Trabalho de Conclusão de Curso

Uma Aplicação Descentralizada (dApp / Web3.0) para Registros de Autorias em Blockchain

Lucas dos Santos Sales
ldss@ic.ufal.br

Orientador:

DSc. Leandro Melo de Sales

Maceió, Outubro de 2022

Lucas dos Santos Sales

Uma Aplicação Descentralizada (dApp / Web3.0) para Registros de Autorias em Blockchain

Monografia apresentada como requisito parcial para obtenção do grau de Bacharel em Ciências de Computação do Instituto de Computação da Universidade Federal de Alagoas.

Orientador:

DSc. Leandro Melo de Sales

Maceió, Outubro de 2022

Monografia apresentada como requisito parcial para obtenção do grau de Bacharel em Engenharia de Computação do Instituto de Computação da Universidade Federal de Alagoas, aprovada pela comissão examinadora que abaixo assina.

DSc. Leandro Melo de Sales - Orientador
Universidade Federal de Alagoas

DSc. Eduardo Setton Sampaio da Silveira - Examinador
Professor Titular
Universidade Federal de Alagoas

Fellipe Chargel - Examinador
Secretário Parlamentar
Senado Federal do Brasil

Maceió, Outubro de 2022

Catalogação na fonte
Universidade Federal de Alagoas
Biblioteca Central
Divisão de Tratamento Técnico
Bibliotecária: Taciana Sousa dos Santos – CRB-4 – 2062

S163a Sales, Lucas dos Santos.
Uma aplicação descentralizada (dApp / Web 3.0) para registros de autorias em blockchain / Lucas dos Santos Sales. – 2022.
79 f. : il. color.

Orientador: Leandro Melo de Sales.
Monografia (Trabalho de Conclusão de Curso em Ciência da Computação) – Universidade Federal de Alagoas. Instituto de Computação. Maceió, 2022.

Bibliografia: f. 75-79.

1. Direito autoral. 2. Tecnologia *blockchain*. 3. *Non-Fungible Token.* I. Título.

CDU: 004

Agradecimentos

Agradeço aos meus pais, Nivaldo e Sandra, por terem me apoiado durante todo o percurso, desde a escolha do curso até aqui, sempre dando suporte emocional e aguardando para presenciar o final de uma etapa da minha vida.

Agradeço a minha irmã, Nilva, e a sua amiga, Patricia, que me deram o apoio moral e o empurrão necessário para eu conseguir ir atrás de um tema de projeto e finalmente iniciar a fazer este trabalho.

Agradeço ao meu orientador, Leandro, que aceitou me orientar e sempre foi compreensivo com minhas dificuldades de encontrar um foco para o trabalho e me auxiliou com suas sugestões. Também agradeço a Leila por confiar em mim para "tornar realidade" as ideias propostas em sua tese.

Por fim, agradeço a todos os meus amigos, desde os que conheci na época do colégio e continuam comigo, aos que conheci na época da faculdade, tanto dentro do campus quanto fora dele, que conseguem sempre me manter com os pés no chão, mesmo quando as coisas parecem muito complicadas, conto com eles para esquecer os problemas e valorizar os pequenos momentos de alegria da vida.

Lucas Sales

"Depois da longa caminhada subimos [...], passando por uma pequena abertura na pedra, para enfim, rever as estrelas."

– Alighieri, *Dante*

Resumo

Requisitar o registro de uma obra é um processo que requer paciência. O processo, atualmente, requer que haja muita documentação, exemplares físicos do trabalho e deve ser enviado para um órgão específico que irá, manualmente, realizar todo o processo de verificação e validação para oficializar o registro do autor.

O uso de tecnologias *blockchain* para registros é uma questão que vem sendo bastante discutida. As suas características de imutabilidade e transparência permitem uma maior confiabilidade na hora de submeter um registro na rede. Com o avanço de suas tecnologias, um novo modelo de negócios baseado em uma ferramenta emergente chamada NFT (*Non-Fungible Token*) tem ganhado bastante notoriedade por permitir que artistas pudessem ter mais controle e retorno financeiro com suas artes na Internet.

Com isso em mente, este trabalho vem com o intuito de demonstrar um exemplo de aplicação web que consegue replicar o processo de registro de forma totalmente digital gerando um NFT na rede *Ethereum* que representa o registro de autoria de uma obra digitalizada no formato PDF de forma mais rápida que a usual.

Palavras-chave: Direito Autoral, Blockchain, Non-Fungible Token

Abstract

Requesting or registering a work is a process that requires patience. The process, nowadays, requires lots of documentation, physical specimens of the work and it must be sent to an specific agency that will manually carry out the entire process of verification and validation to make the author's registration official.

The use of Blockchain technologies for records is an issue that has been widely discussed. Its immutability and transparency characteristics allow greater reliability when submitting a record on the network. As their technologies advance, a new bussiness model based in an emerging tool called NFT (Non-Fungible Token) has gained considerable notoriety for allowing artists to have more control and financial feedback with their arts on internet.

With this in mind, this work comes with the objective of demonstrate an example of web application that can replicate the register process in digital way minting an NFT on Ethereum network that representates the authory register of a digital work in PDF format faster than usual.

Key-words: Copyright, Blockchain, Non-Fungible Token

Lista de Figuras

Lista de Algoritmos

Conteúdo

1

Introdução

Diariamente, artistas ao redor do mundo estão se esforçando para dar vida às suas ideias. Muitos deles tiram seu sustento a partir daquilo que produzem, seja em forma de desenho, música ou livro. Por isso é necessário que eles tenham segurança na hora de colocarem seus produtos no mercado, pois somente aquele que detiver os direitos de autoria da obra pode se beneficiar dela. Para isso, existem leis, como a L9.279 e a L9.610, que regularizam os registros de autoria e garantem proteção legal para os criadores.

Cada dia que passa, as tecnologias vão avançando cada vez mais. Da mesma forma, tem ficado cada vez mais perigoso para os artistas manterem os direitos de suas obras. Graças a Internet, eles estão suscetíveis a qualquer mal-intencionado que queira se passar pelo autor para lucrar em cima do trabalho de outra pessoa, sendo que basta uma simples de prova de anterioridade que mostre que, historicamente, você foi o primeiro a conceber tal obra que você poderá reclamar os diretos de autor sobre ela.

Com isso em mente, esse trabalho vem com a proposta de oferecer uma plataforma que seja "amigável" para os usuários que queiram realizar um registro de autoria de suas obras utilizando tecnologias *Blockchain* para gerar um registro na forma de um ativo digital conhecido como NFT. Assim, utilizando a rapidez da tecnologia moderna e as propriedades de imutabilidade e marcação temporal associadas da *Blockchain* para facilitar esse processo de registro.

1.1 Problemática

Apesar de existirem métodos para o reconhecimento de autoria, o processo para registrar é complicado e, muitas vezes, demorado. Segundo o site da própria Biblioteca Nacional, atualmente o tempo médio para análise de um requerimento é de 180 dias, graças ao período da pandemia, o serviço operou com equipe reduzida. Os serviços físicos estão suscetíveis a imprevistos que possam afetar as pessoas responsáveis por ele.

Além da demora e imprevistos que podem dificultar o processo, também existe a questão do alcance que a validade de um registro possui. Normalmente ele é apenas válido em território nacional e não há convenções e acordos como as propriedades industriais que expandem os territórios onde certos registros são válidos.

1.2 Objetivos

O objetivo geral deste trabalho é a criação de uma plataforma web para realizar o registro de autoria de propriedades intelectuais em *Blockchain*.

1.2.1 Objetivos Específicos

Este trabalho tem como objetivos específicos o desenvolvimento de:

- Um contrato inteligente para geração de *Tokens* Não-Fungíveis.
- Uma API (*Application Programming Interface*) capaz de lidar com os dados dos usuários.
- Uma aplicação web capaz de registrar documentos digitais de forma simples e rápida numa *Blockchain*;

1.3 Relevância

O ramo do Direito Autoral realiza um trabalho importante na sociedade, garantindo a proteção dos artistas que vivem de suas obras, permitindo que possam tirar seu sustento a partir do que produzem. Leis como a de Direitos Autorais (Lei n.º 9610) e órgãos como a Biblioteca Nacional são os principais incentivadores dos artistas no território nacional.

É importante que haja tais incentivos para que mais pessoas possam se sentir a vontade para criar mais obras de cunho artístico para enriquecerem ainda mais a cultura de onde vem e espalhem mensagens que podem ser transformadoras na vida de outras pessoas.

A tecnologia é um elemento transformador na vida de todos os seres humanos, sempre avançando a passos largos e permitindo que diversos processos possam ser acelerados e até organizados de forma mais eficiente. As tecnologias *blockchain* chegaram prometendo serem um transformador de paradigmas de como usamos a Web hoje em dia, especialmente na área de registros.

O ramo de registro de propriedades intelectuais tem olhado bastante para estas tecnologias e tentado desenvolver formas de utilizar as vantagens da *blockchain* em prol de beneficiar todos os que desejarem obter um registro intelectual, seja industrial ou autoral, de forma segura e aberta.

Esse trabalho visa ressaltar a importância de se ter um sistema que consiga tornar esses processos de registros mais rápidos e acessíveis aos artistas, mas sem a intenção de ser um "substituto" aos meios tradicionais já existentes e ainda tentando trazer uma luz para formas de usos das tecnologias blockchain, especialmente NFTs, que costumam ser mal interpretadas pelos usuários comuns que veem como enganações ou artigos de ostentação sem propósito na Internet.

1.4 Trabalhos Relacionados

Como comentado no tópico anterior, a área de registros de propriedades tem visado muito o usos de tecnologias para tentar melhorar o processo de registro de Direito Autoral, seja por meios tradicionais ou pelo uso de tecnologias *blockchain* através de plataformas web, .

Existem plataformas que focam nesse registro, mas que não se utilizam de tecnologias descentralizadas, como a Registro de Obras, que foca em armazenar em servidores centralizados os arquivos registrados.

Das plataformas que se utilizam de *blockchains* para manter seus registros, temos a Avctoris e a Câmara Brasileira do Livro (CBL), ambos se utilizam de *blockchain* para gerar um certificado utilizado como forma de validar o registro feito. Diferente deste trabalho, elas não se utilizam de *Tokens* Não-Fungíveis como principal forma de manter o registro na rede.

A *startup* InspireIP é um exemplo de aplicação semelhante à proposta que traz a ideia de realizar os registros na rede descentralizada utilizando NFTs e trazendo um *marketplace* interno que permite colocar as artes registradas a disposição de quem desejar comprá-las.

1.5 Estrutura

No segundo capítulo serão abordados os conceitos teóricos por trás do desenvolvimento desse projeto, sendo eles o conceito de *Blockchain* e suas ferramentas associadas, os algoritmos de consenso, a ideia de *tokens* e NFTs, contratos inteligentes, o IPFS e sua proposta de armazenamento descentralizado e as definições de Propriedade Intelectual e suas ramificações. No terceiro capítulo será abordado o desenvolvimento da aplicação proposta discutindo sobre suas tecnologias e integrações. No quarto capítulo serão apresentados os resultados obtidos a partir da experimentação. No quinto capítulo serão abordadas as considerações finais do projeto bem como trabalhos futuros associados para um melhor desenvolvimento do mesmo.

2

Fundamentação teórica

Neste capítulo serão abordados os principais temas que envolvem a solução do trabalho. Entre eles os conceitos de Propriedade Intelectual, Direito Autoral, Registro de Anterioridade, *Blockchain*, *Smart Contracts*, NFT e IPFS.

2.1 Propriedade Intelectual

A Propriedade Intelectual é uma área do Direito que visa garantir que inventores e criadores possam ter proteção legal e o reconhecimento da propriedade de suas obras. Tal direito garante a proteção da inovação e incentiva aos criadores a continuarem não só criando, mas também tirando seu sustento disso. Segundo a OMPI (Organização Mundial da Propriedade Intelectual), o conceito de Propriedade Intelectual se divide em duas vertentes: a Propriedade industrial e o Direito Autoral [da Indústria].

2.1.1 Propriedade Industrial

Englobam as propriedades cujo foco está voltado para atividade empresarial como marcas, patentes, desenhos industriais e indicações geográficas. Sua importância se dá por garantir proteção das invenções contra cópias de empresas concorrentes, incentivando ainda mais a atividade industrial e movendo a economia global.

Esse ramo da Propriedade Intelectual é regulado pela Lei número 9.279, conhecida como Lei da Propriedade Industrial (LPI), de 14 de maio de 1996. Esta lei regula todos os direitos e deveres relacionados a propriedade industrial.

