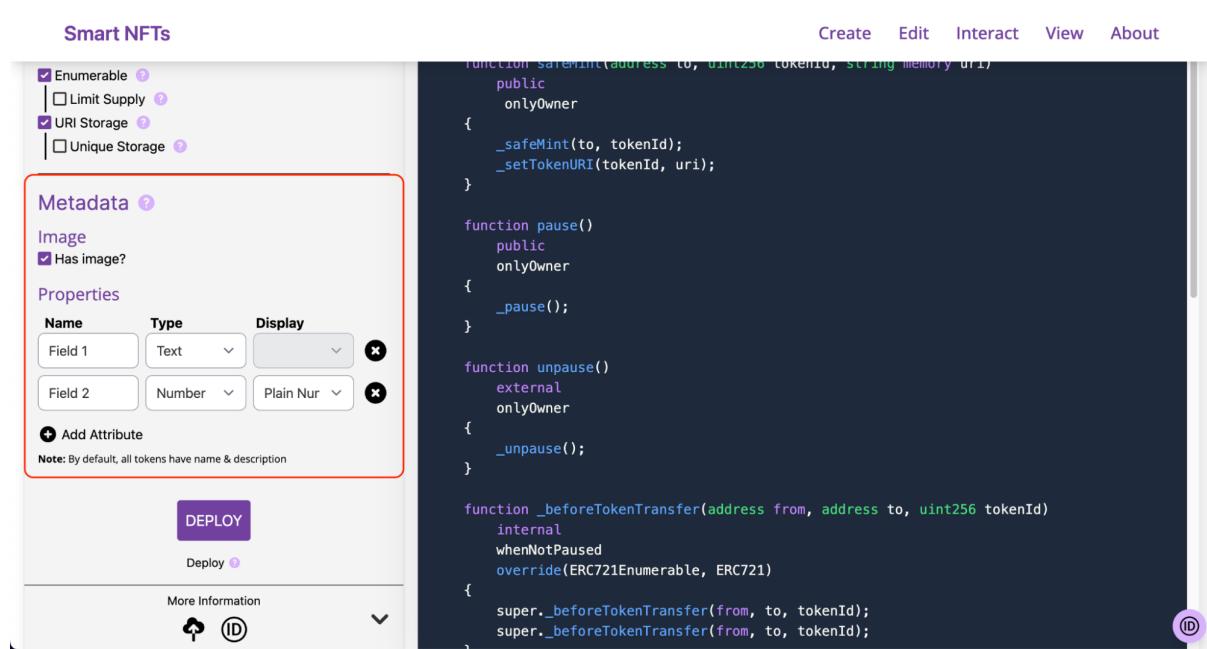


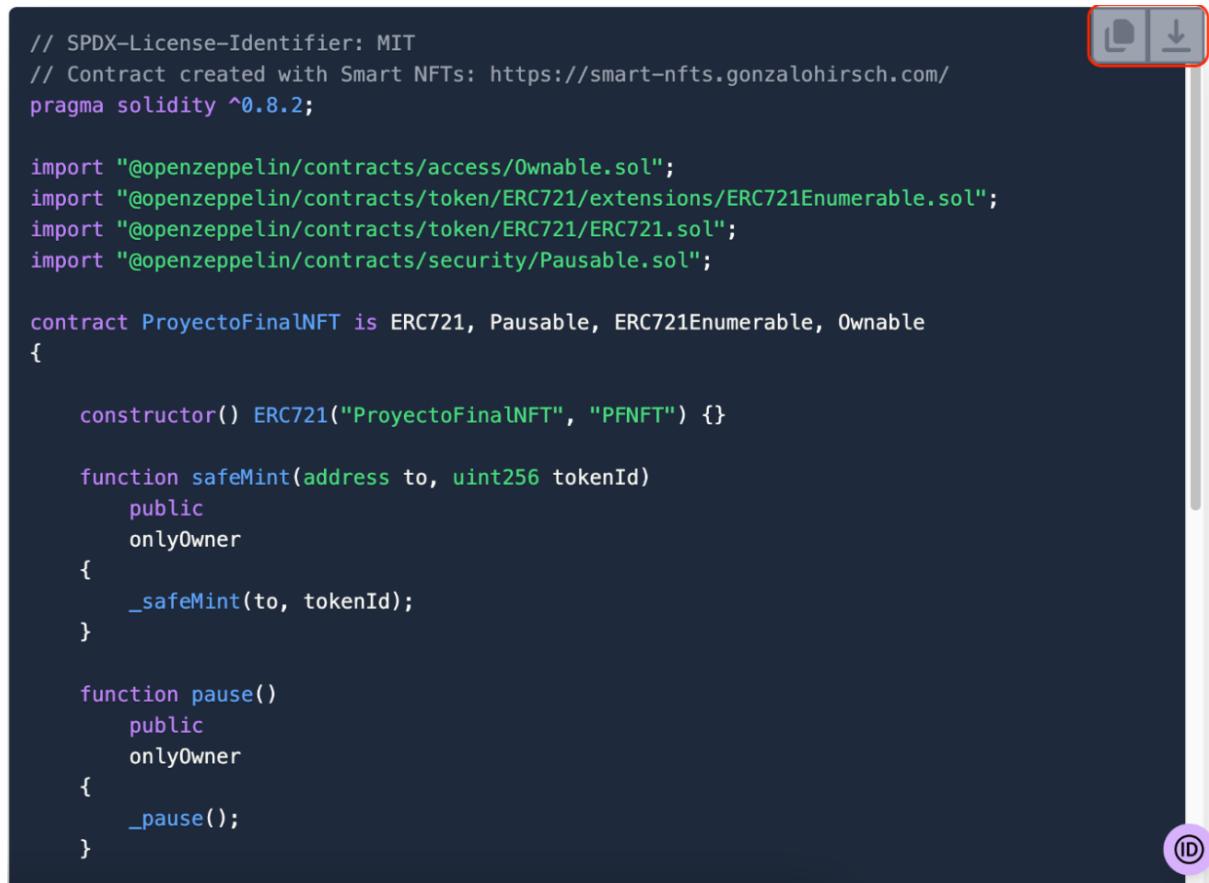
Figura 7.6: Panel de configuración de metadata activo.