Além disso, para facilitar o processo de registro dessas propriedades, o Governo Federal criou, em 11 de dezembro de 1970, o Instituto Nacional de Propriedade Industrial (INPI). Sua função é auxiliar os empreendedores no registro de Propriedades Industriais para que tudo esteja devidamente regulado nas normas estabelecidas pelo território nacional [da Indústria].

2.1.2 Direito Autoral

Englobam as obras cujo caráter é intelectual, artístico ou literário. Diferente da propriedade industrial, que possui elementos mais bem definidos, as obras inclusas no Direito Autoral são mais abstratas, devido ao caráter "artístico", abrangendo músicas, pinturas ou livros.

Esse ramo é regulado pela Lei número 9.610, conhecida como a Lei de Direitos Autorais, de 19 de fevereiro de 1998. Seu órgão regulador, a Biblioteca Nacional, que será melhor abordado na Seção 2.1.3 [Planalto; Sebrae].

2.1.3 Biblioteca Nacional

Fundada em 1810 e localizada no Rio de Janeiro, a Fundação Biblioteca Nacional visa garantir a captação, guarda, preservação e difusão de documentos históricos e da produção intelectual do país em um acervo que se aproxima de 9 milhões de itens. Considerado pela UNESCO como a maior biblioteca nacional da América latina e umas das principais do mundo. A Biblioteca Nacional conta também com um avançado laboratório para restauração e conservação de papéis, uma oficina de encadernação e um centro de microfilmagens, fotografia e digitalização.

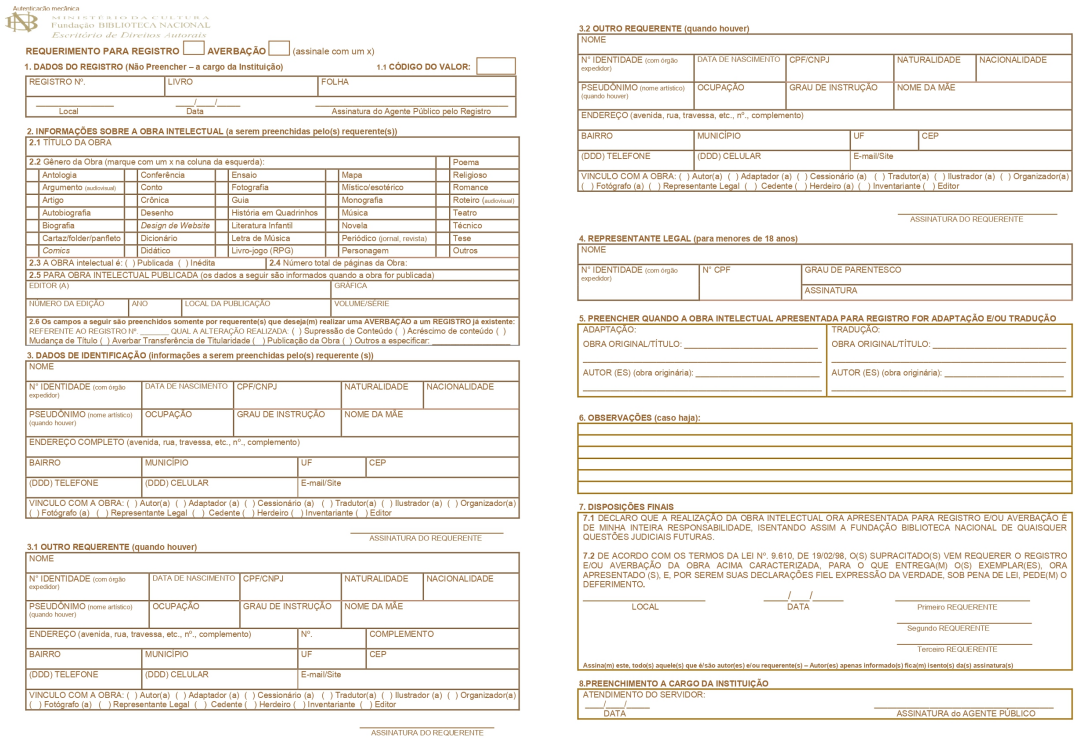
A Biblioteca Nacional começou a ser responsável pelos registros de obras intelectuais em 1898, quando surgiu no Brasil a primeira lei que abordava essa questão, a Lei Medeiros e Albuquerque. Desde então, a fundação é a principal responsável pela manutenção dos registros de anterioridade e autoria das produções intelectuais, artísticas e culturais do país [Nacional (a); do deputados].

2.1.4 Processo para registro na Biblioteca Nacional

A biblioteca exige que você possua uma cópia física da sua obra e que ela seja enviada ou em forma de folhas avulsas, sem verso e presas pro clipe, ou no formato de livro publicado. Para o caso de folhas avulsas, é recomendado que ou todas as folhas possuam rubricas, ou possuir uma folha de rosto contendo a rubrica e a informação do número de páginas presentes no documento. Além disso, também é necessário reunir quaisquer documentos associados a obra, como contratos de cessão, procurações e a cópias de identidade de terceiros associados a eles pela assinatura.

Também é necessário gerar um boleto na Guia de Recolhimento da União utilizando os dados da Tabela de retribuição e ser pago no Banco do Brasil ou qualquer instituição que seja conveniada. O comprovante deve constar junto aos documentos a serem enviados [Nacional (b)].

Outro procedimento requerido é um formulário de requerimento de registro, que pode ser baixado no site deles, impresso e preenchido com os dados necessários sobre a obra a ser registrada e seus requerentes. O primeiro e último itens devem permanecer em branco, pois devem



Autenticação mecânica

MINISTÉRIO DA CULTURA
Fundação BIBLIOTECA NACIONAL
Escritório de Direitos Autorais

REQUERIMENTO PARA REGISTRO ☐ AVERBAÇÃO ☐ (assinale com um x)

1. DADOS DO REGISTRO (Não Preencher – a cargo da Instituição) 1.1 CÓDIGO DO VALOR:

REGISTRO Nº.	LIVRO	FOLHA
Local	Data	Assinatura do Agente Público pelo Registro

2. INFORMAÇÕES SOBRE A OBRA INTELECTUAL (a serem preenchidas pelo(s) requerente(s))

2.1 TÍTULO DA OBRA

2.2 Gênero da Obra (marque com um x na coluna da esquerda):

	Antologia		Conferência		Ensaio		Mapa		Poema
	Argumento (audiovisual)		Conto		Fotografia		Místico/esotérico		Religioso
	Artigo		Crônica		Guia		Monografia		Romance
	Autobiografia		Desenho		História em Quadrinhos		Música		Roteiro (audiovisual)
	Biografia		*Design de Website*		Literatura Infantil		Novela		Teatro
	Cartaz/folder/panfleto		Dicionário		Letra de Música		Periódico (jornal, revista)		Técnico
	Comics		Didático		Livro-jogo (RPG)		Personagem		Tese
									Outros

2.3 A OBRA intelectual é: ( ) Publicada ( ) Inédita 2.4 Número total de páginas da Obra:

2.5 PARA OBRA INTELECTUAL PUBLICADA (os dados a seguir são informados quando a obra for publicada)

EDITOR (A)		GRÁFICA
NÚMERO DA EDIÇÃO	ANO	LOCAL DA PUBLICAÇÃO	VOLUME/SÉRIE

2.6 Os campos a seguir são preenchidos somente por requerente(s) que deseja(m) realizar uma AVERBAÇÃO a um REGISTRO já existente:
REFERENTE AO REGISTRO Nº. _____ QUAL A ALTERAÇÃO REALIZADA: ( ) Supressão de Conteúdo ( ) Acréscimo de conteúdo ( ) Mudança de Título ( ) Averbar Transferência de Titularidade ( ) Publicação da Obra ( ) Outros a especificar: _____

3. DADOS DE IDENTIFICAÇÃO (informações a serem preenchidas pelo(s) requerente (s))

NOME				
Nº IDENTIDADE (com órgão expedidor)	DATA DE NASCIMENTO	CPF/CNPJ	NATURALIDADE	NACIONALIDADE
PSEUDÔNIMO (nome artístico) (quando houver)	OCUPAÇÃO	GRAU DE INSTRUÇÃO	NOME DA MÃE	
ENDEREÇO COMPLETO (avenida, rua, travessa, etc., nº., complemento)				
BAIRRO	MUNICÍPIO	UF	CEP	
(DDD) TELEFONE	(DDD) CELULAR	E-mail/Site		

VÍNCULO COM A OBRA: ( ) Autor(a) ( ) Adaptador (a) ( ) Cessionário (a) ( ) Tradutor(a) ( ) Ilustrador (a) ( ) Organizador(a) ( ) Fotógrafo (a) ( ) Representante Legal ( ) Cedente ( ) Herdeiro ( ) Inventariante ( ) Editor

ASSINATURA DO REQUERENTE

3.1 OUTRO REQUERENTE (quando houver)

NOME				
Nº IDENTIDADE (com órgão expedidor)	DATA DE NASCIMENTO	CPF/CNPJ	NATURALIDADE	NACIONALIDADE
PSEUDÔNIMO (nome artístico) (quando houver)	OCUPAÇÃO	GRAU DE INSTRUÇÃO	NOME DA MÃE	
ENDEREÇO (avenida, rua, travessa, etc., nº., complemento)		Nº.	COMPLEMENTO	
BAIRRO	MUNICÍPIO	UF	CEP	
(DDD) TELEFONE	(DDD) CELULAR	E-mail/Site		

VÍNCULO COM A OBRA: ( ) Autor(a) ( ) Adaptador (a) ( ) Cessionário (a) ( ) Tradutor(a) ( ) Ilustrador (a) ( ) Organizador(a) ( ) Fotógrafo (a) ( ) Representante Legal ( ) Cedente ( ) Herdeiro ( ) Inventariante ( ) Editor

ASSINATURA DO REQUERENTE

3.2 OUTRO REQUERENTE (quando houver)

NOME				
Nº IDENTIDADE (com órgão expedidor)	DATA DE NASCIMENTO	CPF/CNPJ	NATURALIDADE	NACIONALIDADE
PSEUDÔNIMO (nome artístico) (quando houver)	OCUPAÇÃO	GRAU DE INSTRUÇÃO	NOME DA MÃE	
ENDEREÇO (avenida, rua, travessa, etc., nº., complemento)				
BAIRRO	MUNICÍPIO	UF	CEP	
(DDD) TELEFONE	(DDD) CELULAR	E-mail/Site		

VÍNCULO COM A OBRA: ( ) Autor(a) ( ) Adaptador (a) ( ) Cessionário (a) ( ) Tradutor(a) ( ) Ilustrador (a) ( ) Organizador(a) ( ) Fotógrafo (a) ( ) Representante Legal ( ) Cedente ( ) Herdeiro ( ) Inventariante ( ) Editor

ASSINATURA DO REQUERENTE

4. REPRESENTANTE LEGAL (para menores de 18 anos)

NOME		
Nº IDENTIDADE (com órgão expedidor)	Nº CPF	GRAU DE PARENTESCO
		ASSINATURA

5. PREENCHER QUANDO A OBRA INTELECTUAL APRESENTADA PARA REGISTRO FOR ADAPTAÇÃO E/OU TRADUÇÃO

ADAPTAÇÃO:	TRADUÇÃO:
OBRA ORIGINAL/TÍTULO: _____	OBRA ORIGINAL/TÍTULO: _____
AUTOR (ES) (obra originária): _____	AUTOR (ES) (obra originária): _____

6. OBSERVAÇÕES (caso haja):

7. DISPOSIÇÕES FINAIS

7.1 DECLARO QUE A REALIZAÇÃO DA OBRA INTELECTUAL ORA APRESENTADA PARA REGISTRO E/OU AVERBAÇÃO É DE MINHA INTEIRA RESPONSABILIDADE, ISENTANDO ASSIM A FUNDAÇÃO BIBLIOTECA NACIONAL DE QUAISQUER QUESTÕES JUDICIAIS FUTURAS.

7.2 DE ACORDO COM OS TERMOS DA LEI Nº. 9.610, DE 19/02/98, O(S) SUPRACITADO(S) VEM REQUERER O REGISTRO E/OU AVERBAÇÃO DA OBRA ACIMA CARACTERIZADA, PARA O QUE ENTREGA(M) O(S) EXEMPLAR(ES), ORA APRESENTADO (S), E, POR SEREM SUAS DECLARAÇÕES FIEL EXPRESSÃO DA VERDADE, SOB PENA DE LEI, PEDE(M) O DEFERIMENTO.

LOCAL _____ DATA _____

Primeiro REQUERENTE _____
Segundo REQUERENTE _____
Terceiro REQUERENTE _____

Assina(m) este, todo(s) aquele(s) que é/são autor(es) e/ou requerente(s) – Autor(es) apenas informado(s) fica(m) isento(s) da(s) assinatura(s)

8.PREENCHIMENTO A CARGO DA INSTITUIÇÃO

ATENDIMENTO DO SERVIDOR:
DATA _____ ASSINATURA do AGENTE PÚBLICO _____

Figura 2.1: Formulário para registro na Biblioteca Nacional.

ser preenchidos pela equipe da Biblioteca. Conforme ilustrado na Figura 2.1, as seções presentes são:

- Informações sobre a obra (título, gênero, número de páginas, etc.);
- Informações pessoais sobre os requentes (contendo até 3 espaços);
- Informações sobre representantes legais (em caso de menores de idade);
- Informações sobre a obra original (em caso de traduções ou adaptações);
- Observações;
- Disposições finais.

Com todos os documentos recolhidos, a obra separada, GRU paga e formulário preenchido, é a hora de enviar para a Biblioteca Nacional. É só se dirigir presencialmente a um Escritório de Direitos Autorais portando um documento com foto e protocolar o requerimento para seu registro. Após ser analisado, um comprovante será emitido contendo um número que serve de identificador para o acompanhamento do processo. Caso não seja possível ir pessoalmente, deve pode ser autorizado um responsável que vá.

Após tudo isso é só aguardar. Graças a pandemia os prazos médios estipulados estão em torno de 180 dias. Quando os procedimentos forem analisados, eles entrarão em contato através de uma carta informando o resultado do procedimento [Nacional (c)].

2.2 Web3

No começo da Internet, havia apenas páginas e documentos estáticos que as pessoas acessavam nas máquinas de outras pessoas, chamada Web1. Nessa época, apenas documentos eram compartilhados e tudo era "apenas leitura", com praticamente nenhuma, interação do usuário com as páginas.

Com o tempo, a web evoluiu para a chamada Web2, onde o usuário começou a ter mais poder nos sites. Começaram a surgir redes sociais, contas e outros tipos de serviços que conseguem personalizar a sua experiência com base em como você a usa. Por exemplo, o uso de *cookies*, que permitem que se você pesquisar por um produto, outro site que tiver acesso a eles vai poder mostrar propagandas relacionadas ao que você procurou anteriormente. As redes sociais e serviços de *streaming* que usam algoritmos para te recomendar páginas ou vídeos relacionados com aquilo que você consome ou a outros usuários que consomem o mesmo que você.

Com isso a Internet se tornou um local mais centralizado e com menos privacidade. Uma rede social como Facebook ou Twitter possuem servidores centrais, por onde tudo deve passar. Isso garante mais controle sobre seus dados e permitindo que elas possam, mesmo sem você saber, coletar dados pessoais e vendê-los para empresas.

Com isso surgiu o conceito da chamada Web3, onde ao invés de termos servidores que centralizam o poder, todo mundo é seu próprio servidor. A Web3 se baseia bastante na ideia de ser totalmente descentralizada e de todo mundo, sem ter uma grande empresa por trás controlando e ditando o que pode ser feito, especialmente sem a influência dos governos. Se o governo de um país decide bloquear um site, eles só precisam que um servidor pare de rodar e o sistema não funcionará mais. Já na Web3, um site descentralizado continuará existindo enquanto alguém na rede possuir uma cópia. A Figura 2.2 ilustra um esquema comparativo entre a Web2 e a Web3.

Com esse paradigma em mente, começaram a surgir os chamados *dApps* (*Decentralized Application*) sendo aplicações que não utilizam servidores já que rodam de maneira distribuída. Também nessa linha teve o surgimento das *deFi* (*Decentralized Finances*) sendo métodos de finanças que não necessitam de intermédios de bancos ou qualquer outra instituição para controlá-los. Assim chegamos a uma das principais tecnologias quando se fala de redes descentralizadas, as *Blockchains*, cujos conceitos associados serão abordados na Seção 2.4 [Marlinspike; Laranjeira (b); is going great; Crypto (f); Coinbase; Freitas; Cabral].

2.3 Hash

É um conceito de codificação de dados que se refere a uma função matemática que recebe uma mensagem **m** de tamanho variável como sendo sua entrada e gera como saída um texto de tamanho fixado único para essa entrada. Ou seja, para diferentes valores de entrada atribuídos para a função **H** ela deve sempre gerar uma saída diferente, e uma alteração por mais que miníma em uma certa entrada **m**, gerando uma entrada **m'**, deve sempre causar drásticas alterações no

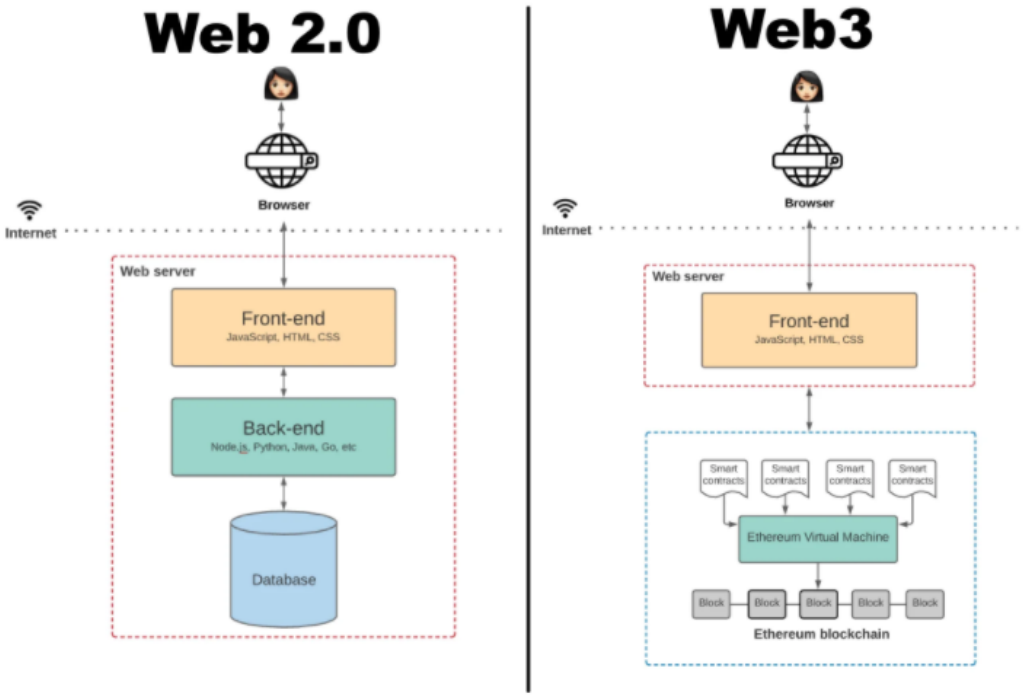


Figura 2.2: Esquema de Web2 vs Web3 [Ibosiola].

valor da *hash* de saída.

Uma das vantagens da conversão em *hash* é que, dada uma *hash* **H(m)**, deve ser extremamente difícil, beirando o impossível, de se obter a entrada **m** original a partir apenas do valor de *hash*, dificultando assim, em caso de vazamento de dados, que dados sigilosos como senhas sejam recuperadas, tornando essa função uma conversão de "mão-única". A Figura 2.3 representa o esquema de aplicação de uma função *hash* [Crypto (b)].

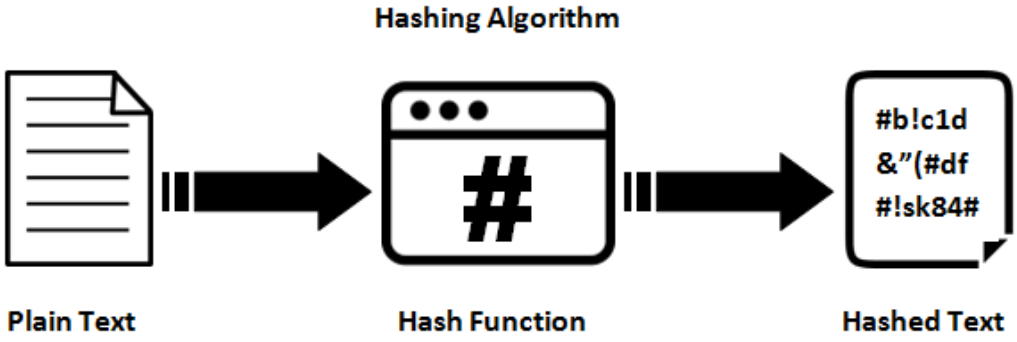


Figura 2.3: Esquema de uma função *Hash* [Anand].

2.3.1 Exemplos de aplicação

Como já citado anteriormente, esse tipo de codificação é muito usado para armazenamento de senhas, pois sempre que você digitar a senha para acessar uma conta, o valor digitado é convertido em uma *hash* e comparado com o valor previamente salvo durante um cadastro, como pela regra apenas duas *hashes* serão iguais se e somente se as entradas forem a mesmas, o acesso só será permitido a quem souber qual era mensagem original e caso uma falha de segurança leve a terceiros a acessarem os dados salvos em banco, eles não conseguiram descobrir as senhas a partir dos valores apenas das *hashes*.

Outra aplicação muito comum é em redes *blockchain*, a qual será abordada em mais detalhes na Seção 2.4 deste trabalho, onde os dados de transação são sempre convertidos em *hash* para facilitar o armazenamento, além de também serem relacionados com os conceitos de "cadeia"ligando os blocos e pela formação de um estrutura de dados conhecida como *Merkle Tree*.

2.4 Blockchain

Blockchains são um modelo de armazenamento de dados caracterizado por utilizar uma “Cadeia de Blocos” como estrutura principal de sua rede. Elas atuam como um “livro-razão” que registra dados referentes a transações feitas utilizando a moeda principal que roda dentro da rede, mantendo assim o controle das quantidades de moedas que cada usuário possui.

Cada bloco presente na rede está interligado através de um valor codificado por uma função *hash* (Seção 2.3). Os blocos possuem eu seu interior um grupo de transações autorizadas pela rede, uma marcação temporal indicando quando o bloco foi gerado e o *hash* gerado pelo bloco anterior a ele, com exceção do bloco inicial da cadeia denominado de "Bloco Gênesis".

Dessa forma, cada bloco presente na rede está ligado a seu antecessor pelo valor de sua *hash* que, por ser único, impede a fraude de um bloco no meio da rede. Qualquer miníma alteração no seu conteúdo alterar drasticamente o valor de sua *hash*, levando ao valor da *hash* de seus sucessores irem causando uma reação em cadeia de conflitos que leva a rede a descartar o bloco adulterado. A Figura 2.4 exemplifica o modelo de cadeia em que os blocos se apresentam na rede.

As redes *blockchain* operam de forma descentralizada, assim cada aparelho conectado na rede, denominados de “nós” possui uma cópia própria dos dados da rede, não existe uma autoridade maior que consiga controlar a rede e todas as novas entradas são validadas pelos algoritmos de consenso (Seção 2.4.1) que rodam na rede em questão.

Quando um bloco novo é gerado e adicionado a rede, processo denominado “mineração de blocos”, normalmente, para incentivar que os nós participantes gastem seus recursos computacionais para gerar um bloco e um valor de recompensa é oferecido para aqueles que conseguirem resolver o quebra cabeça necessário para gerar um bloco válido.

Na rede *Bitcoin*, por exemplo, as “*hashes*” dos blocos devem seguir um padrão. Normal-

mente todas devem possuir um certo número de zeros no final da cadeia de caracteres, e dentro de cada bloco existe um valor denominado *nonce* que é um número desconhecido que todos os nós mineradores devem tentar adivinhar para que a *hash* atenda as condições necessárias e seja minerado para rede. Esse método é denominado *Proof of Work* (Seção 2.4.1). Quando um nó vencedor dessa "corrida", ele recebe um valor pago com a moeda corrente da rede cujo bloco foi minerado em sua carteira como agradecimento pelos recursos gastos nessa missão [Nakamoto (2008); Hayes; BBVA; Moreland (a); Euromoney; Lu et al. (2019); Gensler (a,b,c)].