En caso de necesitar que los tokens incluyan una imagen, se selecciona la opción “*Has Image?*”. La definición de las propiedades de la metadata se hace a través del botón “*Add Attribute*”; la cruz a la derecha de cada campo se utiliza para remover el campo en sí.

Los campos pueden ser de dos tipos, textuales o numéricos. Asimismo, los campos numéricos pueden ser de tres tipos, planos, numéricos porcentuales y numéricos de aumento. Todos los tipos se encuentran explicados en la sección de metadata de la ayuda acompañante del editor. Es necesario notar que por defecto, al incluir metadata, los tokens ya poseen un campo de nombre y descripción. Los cambios en la configuración de metadata no generan modificaciones en el panel de previsualización del contrato.

## 7.5 Copiar o Descargar el Contrato

Se proveen botones para la copia o descarga del contrato desde el panel de previsualización del contrato. En la esquina superior derecha de la previsualización se encuentra un botón para la copia del código al portapapeles y la descarga del contrato en un archivo ZIP que lo contiene en un archivo “.sol”. Los botones se encuentran señalados en la Figura 7.7.



```
// SPDX-License-Identifier: MIT
// Contract created with Smart NFTs: https://smart-nfts.gonzalohirsch.com/
pragma solidity ^0.8.2;

import "@openzeppelin/contracts/access/Ownable.sol";
import "@openzeppelin/contracts/token/ERC721/extensions/ERC721Enumerable.sol";
import "@openzeppelin/contracts/token/ERC721/ERC721.sol";
import "@openzeppelin/contracts/security/Pausable.sol";

contract ProyectoFinalNFT is ERC721, Pausable, ERC721Enumerable, Ownable
{
    constructor() ERC721("ProyectoFinalNFT", "PFNFT") {}

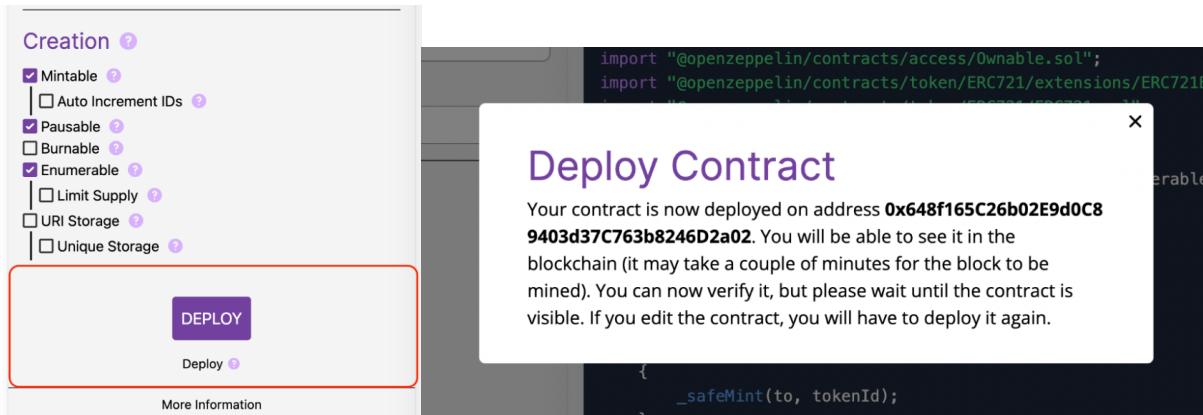
    function safeMint(address to, uint256 tokenId)
        public
        onlyOwner
    {
        _safeMint(to, tokenId);
    }

    function pause()
        public
        onlyOwner
    {
        _pause();
    }
}
```

Figura 7.7: Botones de copia y descarga de contrato respectivamente.

## 7.6 Publicar un Contrato

Para comenzar el proceso de publicación de contrato es necesario hacer click en el botón “Deploy”, que puede verse en la Figura 7.8a. Una vez que se hace click, se comienza el proceso y se puede visualizar en un modal que aparece. Luego de terminar se muestra el mensaje de la Figura 7.8b, donde indica la dirección donde se encuentra publicado el contrato y como proseguir.



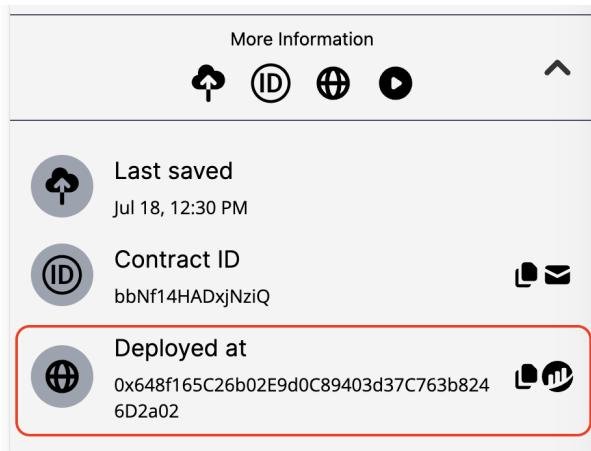
(a) Botón de publicación de contrato.

(b) Modal de confirmación.

**Figura 7.8:** Conjunto de capturas relacionadas a la publicación de un contrato.

Una vez publicado el contrato aparece un ícono en el editor para acceder al contrato a través de Etherscan o copiar la dirección, como se puede ver en la Figura 7.9. Al hacer cambios en el contrato, es necesario volver a publicarlo para poder utilizar los cambios en la interacción.

En caso de necesitar más información sobre la publicación, se puede usar el botón con el signo de pregunta provisto debajo del botón “*Deploy*”.