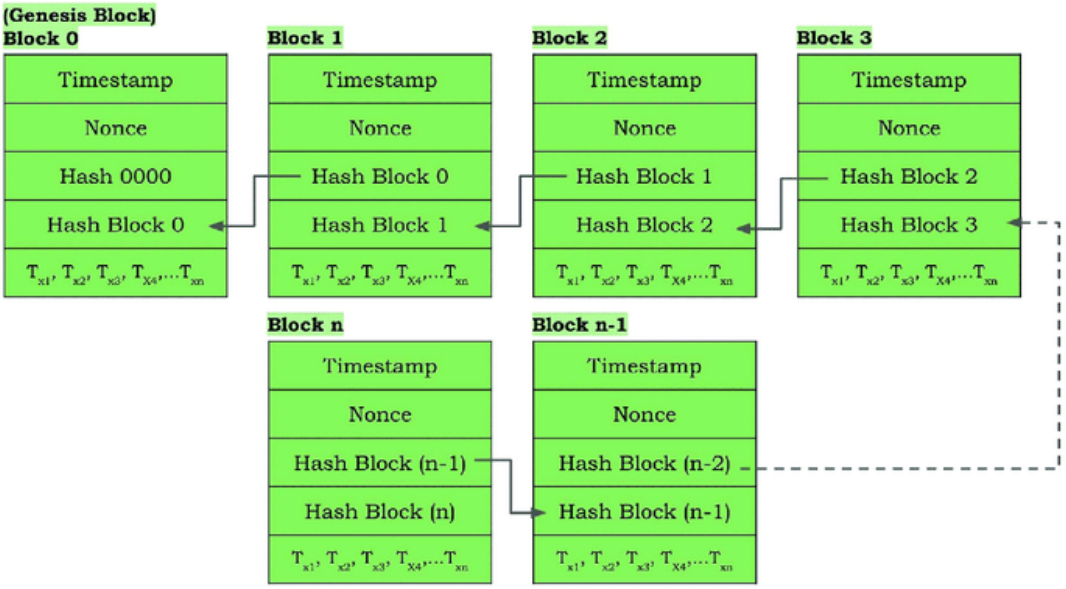


Figura 2.4: Esquema de uma *Blockchain* [Rahardja et al.].

2.4.1 Consenso

Os algoritmos de consenso em uma *blockchain* servem para haver uma "harmonia" na cadeia em todos os seus nós. Como a rede opera de modo descentralizado, todas as partes podem adicionar seus próprios nós na cadeia. Para gerar um acordo entre todos os nós e validar um bloco novo para ser adicionado e distribuído para todos na rede, os algoritmos de consenso entram em ação. Alguns exemplos de algoritmos são:

Proof of Work

Um dos mais utilizados, requer que as máquinas tentem solucionar um problema matemático, que normalmente onde a primeira a concluir consegue o direito de minerar o bloco e com isso receber o "agrado" que a rede proporciona ao vencedor dessa competição. Na rede *Bitcoin* consiste em variar um valor de *nonce* ate a *hash* gerada atingir um determinado número de zeros no seu fim. Ele carrega diversas críticas e polêmicas devido ao uso de força bruta das máquinas que leva a um consumo excessivo de energia, especialmente por parte das chamadas "fazendas

de mineração" [Nakamoto (2008); Geeksforgeeks; Laranjeira (a)]. A Figura 2.5 retrata um esquema de *Proof of Work*.

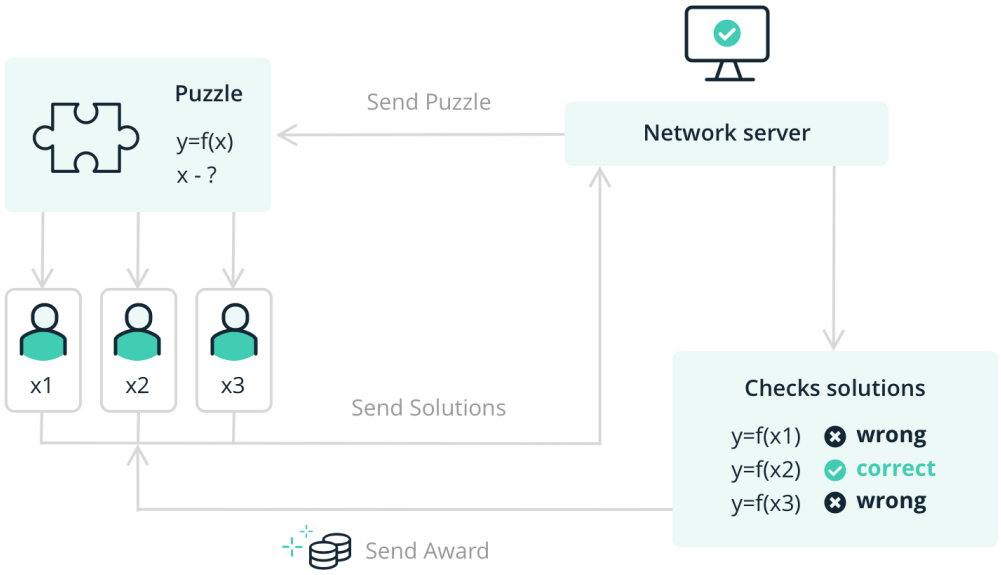


Figura 2.5: *Proof of Work* [Moreland (b)].

Proof of Stake

A alternativa mais comum ao PoW, consistem em, ao invés de resolver problemas matemáticos em uma corrida computacional, alguns nós da rede que possuem uma certa quantia miníma da moeda da rede são selecionados aleatoriamente e eles se tornam os validadores que vão juntos montar o bloco que irá para a rede. Os nós selecionados tem uma quantidade de moedas bloqueadas para serem usadas no processo e elas serão perdidas em caso de fraudes. Após o bloco ser gerado, os validadores recebem seus pagamentos [Explained (a); Staff; Crypto (a)]. A Figura 2.6 retrata um esquema de *Proof of Stake*.

Delegated Proof of Stake

Considerado uma versão mais "eficiente" do PoS, atua de forma semelhante, só que os nós escolhidos para serem validadores se tornam eleitores num sistema de votação para eleger nós delegados que terceirizam as tarefas sendo responsáveis pela obtenção do consenso na rede. Quando o bloco é gerado, o valor recebido pelo nó delegado é partilhado entre os nós que o elegeram [Academy; Staff].

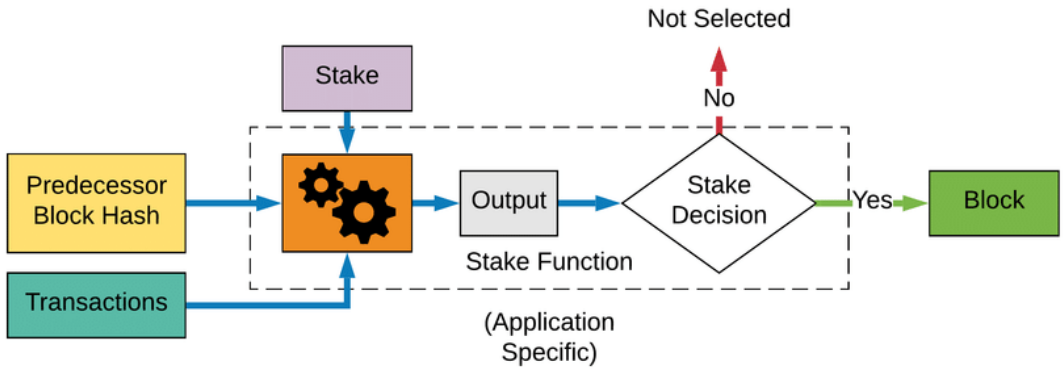


Figura 2.6: *Proof of Stake* [Gramoli].

Proof of Elapsed Time

É um algoritmo desenvolvido pela Intel que visa dar chances iguais a todos os nós de poderem gerar o próximo bloco da rede. Ele funciona dando a todos os nós um valor de espera onde cada nó irá "dormir" durante esse tempo. Os valores são aleatórios, então o nó que tiver a sorte de sair com o menor tempo de espera "acorda" primeiro e leva o bloco como recompensa [Centieiro; Frankenfield (a)].

2.4.2 Smart Contracts

Um *Smart Contract* ou "Contrato Inteligente" consiste em um arquivo de código executado dentro de uma rede *blockchain*, desde que ela tenha suporte. Redes como *Bitcoin* apenas conseguem armazenar os dados de transação, já redes como a *Ethereum*, possuem suporte a *smart contracts*.

Uma vez na rede, eles não podem ser alterados, portanto, é necessário garantir que o contrato está funcionando devidamente antes de ser enviado para rede. Caso seja necessário ajustar o contrato, é necessário realizar o *deploy* de um novo contrato e deixando o anterior de lado [IBM; Okabayashi; Crypto (c)].

2.4.3 ABI

A *Application Binary Interface* é um objeto gerado após a compilação do contrato que age como um facilitador entre os métodos do contrato e a aplicação. Por ser uma Interface, ela indica o formato de acesso para os métodos e atributos disponíveis no contrato, bem como seus parâmetros e retornos. Dessa forma, permitindo que a plataforma possa se comunicar com o contrato sem problemas de tipagem ou nomenclaturas.

2.4.4 Token

Tokens são cripto-ativos que operam em uma rede *blockchain* e se utilizando de sua infraestrutura, eliminando a necessidade de criar uma rede para o *token* operar. Assim facilitam o desenvolvimento de cripto-ativos que podem ser utilizados, compartilhados ou até negociados dentro da *blockchain* em que ele foi gerado. É comum que projetos que se iniciam como *tokens* evoluam a ponto de criarem sua própria *blockchain*, mudando o *token* para uma Cripto-moeda.

Os *Tokens* podem possuir funções mais amplas do que apenas transações. Por exemplo: *tokens* de governança que dão mais autoridade para aqueles que o possuírem e quanto mais possuir, maior o poder de decisão, *tokens* para representar outros bens para não ser necessário adquiri-los, similar a uma compra de ações, *tokens* que facilitam transações com dinheiro real, etc [Kovacs; Clarke-Potter; Frankenfield (b); Crypto (d)].

2.4.5 NFT

NFT é a sigla para *Non-Fungible Token* (*Token* Não Fungível) e se refere a um tipo específico de *token* que é "único" na rede. Enquanto, por exemplo, um bem "fungível" como uma nota de 10 reais pode ser trocado por outra nota de 10 reais sem haver alguma "perda de valor", pois ambas são, em valor, idênticas, um bem não fungível não possui essa característica. Sua unicidade na rede o torna raro e valioso aos olhos de quem coleciona ou compra tais ativos na rede.

O termo NFT tem ganhado muito reconhecimento ultimamente e se tornado uma tendência apesar de sofrer muito preconceito devido a usos indevidos ou até mesmo "questionáveis"para os quais ele está sendo aplicado, o mais notório de todos é, sem dúvida, o mercado de arte digital, onde plataformas como *Opensea* permitem que artistas digitais exponham suas artes e a coloquem a venda utilizando blockchain como método de transação. A Figura 2.7 mostra uma listagem de NFTs disponíveis na *Opensea*.

Cada imagem está armazenada, seja em uma "nuvem" ou até mesmo utilizando um protocolo de armazenamento de arquivos descentralizado chamado IPFS (Seção 2.5), responsáveis por alocar a imagem original. O *Opensea* gerencia o processo de criação do NFT utilizando metadados associados a imagem e registrando na rede *blockchain*, processo denominado "*mint*".

Uma vez que o NFT tenha sido "mintado" na rede, ele pode ser adicionado a uma carteira como um cripto-ativo qualquer, sendo possível transferir, vender ou revender conforme ele for valorizando, como um ativo especulativo [Silveira et al.; Crypto (e); Harris; Marlinspike; Mudgil].