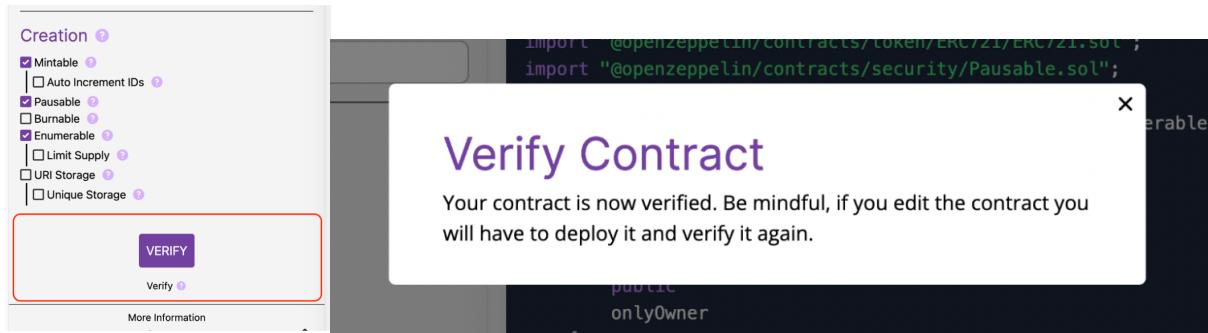


**Figura 7.9:** Dirección de publicación del contrato.

## 7.7 Verificar un Contrato

Para comenzar el proceso de verificación, es necesario primero haber publicado el contrato y también es recomendable esperar mínimo 5 minutos entre la publicación y verificación,

de manera que el servicio externo de verificación pueda encontrarlo. Si se encuentra habilitada la opción, se comienza el proceso haciendo click en el botón “*Verify*”, visible en la Figura 7.10a. Al igual que el proceso de publicación, esto muestra un modal con el progreso y un mensaje al terminar, como se puede ver en la Figura 7.10b.



(a) Botón de verificación de contrato.

(b) Modal de confirmación.

**Figura 7.10:** Conjunto de capturas relacionadas a la verificación de un contrato.

Una vez verificado el contrato, se puede ver en el editor un ícono que indica que se encuentra en efecto verificado (Figura 7.11) y también se puede comprobar en Etherscan, como se muestra en la Figura 7.12. En caso de verificar un contrato y luego publicar una nueva versión, se debe verificar la nueva versión del contrato.



**Figura 7.11:** Ícono de verificación exitosa en la barra de acceso rápido del editor.

The screenshot shows the Etherscan interface for a Solidity contract. At the top, it displays the contract address: 0x648f165C26b02E9d0C89403d37C763b8246D2a02. Below this, the 'Contract Overview' section shows a balance of 0 Ether. The 'More Info' section includes fields for 'My Name Tag' (Not Available) and 'Contract Creator' (0xa3d9e7d1c697edb3). The navigation bar at the bottom has tabs for 'Transactions', 'Erc20 Token Txns', 'Contract' (which is selected), and 'Events'. Under the 'Contract' tab, there are buttons for 'Code', 'Read Contract', and 'Write Contract'. A note indicates that the contract source code is verified (Exact Match). Below this, details about the contract are listed: Contract Name (ProyectoFinalNFT), Compiler Version (v0.8.11+commit.d7f03943), Optimization Enabled (No with 200 runs), and Other Settings (default evmVersion). A link to the contract's Solidity source code is also provided.

**Figura 7.12:** Captura de pantalla de Etherscan, mostrando que el contrato se encuentra verificado.

En caso de necesitar más información sobre la verificación, se puede usar el botón con el signo de pregunta provisto debajo del botón “*Verify*”.

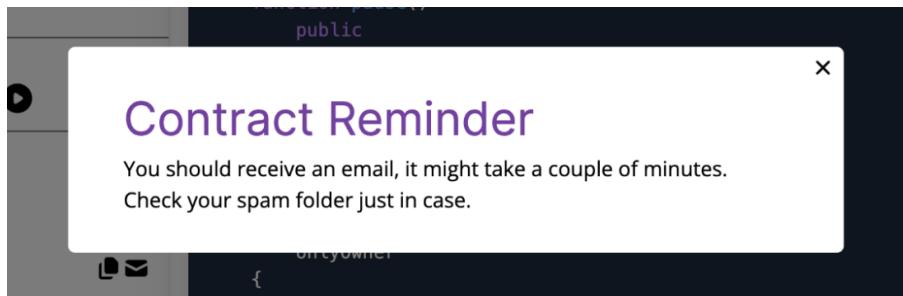
## 7.8 Enviar un Recordatorio

Para enviar un recordatorio se puede utilizar el botón de envío de email en la sección de acciones, como se puede ver en la Figura 7.13. Este botón activa un modal en el que uno puede completar su email, también Figura 7.13.

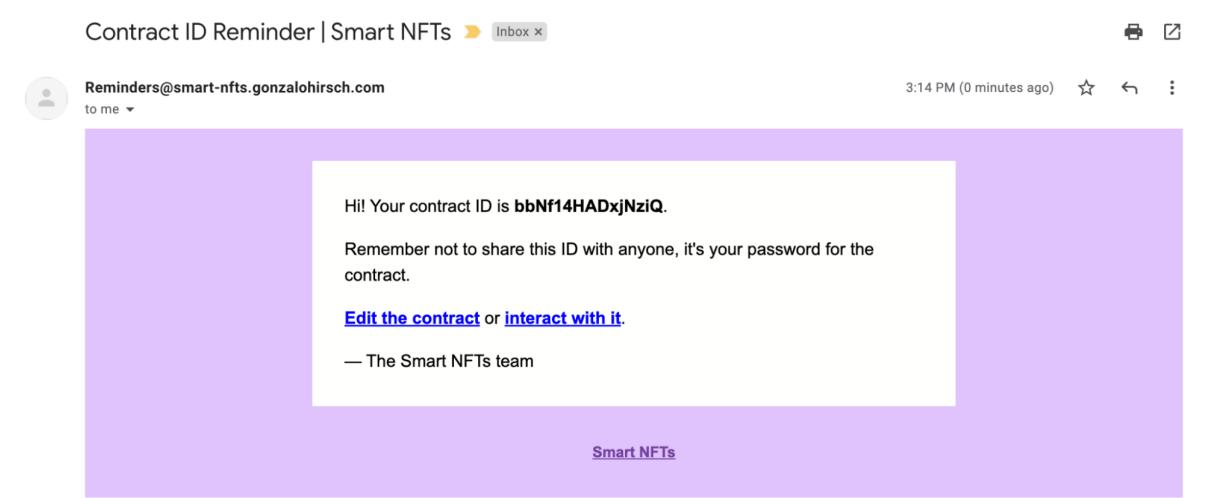
The screenshot shows a user interface for sending a contract reminder. On the left, there is a sidebar with icons for file operations (Upload, ID, Deploy, Run, Verify, Send Email) and information about the last save (Jul 18, 12:30 PM), the contract ID (bbNf14HADxjNziQ), and the deployment address (0x648f165C26b02E9d0C89403d37C763b8246D2a02). On the right, a modal window titled "Contract Reminder" is displayed. It contains a message stating: "We'll send a reminder email with your contract ID. Be careful which email you choose, you can send reminders every 8hs." Below this is an input field with the placeholder "example@example.com" and a "SEND EMAIL" button.

**Figura 7.13:** Conjunto de capturas que muestran el botón de envío de emails y el modal para completar con la dirección de correo electrónico.

Una vez completada la dirección de correo electrónico, se puede hacer el envío del email. Eso muestra el progreso y su resultado en un modal, como se puede ver en la Figura 7.14. El correo en sí suele llegar en algunos segundos, como se muestra en la Figura 7.15.



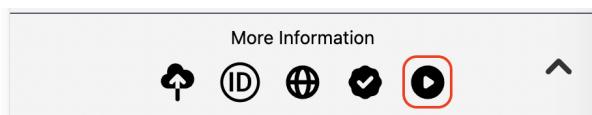
**Figura 7.14:** Modal de confirmación para el envío del correo electrónico.



**Figura 7.15:** Ejemplo de correo electrónico enviado por el sistema.

## 7.9 Interactuar con Contrato

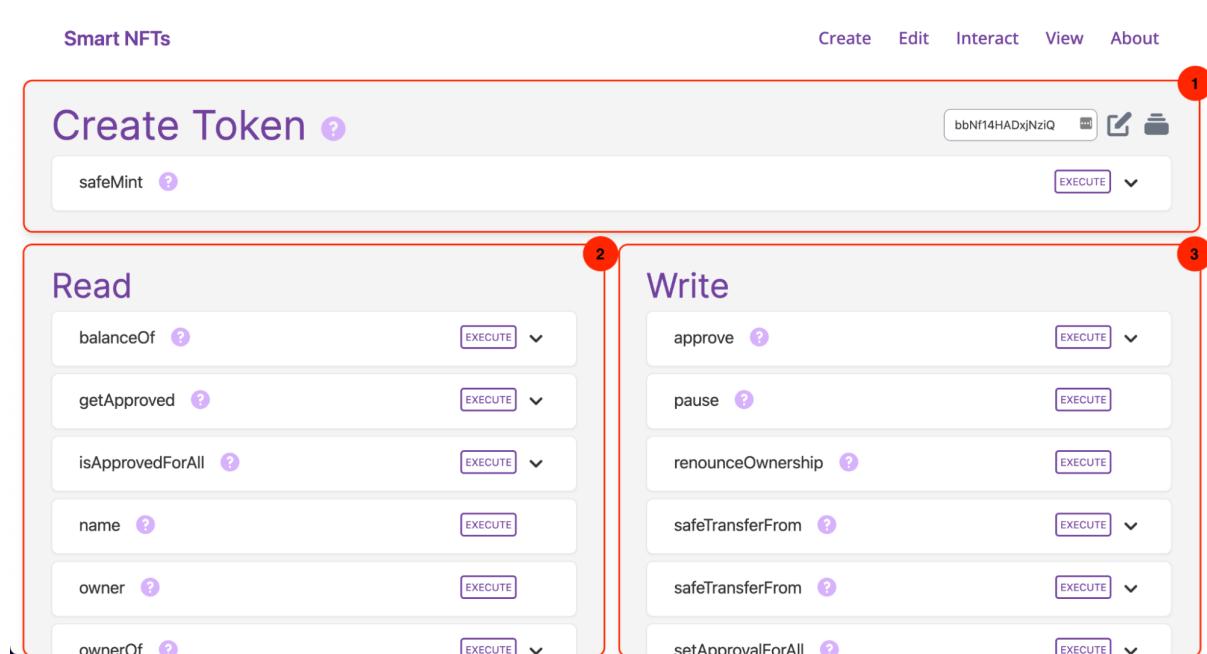
Para poder interactuar con un contrato es necesario haberlo publicado antes. Se puede alcanzar la sección de interacción a través de 3 métodos, desde la barra de navegación con “*Interact*”, desde la opción descrita en la sección “Selección de Opciones” del manual de uso o desde el editor con el atajo, visible en la Figura 7.16.



**Figura 7.16:** Botón para navegación rápida a la sección de interacción.

La página de interacción posee 3 áreas principales, señaladas en la Figura 7.17:

1. Panel de creación de tokens; es la sección que contiene los métodos que se utilizan para generar tokens. No aparece el método “safeMint” si el contrato no se configura de esa manera.
2. Panel de métodos de lectura; es la sección que contiene sólo métodos que realizan acciones de lectura sobre el contrato.
3. Panel de métodos de escritura; es la sección que contiene sólo métodos que realizan operaciones de escritura sobre el contrato.