2.4.6 Carteiras

No contexto de *blockchain*, carteiras são as aplicações que permitem que os usuários possam realizar as transações na rede. Elas também são as responsáveis por manter seu saldo, seja em

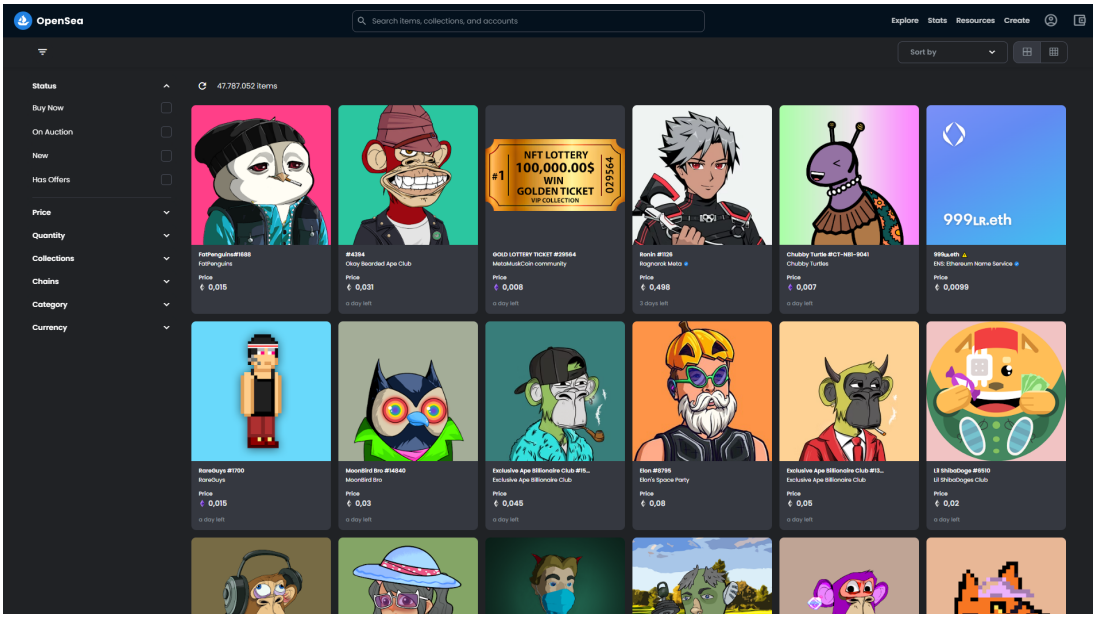


Figura 2.7: Lista de NFTs na plataforma *Opensea*

criptomoedas ou outros ativos digitais como *tokens* válidos na rede em que estiver conectado.

As carteiras podem ser virtuais, na forma de softwares que operam em desktop, navegadores ou aplicativos para celulares, como também podem se apresentar de forma física como um hardware que pode ser conectado a um computador.

Quando um usuário acessa uma carteira pela primeira vez, ele precise gerar uma conta, que funcionará com um esquema de criptografia assimétrica, possuindo duas chaves: uma pública, o endereço da carteira visível a todos para poderem utilizar como destinatário de transações, e uma secreta, que deve ser mantida em sigilo.

Além disso, muitas vão precisar de uma senha de acesso e, para o caso do acesso ser perdido ou for necessário em outro dispositivo, existe um conjunto secreto de no mínimo 12 palavras em inglês, chamada *Mnemonic Seed*, que devem ser digitadas em uma ordem específica para recuperar o acesso. É aconselhável que esse conjunto de palavras seja armazenado em um local seguro, de preferência no mundo físico.

Alguns exemplos comuns de carteiras são a *Metamask*, Figura 2.8, que será o foco nesse projeto, *Electrum*, *Trust*, *Rainbow*, *Coinbase Wallet*, *Phantom*, *Glow*, dentre muitas outras. Nem todas tem suporte a qualquer rede, algumas podem ter suporte a uma rede específica como a *Electrum* que só provê suporte a rede *Bitcoin* e possui uma versão para a rede *Litecoin*, mas muitas podem operar em diversas redes e é possível adicionar redes novas dentro delas [Frankenfield (c)].

2.4.7 Ethereum

Após a popularização das *blockchains* graças a rede *Bitcoin*, muitos serviços além de finanças, começaram a ser cogitados para operarem em redes descentralizadas. Infelizmente, a rede do

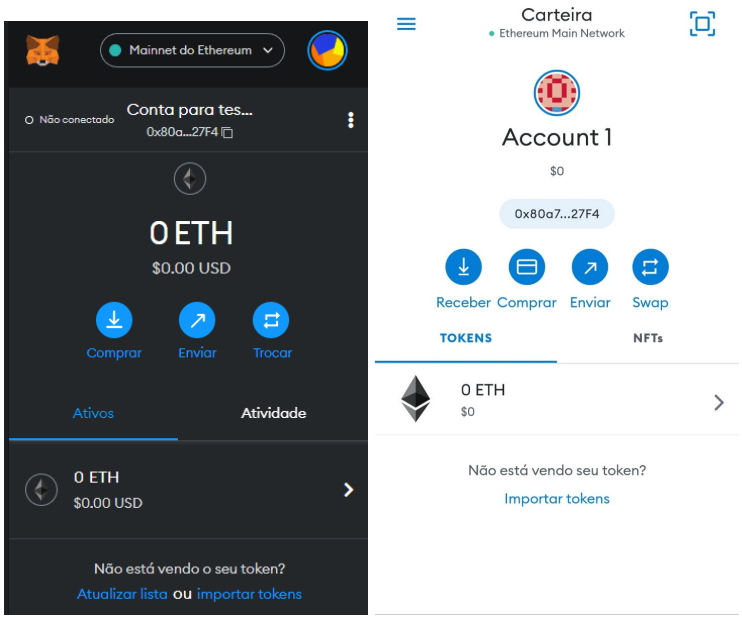


Figura 2.8: Carteira *Metamask* (Navegador a esquerda e Mobile a direita).

Bitcoin não oferecia suporte para outras operações além das transações. Dessa forma, surgiu a *Ethereum* como uma forma de facilitar que desenvolvedores que desejassem criar *dApps* sem ter que se preocupar em ter que montar a própria rede e dar suporte.

Diferente da rede *Bitcoin*, a *Ethereum* consegue rodar muito mais do que apenas as transações, permitindo o desenvolvimento de *dApps* dentro dela. A possibilidade de rodar contratos inteligentes (Seção 2.4.2) expandiu bastante as possibilidades de implementações na rede. Para criar um contrato, basta escrever um *script* utilizando a linguagem *Solidity*, criada para esse fim, e depois enviar para a rede, fazendo um *deploy*.

Além disso, a rede também trabalha com a sua própria moeda, o *Ether* (ETH). Também consegue suportar *tokens*, permitindo a criação de plataforma que operem com seus próprios ativos digitais ou até mesmo a geração de *tokens* não-fungíveis (Seção 2.4.5) associados a arquivos, como no caso da plataforma *Opensea* que comercializa arte digital utilizando NFTs associados as imagens [Joshua; Smith].

2.5 Inter-Planetery File System

Inter-Planetery File System (IPFS) é um protocolo descentralizado criado para realizar o armazenamento e acesso de arquivos, websites, aplicações e dados. Diferente da web convencional, onde os arquivos são acessados através do endereço de sua localização, O IPFS traz a proposta de endereçamento com base no conteúdo.

O endereçamento através da localização do arquivo traz uma questão de acesso onde ao requisitarmos uma certa página HTML ou imagem de um servidor, ele busca o arquivo naquele servidor e o retorna. Se o arquivo for substituído por outro de mesmo nome e extensão, mas

com um conteúdo totalmente diferente do original, a requisição irá retornar o novo arquivo alterado como se ele fosse o mesmo arquivo de sempre. Como a maioria dos servidores são centralizados, existe essa possibilidade, seja por parte dos donos ou de algum invasor que altere os arquivos.

O endereçamento através do conteúdo recorre à ideia de *hash* (Seção 2.3). O endereçamento por conteúdo gera um *Content Id* (CID) a partir dos bytes do arquivo e o torna único em toda a rede. Desse modo, sempre que você procurar o arquivo por seu CID ele sempre será o mesmo sem risco de ser trocado, pois um arquivo diferente, mesmo com uma alteração miníma, vai registrar um CID diferente.

Outra vantagem do IPFS sobre a web tradicional está na sua descentralização. Sites centralizados podem ter seus servidores fechados ou censurados em algum lugar que não queira que o sistema exista por lá ou simplesmente um dia pode só desistir e remover todos esses arquivos da web. Os arquivos registrados no IPFS estão espalhados pela rede e permitem que mesmo que a pessoa que publicou o arquivo não o possua mais, alguém em algum lugar o possui e pode compartilhar para quem quiser acessá-lo no futuro. A Figura 2.9 demonstra um comparativo entre os acessos tradicional e o IPFS.

O sistema foi nomeado de "Interplanetário", pois ele possui como objetivo permitir que, por exemplo, num futuro longínquo onde a humanidade comece a se expandir pelo espaço, alguém que esteja em Marte possa acessar sites que estejam na Terra, tendo que apenas demorar o tempo de requisição de um planeta a outro apenas uma vez, pois assim que alguém no planeta já esteja com acesso à página, a próxima pessoa que tentar acessar em marte já poderá requisitar do primeiro. isso torna o acesso mais rápido, diferente de ter que requisitar a um servidor na Terra toda vez com intervalos de requisição longos devido à distância entre os mundos [Benet; Explained (b); Docs; Dupres].

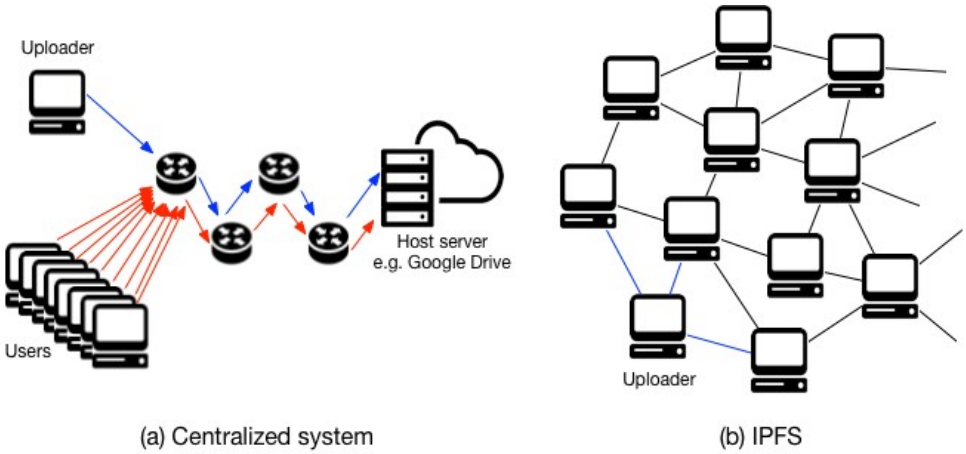


Figura 2.9: Esquema de acesso a um arquivo na web centralizada e no IPFS [zk Capital].

3

Registros de autorias baseados em NFTs

Neste capítulo será abordada uma solução para as problemáticas citadas na seção 1.1 visando uma melhoria na duração dos processos de registro e na expansão de seu alcance.

A proposta é utilizar das tecnologias *blockchain*, citadas na seção 2.4, para a implementação de uma plataforma online capaz de realizar os registros de autoria de forma rápida e com propriedades imutáveis e incontestáveis perante qualquer acusação. O uso dos NFTs (Seção 2.4.5) é a principal ferramenta para atingir os objetivos propostos.

Apesar de possuir foco no registro de PDFs, é importante ressaltar que as ideias aplicadas neste projeto são mais abrangentes no quesito de registro de obras na totalidade.

3.1 IP Register

Para este trabalho, foi desenvolvida uma aplicação web denominada de IP Register, uma plataforma que permite quem a utiliza possa acessar utilizando uma carteira de criptomoedas registar um documento no formato PDF na rede Ethereum na forma de um NFT e podendo listar todos os registros feitos até então pela carteira conectada.

3.2 Token IPR

Para este projeto, foi desenvolvido um modelo de *token* para ser utilizado na rede *Ethereum* que serve como identificador e validador do registro de autoria do artista que utilizar a plataforma. Ele foi batizado como IPR, sendo uma sigla para *Intellectual Property Register* (Registro de propriedade Intelectual em português), e possui o padrão ERC721, o padrão mais comuns utilizados na Ethereum para NFTs.