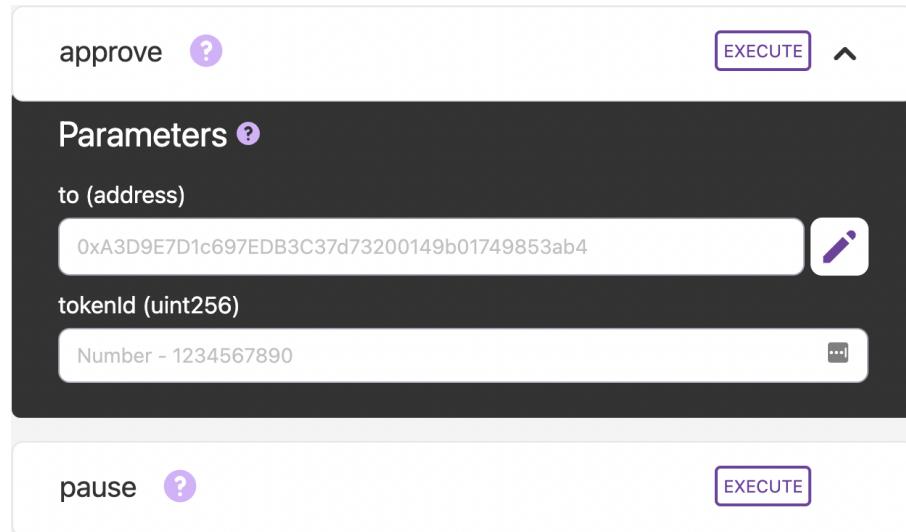


**Figura 7.17:** Pantalla de interacción, con las secciones señaladas.

En caso de querer cambiar el contrato para interactuar, solo hace falta reemplazar el ID de contrato en el campo que se encuentra en la esquina superior derecha de la sección de creación de tokens (sección 1).

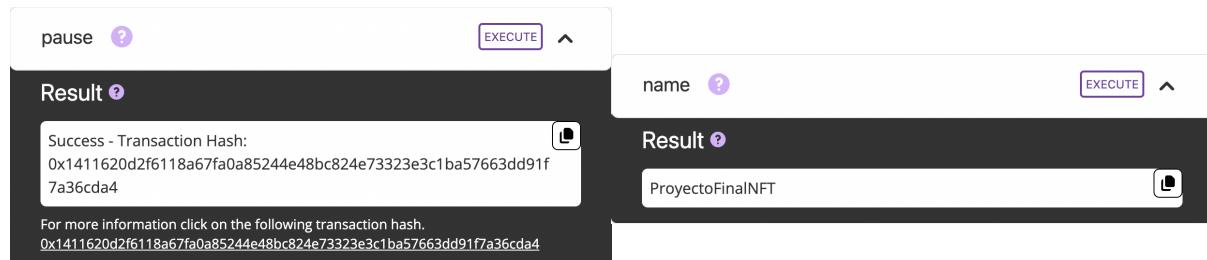
## 7.10 Ejecutar un Método

Los métodos se encuentran definidos según las extensiones que se eligen para el contrato. Algunos métodos pueden tener o no parámetros, eso se puede observar cuando no poseen una flecha en el acordeón, como se puede ver en la Figura 7.18. Por ende, no se pueden expandir métodos que no tengan parámetros.



**Figura 7.18:** Comparación de método con y sin parámetros, respectivamente.

La ejecución de los métodos se puede hacer, una vez completados y validados los parámetros, con el botón “*Execute*”. Al obtener un resultado, se habilita la extensión del acordeón (si es que el método no lo permitía) y aparece una sección de resultados en donde se indica si fue una ejecución exitosa o mostrando el resultado, dependiendo de si el método es de escritura o lectura respectivamente. Se puede ver en la Figura 7.19 un ejemplo de ambos tipos de resultados. En el caso de ejecuciones de métodos de escritura, se proveen los hash de las transacciones para poder buscar las ejecuciones en la blockchain.



**Figura 7.19:** Ejemplos de ejecuciones de métodos de escritura y lectura.

En caso de necesitar ayuda sobre los métodos, los parámetros o las secciones que aparecen al expandir el acordeón, se pueden utilizar los botones con signos de pregunta para buscar la ayuda correspondiente.

Los errores de ejecución de métodos pueden incluir hash de transacción en caso de ser métodos de escritura o simplemente un mensaje de error en caso de ser métodos de lectura. En la Figura 7.20 se comparan ambos tipos de errores.

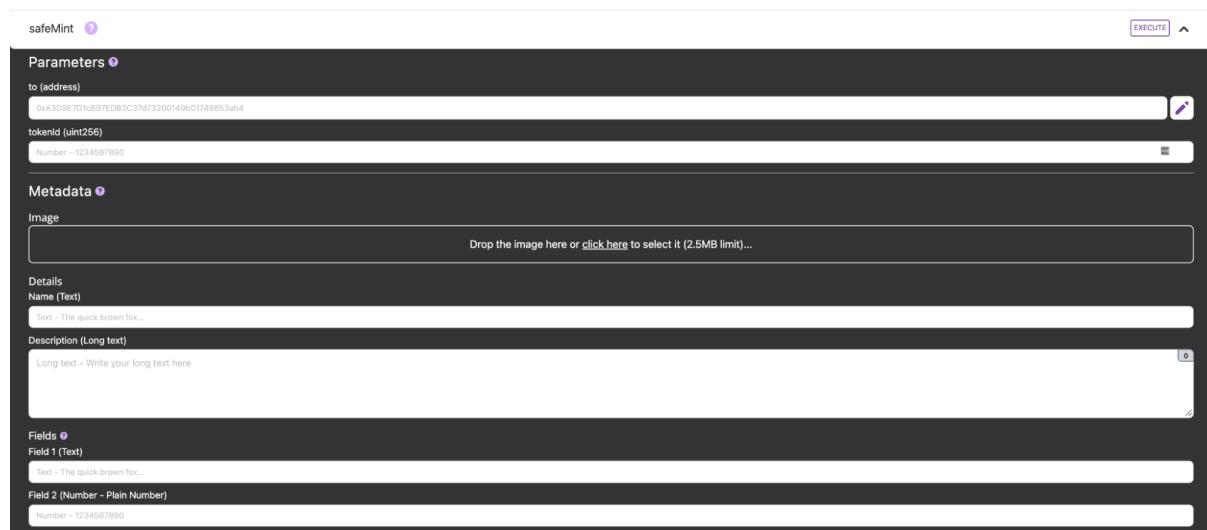


**Figura 7.20:** Ejemplos de ejecuciones que retornan errores para métodos de escritura (arriba) y lectura (abajo).

## 7.11 Crear un Token

Para esto es necesario que el contrato incluya la extensión “*Mintable*”, que permite emitir tokens. En caso de tenerla, el proceso para la creación de un token es el mismo para la ejecución de un método, como fue descrito anteriormente. En caso de que el contrato posea una configuración de metadata, los campos se completan y validan previo a la ejecución del método. De lo contrario, si el contrato no posee configuración de metadata, la sección de metadata no aparecerá. A continuación, en la Figura 7.21, se puede ver un ejemplo de método para crear un token.

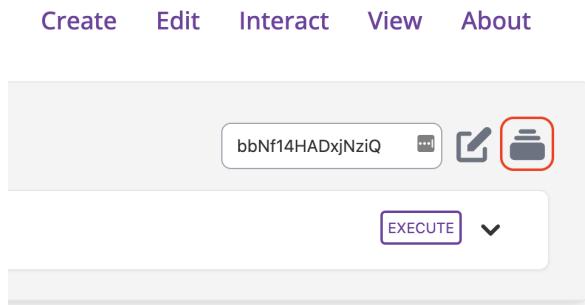
En caso de haber incluido la extensión “*Unique Storage*”, no se permitirá que se emita un token con las mismas propiedades que uno ya existente, lo que generará un error en la blockchain.



**Figura 7.21:** Ejemplo de método para creación de tokens.

## 7.12 Visualizar Tokens

Para esto es necesario que el contrato incluya la extensión “*Mintable*”, que permite emitir tokens, de lo contrario no se habilita la pantalla de visualización de tokens. Se puede alcanzar la sección de visualización a través de 3 métodos, desde la barra de navegación con “*View*”, desde la opción descrita en la sección “Selección de Opciones” del manual de uso o desde la pantalla de interacción con el atajo, como se muestra en la Figura 7.23, ubicado en la esquina superior derecha.



**Figura 7.22:** Botón de visualización dentro del panel de interacción de contrato.

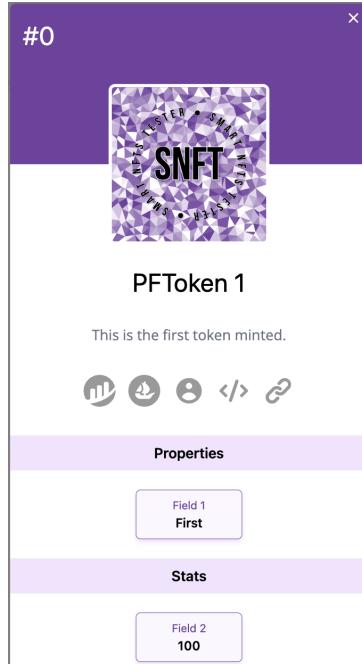
Al entrar a la página de visualización se mostrarán los tokens emitidos. La página se encuentra paginada, por lo que de ser necesario, en la parte inferior de la misma se encuentran controles para controlar el número de página. En la Figura 7.23 se encuentra un ejemplo de como se ve la página de visualización con un token emitido.



**Figura 7.23:** Pantalla de visualización de tokens.

## 7.13 Detalles de un Token

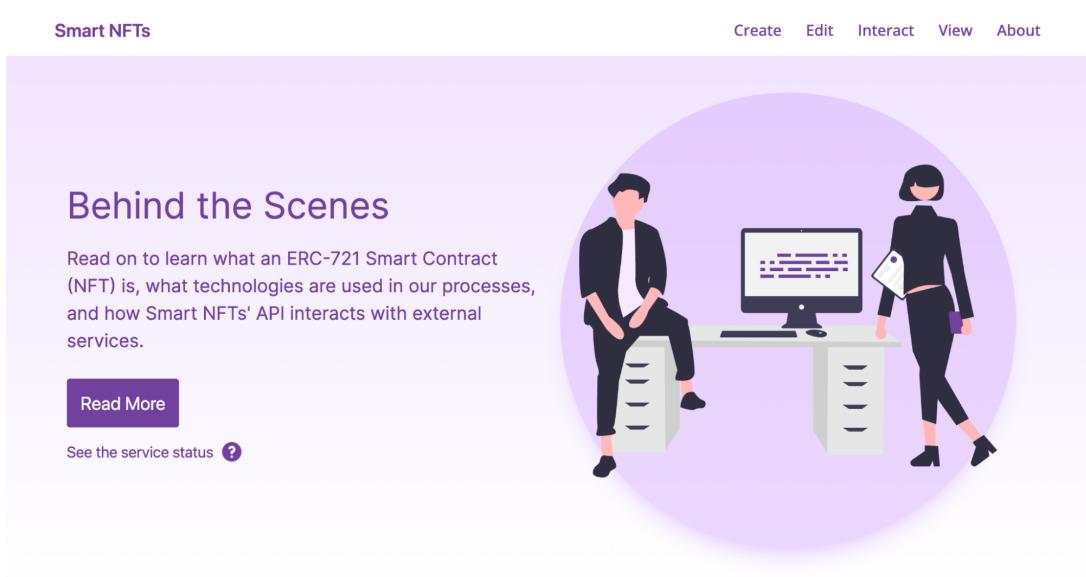
Estando en la pantalla de visualización de un token se pueden expandir los detalles del mismo haciendo click en la tarjeta. En el modal que surge se pueden ver todas las propiedades de metadata que se definieron, junto con hipervínculos e información extra que puede ser útil. Todo esto es visible en la Figura 7.24.