3.2.1 Estrutura dos Metadados

Como um NFT necessita de um objeto JSON na rede IPFS para gerar um registro, foi necessário conceber um modelo de chaves padronizadas para serem suportadas pela aplicação. A estrutura desenvolvida foi inspirada nos formulários utilizados pela Biblioteca Nacional (mostrados na figura 2.1) e possui as seguintes chaves:

- *document* - *String* correspondente ao URI de acesso ao documento dentro do IPFS;
- *image* - *String* correspondente ao URI de acesso da imagem da capa dentro do IPFS;
- *intellectualWork* - Objeto que armazena os dados referentes a obra a ser registrada;
- *applicants* - Lista que armazena objetos com dados referentes aos requerentes que desejam realizar o registro;
- *legalRepresentative* - Objeto que armazena os dados referentes ao representante legal em caso do requerente ser menor de idade;
- *adaptationOrTranslation* - Objeto que armazena os dados referentes a se a obra é uma adaptação ou tradução de outra obra já existente;
- *observations* - *String* que armazena observações registradas pelo requerente;
- *finalDispositions* - Objeto que armazena os dados referentes a concordância do requente em relação às declarações de responsabilidades necessárias antes da finalização do registro;

intellectualWork

- *name* - *String* que contém ao nome da obra a ser registrada;
- *description* - *String* que contém uma breve descrição sobre o que se trata a obra;
- *numberOfPages* - *Number* correspondente a quantidade de páginas presentes na obra;
- *publishedOrUnpublished* - *String* que diz se a obra é Publicada (*published*) ou Inédita (*unpublished*);
- *publishingCompanyData* - Objeto que armazena os dados referentes a editora responsável pela publicação da obra, caso ela seja marcada como “Publicada”, inclui os seguintes dados:
  - *editor* - *String* que contém o nome do Editor(a);
  - *graphic* - *String* que contém o nome da gráfica;

  - *editionNumber* - *Number* correspondente ao número da edição;
  - *year* - *Number* correspondente ao ano da publicação;
  - *publishingPlace* - *String* que contém o local da publicação;
  - *volume* - *Number* correspondente ao volume ou série;
- *documentType* - Lista de *Strings* que contém os gêneros referentes a obra.

applicants

É uma lista que pode conter quantos requerentes forem necessários, cada

- *name* - *String* que contém o nome do requerente;
- *rg* - *Number* correspondente ao RG do requerente;
- *issuingAgency* - *String* que contém o nome do órgão expedidor do documento de identidade;
- *cpfOrCnpj* - *Number* correspondente ao CPF ou ao CNPJ do requerente;
- *birthDate* - *String* correspondente a data de nascimento do requerente;
- *naturality* - *String* que contém a naturalidade do requerente;
- *nacionality* - *String* que contém a nacionalidade do requerente;
- *alias* - *String* que contém o pseudônimo/nome artístico do requerente, caso ele possua;
- *ocupation* - *String* que contém a ocuação do requerente;
- *educationDegree* - *String* que contém o grau de instrução do requerente;
- *motherName* - *String* que contém o nome da mãe do requerente;
- *address* - *String* que contém o endereço do requerente;
- *district* - *String* que contém o bairro do requerente;
- *county* - *String* que contém o município do requerente;
- *uf* - *String* que contém a Unidade Federativa do requerente;
- *zipCode* - *Number* que contém o CEP do requerente;
- *mobileNumber* - *Number* que contém o telefone do requerente;
- *email* - *String* que contém o e-mail do requerente;

- *site* - *String* que contém o site do requerente, caso possua;
- *linkWithWork* - *String* que contém o vínculo que o requerente possui com a obra, os possíveis valores são:
  - *Author* - Autor(a);
  - *Adapter* - Adaptador(a);
  - *Assignee* - Cessionário(a);
  - *Translator* - Tradutor(a);
  - *Illustrator* - Ilustrador(a);
  - *Organizer* - Organizador(a);
  - *Photographer* - Fotógrafo(a);
  - *Legal Representative* - Representante Legal;
  - *Assignor* - Cedente;
  - *Inheritor* - Herdeiro(a);
  - *Inventory* - Inventariante;
  - *Publisher* - Editor(a);

legalRepresentative

- *authorIsOlderThanEighteen* - *Boolean* correspondente a se o autor/requerente é maior de idade;
- *legalRepresentativeData* - Objeto que contém os dados referentes a obra original:
  - *name* - *String* que contém ao nome do representante legal;
  - *rg* - *Number* correspondente ao RG da pessoa;
  - *issuingAgency* - *String* que contém ao nome do órgão expedidor do documento de identidade;
  - *cpf* - *Number* correspondente ao CPF da pessoa;
  - *degreeOfKinship* - *String* contendo o grau de parentesco da pessoa com o requerente;

adaptationOrTranslation

- *isOriginalContent* - *Boolean* correspondente a se o trabalho é um conteúdo original;
- *adaptationOrTranslationData* - Objeto que contém os dados referentes a obra original:
  - *isTranslation* - *Boolean* correspondente a se a obra é uma tradução;

- *isAdaptation* - *Boolean* correspondente a se a obra é uma adaptação;
- *originalTitle* - *String* correspondente ao título da obra original;
- *originalAuthors* - *String* correspondente ao nome do autor da obra original;

finalDispositions

- *agreement01* - *Boolean* que representa se o usuário concordou com o primeiro Termo;
- *agreement02* - *Boolean* que representa se o usuário concordou com o segundo Termo;

A Figura 3.1 mostra um exemplo de metadados gerado salvo dentro do IPFS. Devido à extensão não caber na imagem, apenas os dados da chave *intellectualWork* são exibidos, as outras chaves estão recolhidas.

Figura 3.1: Metadados da aplicação salvos no IPFS.

3.3 Arquitetando a aplicação

A aplicação consiste em 2 partes fundamentais:

- *Front-end*
- *Back-end*

Cada uma delas desenvolvidas em um repositório a parte e integradas para o funcionamento geral da plataforma. Além delas, há também um contrato Inteligente que provê a geração de NFTs na rede *Ethereum*, o qual será detalhado na Seção 3.4;

3.3.1 Back-end

Para a aplicação, será necessário que haja uma API simples que possa realizar um CRUD básico de usuários, mantendo as informações em um banco de dados relacional. Também deve ser possível relacionar usuários e carteiras em uma relação de 1 para N, onde 1 usuário pode ter diversos endereços de carteira associados a sua conta e isso determinará sua autenticação ao logar na plataforma.

Também é necessário salvar os *hashes* das transações efetuadas para facilitar o acesso às informações referentes ao bloco onde o registro se encontra na rede. Esses dados serão armazenados em uma tabela de registros que também armazena o id do usuário, a carteira utilizada, o CID dos metadados salvos no IPFS.

Para isso foi escolhido o *NestJS*, em sua versão 8.2.5, como *framework* de desenvolvimento *back-end*, pois facilita a criação dos sistemas de CRUD para usuários, com *Sequelize* sendo a forma de Mapeador de Objetos Relacionais para integração com banco de dados SQL. Tais escolhas de tecnologias foram pela familiaridade prévia com o mesmo em outros projetos.

3.3.2 Front-end

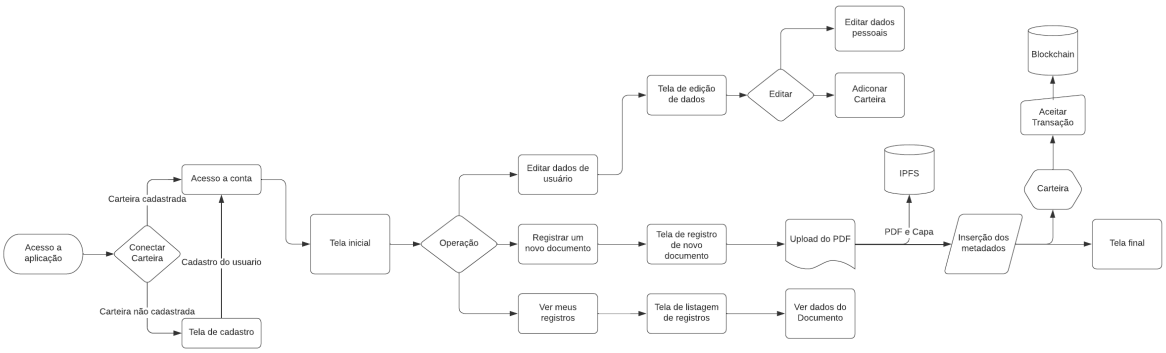


Figura 3.2: Fluxograma proposto para a Aplicação.

A aplicação deve conter um conjunto de telas que possa executar as seguintes tarefas:

- Permitir que o usuário possa conectar sua carteira Metamask;
- Permitir que o usuário possa realizar um cadastro na plataforma;
- permitir que o usuário possa Editar seus dados cadastrados;
- Permitir que o usuário possa Realizar um registro de NFT na rede Ethereum;
- Permitir que o usuário possa ver os registros feitos pela sua carteira no contrato relacionado a rede atualmente conectado;

- Bloquear o acesso do usuário as funcionalidades caso ele não esteja conectado e devidamente cadastrado.

O usuário deve poder se conectar a aplicação utilizando tanto uma carteira Metamask no navegador como também em seu celular. Quando ele conectar sua conta, a aplicação deve verificar se a carteira já consta na base de dados de usuários e decidir entre mandá-lo para tela inicial caso conste, ou mandá-lo para uma tela de cadastro em caso negativo. Uma vez devidamente conectado e cadastrado, devem ser disponibilizado para o usuário o acesso as outras partes da plataforma. Após 30min conectado, a sessão deve expirar e ele deve conectar-se novamente.

Para o desenvolvimento das telas descritas abaixo, foi escolhido o *framework* Angular, em sua versão 13.2.7, utilizando HTML como linguagem de marcação para a estrutura visual das telas, SCSS para a estilização e *Typescript* para a lógica de programação interna. A escolha pelo Angular se deu pela familiaridade e maior experiência com o mesmo. A Figura 3.2 retrata o fluxograma proposto para a aplicação com base no que foi abordado.

Tela de meus registros

A listagem deve apresentar os documentos registrados pela conta no contrato associado a rede em que ele esteja conectado atualmente. Os itens devem permitir que o usuário possa ver seus detalhes ao clicar em um deles e deve permitir o acesso ao documento dentro IPFS para leitura.

Tela de registrar documentos

Para realizar um registro o usuário deve seguir os seguintes passos:

- Fazer o upload de um PDF para a plataforma;
- Realizar a inserção dos metadados em um formulário;
- Confirma a transação para o contrato.

O PDF deverá ser enviado ao IPFS com uma imagem correspondente a sua capa para servir de vitrine para o NFT na listagem de registros tanto na aplicação quanto na sua carteira. Os CIDs de ambos os arquivos devem ser recuperado e registrados no JSON de metadados como um URI de acesso do *Piñata*. Quando os metadados forem registrados no IPFS deve ser acionada a requisição de transação para o contrato inteligente utilizando o seguinte formato:

- *from* - Endereço da carteira do usuário;
- *to* - Endereço do contrato inteligente;
- *nonce* - Valor de *balance* atual do contrato;

- *gas* - Valor de taxa miníma para a transação;
- *data* - Codificação da ABI da chamada do método de *mintNft* recebendo os parâmetros que serão processados pelo contrato;

Tela de editar de dados

A edição deve permitir que qualquer dado cadastrado pelo usuário possa ser alterado e o formulário só poderá ser enviado caso os dados inseridos sejam validos. As carteiras pertencentes ao usuário serão listadas e o mesmo poderá cadastrar e remover carteiras de sua conta.

3.4 Criação do Contrato Inteligente

Como a rede principal do projeto é a Ethereum, a linguagem para a escrita do contrato é a *Solidity*. Sua principal funcionalidade é a função chamada *mintNFT* a que é a responsável pela geração de NFTs na rede e associá-los as carteiras que requisitarem.

O contrato é de livre acesso, permitindo que qualquer usuário, sem ser o criador do contrato, possa usufruir dos serviços do mesmo. Além disso, ele permite que o usuário possa requisitar os registros que já tenha feito na plataforma para visualizá-los na plataforma.

Para facilitar o desenvolvimento, foi utilizado o auxílio de um tutorial da própria Ethereum sobre como criar um contrato inteligente para gerar NFTs. Um projeto em Node foi gerado utilizando as bibliotecas *Hardhat* e *Ethers.js* para facilitar o processo de *deploy* do contrato para rede *Ropsten*.

O projeto conta com uma pasta de *scripts*, onde se localiza o código para a realização do *deploy*, e uma pasta de *contracts*, onde se localiza o código em *Solidity* d contrato desenvolvido. O contrato se utiliza de uma biblioteca chamada Openzeppelin, que será melhor elaborado mais a frente, para manter uma estrutura padronizada aos modelos esperados pela rede. Em seu construtor foram definidos o nome do contrato (IP-Register) e o símbolo do *Token* (IPR).

Após feita sua compilação, foram geradas uma pasta de cache e uma de *artifacts*, nela é possível encontrar o arquivo que contém a ABI do contrato, a qual será copiada e inserida no repositório contendo a interface da aplicação para uso posterior. Após compilado, o contrato teve seu *deploy* realizado e se encontra atualmente na rede *Ropsten* e pode ser acessado pelo *scan* nesse link.

3.4.1 Openzeppelin

É uma biblioteca *open-source* que traz modelos prontos de contratos inteligentes em *Solidity* tendo foco no reuso dos códigos para auxiliar a construção novos contratos. Para utilizar é necessário fazer com que o contrato herde os modelos colocando na declaração do contrato,

assim garantindo acesso às funções básicas. No contrato desenvolvido para este projeto foi necessário herdar os modelos ERC721, ERC721Enumerable e ERC721URIStorage.