**Figura 7.24:** Modal expandido con detalles del token emitido.

## 7.14 Buscar más Información

Se puede buscar más información sobre la página y los NFTs haciendo click en “About” en la barra de navegación. Esto lleva al usuario a la página con información sobre NFTs y detalles sobre la página. En la Figura 7.25 se puede observar la sección principal de esa página.



**Figura 7.25:** Sección principal de la página con información extra.

Al bajar en dicha página se puede encontrar más información.

En caso de requerir ayuda sobre los contenidos de las diferentes secciones del proceso de creación del contrato, es decir, ayuda con la creación del contrato o interacción, se pueden utilizar los botones que contienen un signo de interrogación (?). Dichos botones llevan al usuario a la información de ayuda, localizada en la parte inferior de la pantalla correspondiente.

## 7.15 Verificar Estado del Servicio

Para esto es necesario hacer click en el texto “*See the service status*” en la sección principal ilustrada en la Figura 7.25. Esto muestra la página de estado de servicio, que incluye información como la billetera virtual usada, la red en la que se publican los contratos y el momento de verificación.

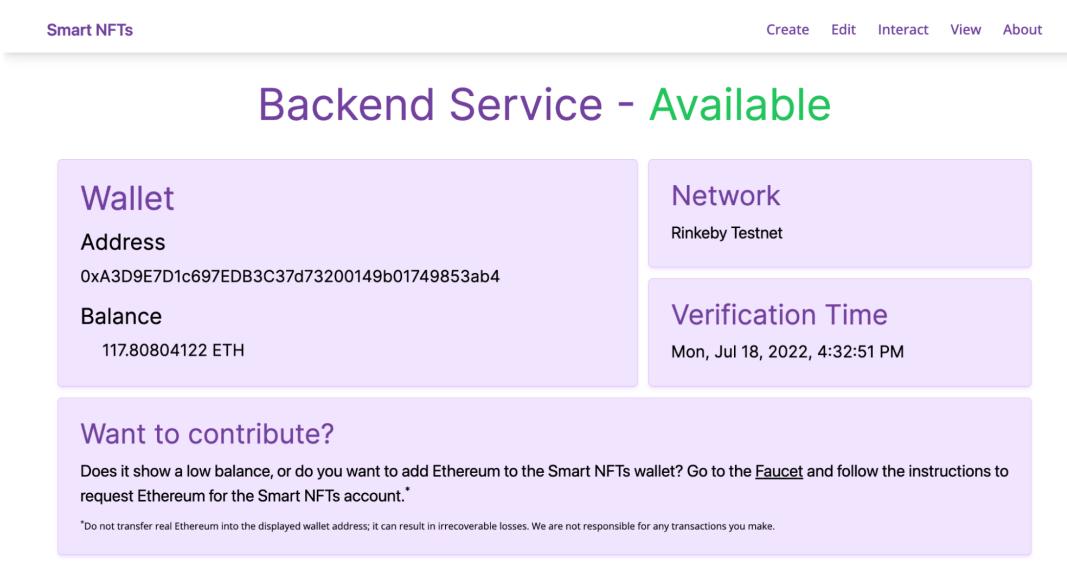


Figura 7.26: Página informativa para el estado del servicio.

## 7.16 Visualización de Analítica

### 7.16.1 Google Analytics

Para visualizar los reportes de analítica generados en Google Analytics se puede navegar al tablero de comandos (dashboard) de Google Analytics. Dentro del tablero se puede utilizar la sección de “*Reports*” para visualizar reportes varios sobre el uso de la aplicación y adquisición de usuarios, como es ilustrado en la Figura 7.27.

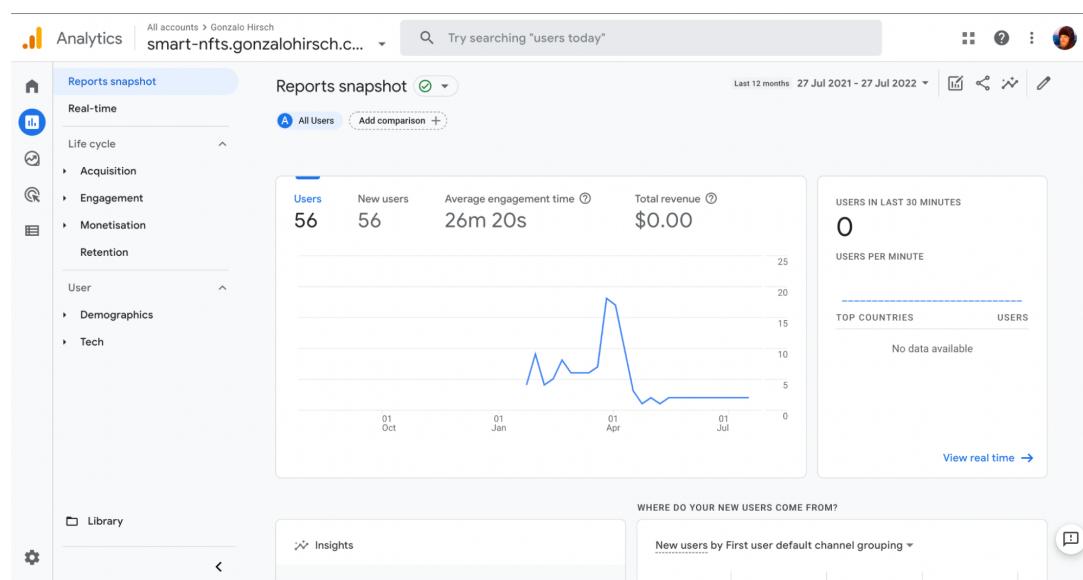


Figura 7.27: Tablero de comando de Google Analytics.

### 7.16.2 Google Search Console

Para visualizar los reportes de analítica generados por Google Search Console se puede navegar al centro de reportes de Google Search Console, específicamente a la sección de “*Performance*”. Dentro de esa sección se pueden visualizar diferentes métricas sobre las búsquedas e impresiones que tuvo el sitio. La Figura 7.28 ilustra el resultado visualizado en la sección de “Performance”.

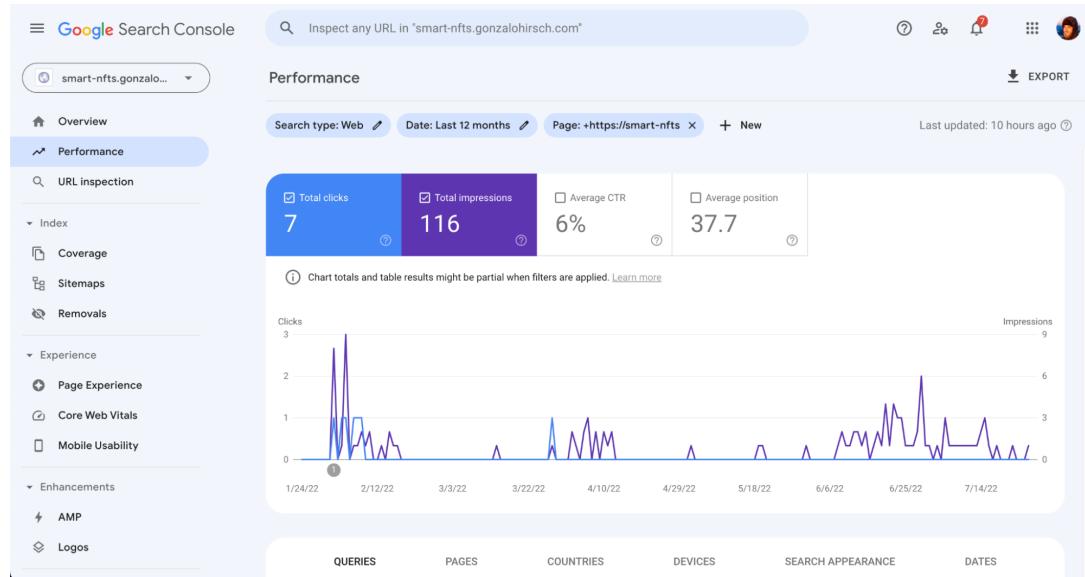


Figura 7.28: Sección de desempeño de Google Search Console.

## 8 Conclusiones y Trabajo Futuro

En el presente trabajo se presentó un sistema informático desde su diseño hasta su puesta en producción, incluyendo todas las etapas intermedias involucradas en el desarrollo. Su idea fue resolver un problema del mundo real, teniendo un posible impacto en el ecosistema de los NFT. Mientras la idea fue validada al tener aplicaciones competidoras aparecer en el mercado meses luego de haber concebido la solución, dicha validación enseña el valor, especialmente en la actualidad, de un ciclo de desarrollo ordenado e incremental que aporta valor continuamente a un producto.

La aplicación desarrollada cumple con el objetivo propuesto inicialmente, con los requerimientos tecnológicos definidos y los 13 requisitos funcionales especificados. El principal beneficio de la misma se puede resumir en la simplificación de los diferentes procesos involucrados en la creación y uso de contratos ERC-721. Dicha simplificación, que incluye los procesos de publicación, interacción y visualización entre otros, permite democratizar el mercado de los NFT, ya que el desarrollo de los mismos se torna más accesible para usuarios novicios. La solución integra la generación de contratos y sus explicaciones pertinentes, poniendo el foco en el contrato en sí, además de proveer la simpleza de no necesitar billeteras virtuales o criptomonedas, proveyendo un entorno perfecto e intuitivo para que los usuarios puedan explorar los contratos ERC-721. A diferencia de las alternativas exploradas, la aplicación desarrollada centraliza todos los procesos mencionados anteriormente, sumando múltiples opciones de personalización de los contratos ERC-721, de manera gratuita y a través de una interfaz sumamente intuitiva.

La solución desarrollada da lugar a poder continuar con la misma, el ecosistema al que pertenece es uno en constante crecimiento y evolución, es por eso que la innovación de los productos es un factor clave. Como una primera aproximación a las posibles extensiones se puede pensar en agregar:

- Posibilidad de agregar descripciones, títulos y/o tags a los proyectos de contrato.
- Sistema de usuarios, para permitir que los usuarios interesados puedan encontrar sus proyectos de contratos con mayor facilidad.
- Niveles de privacidad para proyectos, permitiendo que los usuarios definan si los

contratos diseñados son públicos o privados.

- Búsqueda de proyectos públicos, para poder encontrar o interactuar con proyectos creados por otros, utilizando las descripciones, títulos o tags definidos anteriormente.
- Permitir más tipos de contenido en los contratos, por ejemplo videos MP4 o archivos PDF.

El trabajo futuro no se ve limitado meramente a los ítems mencionados, sinó que también incluye el trabajo de mantenimiento e intercambio continuo con usuarios para asegurarse de que la experiencia siga siendo óptima.

## 9 Referencias

Google. (2020–2022, July 1–August 1). Google Trends [NFTs - Interest Over Time]. URL: <https://trends.google.com/trends/explore?date=2020-07-01%202022-08-01&q=nfts>

Luke Dormehl. (2021, March 10). A brief history of NFTs. Digital Trends. URL: <https://www.digitaltrends.com/computing/what-are-nfts-non-fungible-tokens-history-explained/>

William Entriken, Dieter Shirley , Jacob Evans , Nastassia Sachs. (2018, January 24). EIP-721: Non-Fungible Token Standard. Ethereum Improvement Proposals. URL: <https://eips.ethereum.org/EIPS/eip-721>

Yoni Assia. (2012, March 27). bitcoin 2.X (aka Colored Bitcoin) - initial specs. URL: <https://yoniassia.com/coloredbitcoin/>