O ERC721 é o modelo de contrato padrão que permite que os NFTs gerados por ele sejam reconhecidos pela rede como "válido", pois existem métodos de base que eles devem possuir e esse modelo já traz para o contrato via herança.

ERC721URIStorage é o modelo responsável por prover o armazenamento dos URIs enviados para gerar os NFTs dentro da *chain* em que o contrato está operando.

O ERC721Enumerable é o modelo que traz as funções necessárias para a enumeração e acesso dos *tokens* salvos na carteira através do código. Ela dá a cada *token* registrado um valor de índice permitindo acesso ao URI associado. Sua implementação traz os métodos *_beforeTokenTransfer*, *_burn*, *tokenURI* e *supportsInterface*, porém apenas o método *tokenURI* foi necessário para execução deste projeto e será abordado em mais detalhes mais a frente.

Outra importação utilizada a de *Counters*, que provê para o contrato uma variável contadora que registra a quantidade de registros de NFT realizados. Ela pode ser incrementada ou decrementada a medida que os *tokens* forem gerados ou forem descartados. No escopo do projeto, a variável *_tokenIds* é a responsável pela contagem de NFTs e toda vez que função responsável por gerá-los (que será abordada mais a frente) é chamada o método *increment* é chamado para incrementar o valor do contador.

3.4.2 Função mintNFT

A função-chave para este projeto, pois é a responsável por realizar o registro do NFT na rede. Ela recebe dois parâmetros: o primeiro é o endereço da carteira que está querendo "mintar" o NFT e o segundo é o URI dos metadados que o usuário cadastrou no formulário de Metadados da aplicação.

Algoritmo 3.1 Função *mintNFT* para gerar o registro da rede

```
1  function mintNFT(address recipient, string memory tokenURI)
2      public
3      returns (uint256)
4  {
5      _tokenIds.increment();
6
7      uint256 newItemId = _tokenIds.current();
8      _mint(recipient, newItemId);
9          _setTokenURI(newItemId, tokenURI);
10
11     return newItemId;
12 }
```

3.4.3 Função tokenURI

A função responsável por retornar o URI de um NFT que esteja na carteira do usuário e que tenha sido gerado por este contrato a partir de seu índice. Recebe de parâmetro o valor do índice do *token* na carteira, começando do 0, e retorna uma *string* correspondente a URI de onde os metadados associados a este *token* estão salvos.

Algoritmo 3.2 Função *tokenURI* para retornar o URI dos metadados

```
1 function tokenURI(uint256 tokenId)
2     public
3     view
4     override(ERC721, ERC721URIStorage)
5     returns (string memory)
6 {
7     return super.tokenURI(tokenId);
8 }
```

3.5 Conexão com a Web3

Para utilizar poder usufruir dos serviços que se utilizam das *blockchains*, os usuários devem possuir uma carteira de criptomoedas que seja capaz de se conectar a rede em que o contrato inteligente necessário esteja armazenado. Uma das carteiras mais comumente utilizadas pelos usuários desse tipo de plataforma, especialmente as que lidam com NFTs é a Metamask, por isso ela foi escolhida como a carteira principal para o desenvolvimento e os testes de usabilidade da aplicação deste trabalho.

3.5.1 Rede de testes

Os *deploys* de contratos e as transações que serão apresentadas foram todas realizadas dentro da *testnet* da *Ethereum Ropsten* para uma melhor simulação de como a plataforma operaria em um ambiente de produção real de uma *mainnet*.

3.5.2 Comunicação com IPFS

Para realizar o envio dos arquivos e metadados necessários parta geração do NFT ao IPFS, será utilizado um serviço que atua como *Gateway* entre a rede web2, que atua com protocolo https, e a rede web3, q atua com o IPFS, chamado *Piñata*.

O *Piñata* é um serviço amplamente utilizado no ramo das NTFs para comportar as imagens de base sendo descrito como atamente escalável e seguro. Sua API fornece *endpoints* para realizar o chamado *pinning* dos arquivos na rede descentralizada, tanto documentos como JSONs,

e retorna seus *contentIds*, além de também permitir o acesso via *fetch*() da URL para requisitar os dados armazenados. A Figura 3.3 mostra o *dashboard* do *Piñata*.

A escolha de utilizar esse *gateway* foi devido a sua praticidade e por no início do processo de desenvolvimento da aplicação ter começado com a tentativa de desenvolver uma forma de realizar a conexão entre o Angular e o IPFS mais diretamente por bibliotecas *Javascript*, porém após vários impedimentos e percalços com a integração das tecnologias, foi decidido mudar a abordagem.

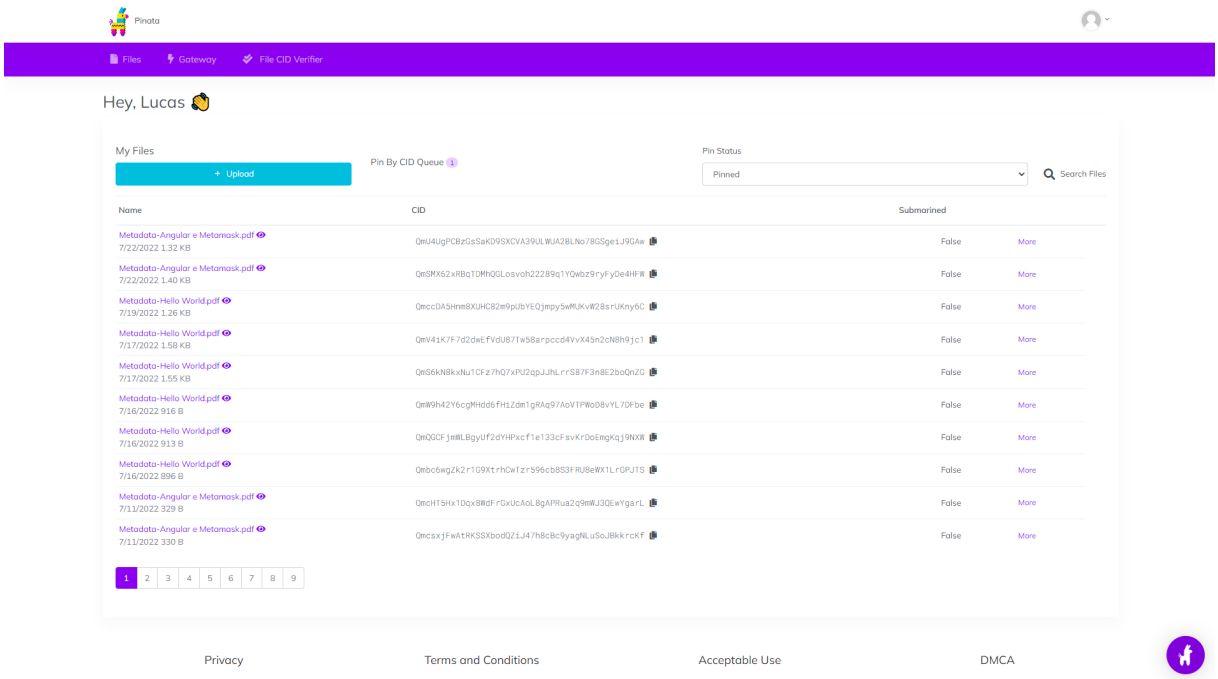


Figura 3.3: *Dashboard* do *Piñata* .

3.5.3 Binance

Outra rede que será abordada de forma secundária neste projeto é a rede de testes da *Binance Smart Chain* (BSC). O objetivo disso é demonstrar que o trabalho desenvolvido não precisa se limitar apenas a rede Ethereum e pode ser distribuído para outras redes, a escolha da BSC *testnet* foi por ela também aceitar contratos escritos na linguagem *Solidity*, possibilitando o reuso do mesmo contrato já desenvolvido.

4

Resultados e Discussões

Com base no que foi discutido no capítulo 3, nesta seção serão apresentados os resultados obtidos da implementação da aplicação, apresentando um acompanhamento visual do resultado e explicando seu funcionamento em cada etapa.

4.1 Back-end

Criada com *NestJS*, ela consegue realizar as operações de CRUD de usuários, carteiras, registros e relacioná-las, de modo que um usuário pode ter várias carteiras, assim como também ter vários registros.

4.1.1 Users

O módulo de *users* é o responsável pelo CRUD de usuários e de sua associação às carteiras. Nele há os DTOs (*Data Transfer Object*), os modelos utilizados para comunicação com banco via *Sequelize*, o arquivo *controller*, que possui os *endpoints* de acesso para comunicação com o *front-end*, e o arquivo de *service*, onde as operações são codificadas.

***Endpoints* para usuário**

- POST 'users'
  - Para criar um usuário.
- GET 'users'
  - Para recuperar todos os usuários.
- GET 'users/:id'

  - Para recuperar os dados de um usuário já existente.

- PATCH ’users/:id’
  - Para editar os dados de um usuário já existente.
- DELETE ’users/:id’
  - Para apagar os dados de um usuário já existente.

Endpoints para carteiras

- POST ’users/wallet’
  - Para vincular uma nova carteira ao usuário.
- GET ’users/wallet/:walletAddress’
  - Para validar se a carteira está vinculada a algum usuário.
- DELETE ’users/wallet/:walletAddress’
  - Para remover uma carteira.

create

- Recebe o DTO com dados do usuário como parâmetro.
- Cria um usuário no banco, ela recebe um DTO com os dados cadastrados pelo usuário e o endereço da carteira usada para acessar a aplicação, a carteira é separada do objeto enviado pro *Sequelize* gerar uma linha nova na tabela de *users*, após ele gerar o usuário o id dele é recuperado e mandado junto do endereço da carteira para a tabela *user-wallets* e gerar uma linha nova vinculando o usuário e a carteira.
- Retorna o novo usuário.

findAll

- Não recebe parâmetros.
- Retorna todos os usuários cadastrados.

findOne

- Recebe o *id* do usuário a ser buscado como parâmetro.
- Retorna os dados do usuário requisitado.
- Caso usuário não exista retorna um objeto de erro *message* sendo “Usuário não encontrado.” e *status* NOT_FOUND.

update

- Recebe o *id* do usuário a ser editado e o DTO com os dados novos como parâmetro.
- Retorna um objeto com a chave *message* e o valor “Usuário atualizado com sucesso!” em caso de sucesso.
- Caso usuário não exista retorna um objeto de erro *message* sendo “Usuário não encontrado.” e *status* NOT_FOUND.

remove

- Recebe o *id* do usuário que será apagado como parâmetro.
- Retorna um Promise<void>.
- Caso usuário não exista retorna um objeto de erro *message* sendo “Usuário não encontrado.” e *status* NOT_FOUND.

createWallet

- Recebe o DTO com dados da carteira como parâmetro.
- Retorna a nova carteira.

findUserByWallet

- Recebe o endereço da carteira que será apagada como parâmetro.
- Retorna o usuário cuja conta esteja associada a esta carteira
- Caso usuário não exista retorna um objeto de erro *message* sendo “Usuário não encontrado.” e *status* NOT_FOUND.

removeWallet

- Recebe o endereço da carteira que será apagada como parâmetro.
- A função busca o usuário dono da carteira e conta quantas carteiras ele tem, se ele apenas possuir uma, a operação é cancelada pois o usuário precisa ter pelo menos uma carteira associada a ele, caso ele possua mais carteiras, ela sera apagada normalmente.
- Retorna uma *Promise<void>* em caso de sucesso.
- Caso usuário possua apenas 1 carteira retorna um objeto de erro *message* sendo "Usuário precisa possuir pelo menos uma carteira." e *status* UNAUTHORIZED.

4.1.2 Registers

O módulo de *registers* é o responsável pelo CRUD dos registros criados pelos usuários. Ele também possui DTOs e modelos utilizados para comunicação com banco, sendo que ara sua criação são necessários o *id* do usuário e a carteira que ele esta utilizando e ara realizar as atualizações, o *contentId* dos metadados e o *hash* da transação que só serão obtidos posteriormente na operação de registro (Seção 4.2.6).

***Endpoints* para registers**

- POST 'registers'
  - Para criar um registro.
- GET 'registers'
  - Para recuperar todos os registros.
- GET 'registers/txhash/:metadataCid'
  - Para recuperar os *hash* de uma transação de um registro a partir de seus metadados.
- GET 'registers/:id'
  - Para recuperar os dados de um registro já existente.
- PATCH 'registers/:id'
  - Para editar os dados de um registro já existente.
- DELETE 'registers/:id'
  - Para apagar os dados de um registro já existente.

create

- Recebe o DTO com dados do registro como parâmetro.
- Retorna o novo registro.

findAll

- Não recebe parâmetros.
- Retorna todos os registros cadastrados.

findOne

- Recebe o *id* do registro a ser buscado como parâmetro.
- Retorna os dados do registro requisitado.
- Caso registro não exista retorna um objeto de erro *message* sendo “Registro não encontrado.” e *status* NOT_FOUND.

update

- Recebe o *id* do registro a ser editado e o DTO com os dados novos como parâmetro.
- Retorna um objeto com a chave *message* e o valor “Registro atualizado com sucesso!” em caso de sucesso.
- Caso registro não exista retorna um objeto de erro *message* sendo "Registro não encontrado."e *status* NOT_FOUND.

remove

- Recebe o *id* do registro que será apagado como parâmetro.
- Retorna um *Promise<void>*.
- Caso registro não exista retorna um objeto de erro *message* sendo “Registro não encontrado.” e *status* NOT_FOUND.

findHashWithMetadataCid

- Recebe o *contentId* dos metadados do registro parâmetro.
- Retorna a *string* correspondente ao *hash* da transação.
- Caso registro não exista retorna um objeto de erro *message* sendo “Registro não encontrado.” e *status* NOT_FOUND.

4.1.3 Banco de dados

O banco é relacional, utilizando MySQL e se comunica com a API através do *Sequelize*. O banco possui três tabelas: *users*, *user_wallets* e *registers*.

A tabela *users* armazena os dados de usuários cadastrados na plataforma. A tabela *user_wallets* armazena as carteiras cadastradas por cada usuário e a relaciona com o id do usuário na tabela *users*. A tabela *registers* armazena os dados dos registros realizados por usuários na plataforma. A Figura 4.1 mostra o modelo entidade-relacionamento das tabelas do banco.

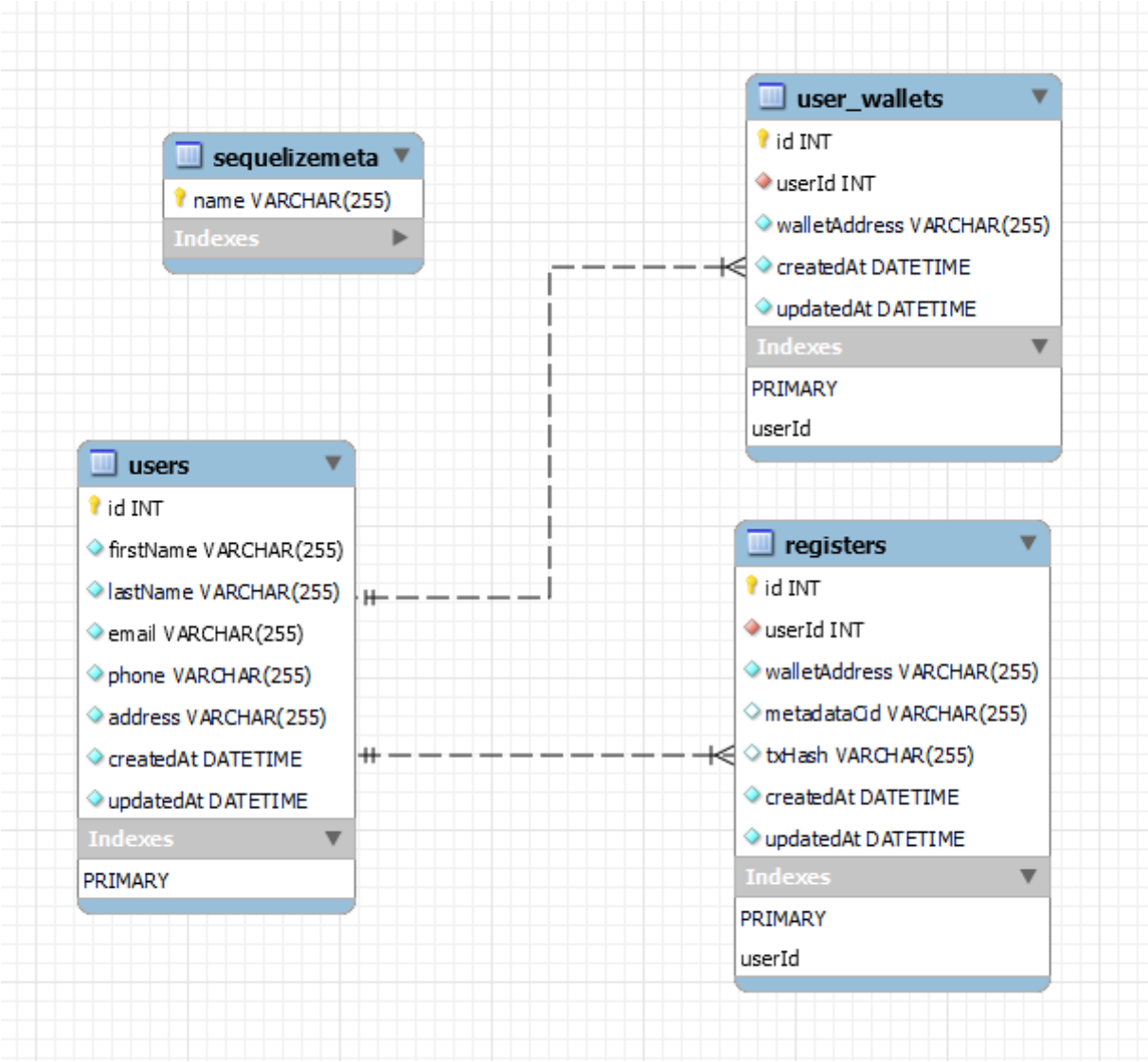


Figura 4.1: Modelo ER do Banco de Dados.

A tabela *users* possui como colunas os atributos: *id*, *firstName*, *lastName*, *email*, *phone*, *address*, dados que o usuário preenche em um formulário no *front-end* que são enviados a API para serem cadastradas.

A tabela *user_wallets* possui como colunas os atributos: *id*, *userId*, *walletAddress*, sendo *userId* uma chave estrangeira que relaciona o endereço da carteira na linha ao usuário com id

equivalente na tabela de usuários, permitindo que haja uma relação de 1 para N entre as tabelas, onde 1 usuário pode ter várias carteiras.

A tabela *registers* possui como colunas os atributos: *id*, *userId*, *walletAddress*, *metadaCid*, *txHash*, sendo *userId* uma chave estrangeira da tabela *users* e os dados de *metadaCid* e *txHash* são obtidos posteriormente a criação da linha na tabela, podendo ser nulos e indicando que o usuário não chegou a finalizar aquele registro.

4.2 Interface da aplicação

Nesta seção serão abordadas as telas desenvolvidas para a aplicação explicando suas funcionalidades, as tecnologias empregadas e a arquitetura dela.

4.2.1 Tela inicial

Esse componente consiste em apenas 2 botões que permitem o usuário escolher como deseja se conectar a aplicação, sendo elas pela extensão da *Metamask* em seu navegador ou pela *Wallet Connect*, a qual permite conectar com carteiras mobile (mais detalhes na seção 4.4). A Figura 4.2 retrata a tela inicial da aplicação desenvolvida.

Uma vez que o usuário escolha a forma de conexão, ele seguirá o fluxo para vincular a carteira a aplicação e um serviço de autenticação é disparado onde o endereço da carteira conectada é recuperado e enviado a API para validar se o mesmo já consta no banco de dados da aplicação, caso se encontre cadastrado, a tela é atualizada e o acesso as outras partes do projeto são liberados na barra do topo (Seção 4.2.3), caso não seja encontrado o endereço no banco, ele será redirecionado a tela de cadastro.

Uma vez que esteja conectado, o botão da opção escolhida se torna um botão de desconectar e a outra opção fica bloqueada. Ao se desconectar pelo *Wallet-Connect* a sessão é encerrada. Já pela *Metamask*, apenas os dados locais de conexão são apagados, mas a carteira mantém a conexão, caso seja clicado para conectar novamente ele irá conectar direto sem pedir a carteira.

Para uma desconexão completa é necessário ir até à carteira e desconectar manualmente, a aplicação detecta caso ainda esteja conectada e recarrega a página para remoer os dados locais de conexão. Essa impossibilidade de desconectar a *Metamask* direto pela aplicação é comum em sistemas que a utilizam, por exemplo na plataforma *Opensea*, sendo uma das mais notórias no ramo de NFTs, pode se observar esse fenômeno com a conexão via *Metamask*.

4.2.2 Tela de Cadastro

Esse componente conta com um formulário para preenchimento de dados básicos da pessoa que deseja acessar a plataforma, são eles:

- Nome;

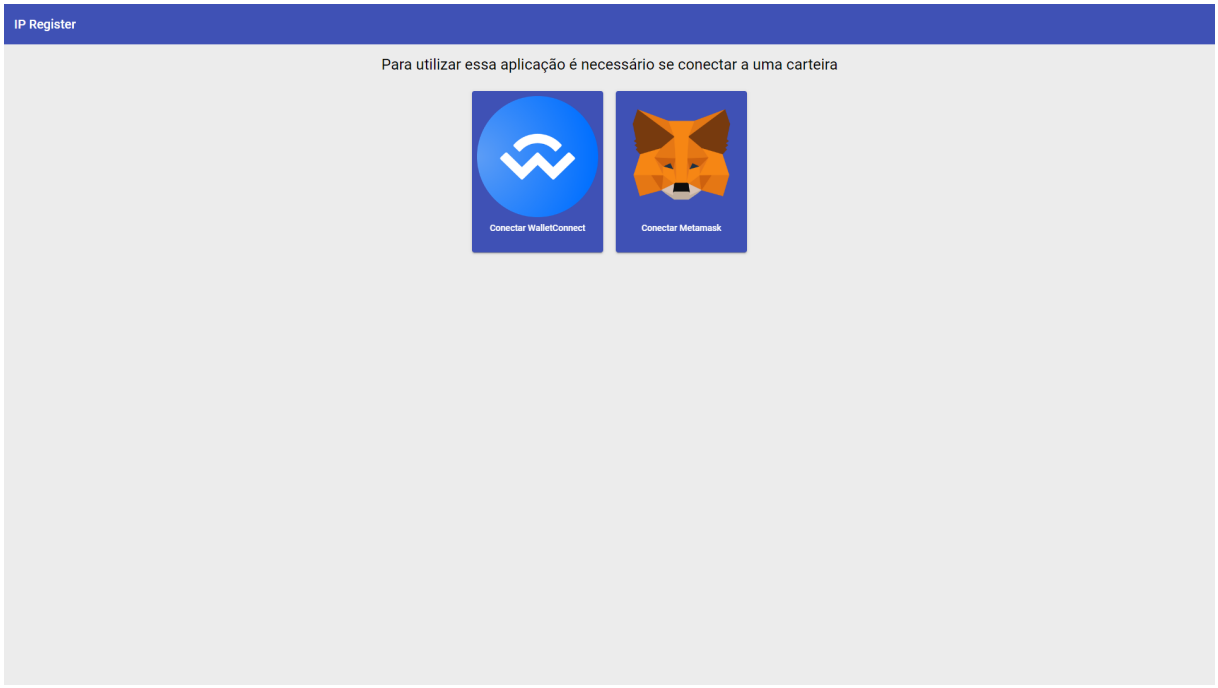


Figura 4.2: Tela Inicial.

- Sobrenome;
- E-mail;
- Telefone;
- Endereço.

Além dos dados da pessoa, existe uma *checkbox* com um "Termo de responsabilidade". Ao marcá-lo, o usuário declara que qualquer documento que registre é de sua autoria e cabe somente a ele responder pro possíveis complicações jurídicas caso faça mau uso da plataforma e que a mesma está isenta dessas questões. A Figura 4.3 retrata a tela de cadastro da aplicação.

Após os dados serem preenchidos e o Termo foi aceito, o botão de "Cadastrar" é habilitado para que os dados serão enviados ao banco de dados e logo em seguida o usuário será redirecionado a tela inicial, caso tudo ocorra bem, ele poderá tentar se conectar novamente e assim poder usufruir da aplicação.

4.2.3 Barra do topo

Esse componente, a princípio, aparece vazio com apenas o nome do projeto no canto esquerdo, uma vez que o usuário conecte uma carteira e esteja cadastrado, as opções de acesso as outras partes da plataforma serão liberadas, elas são:

- Editar Dados (Seção 4.2.4);

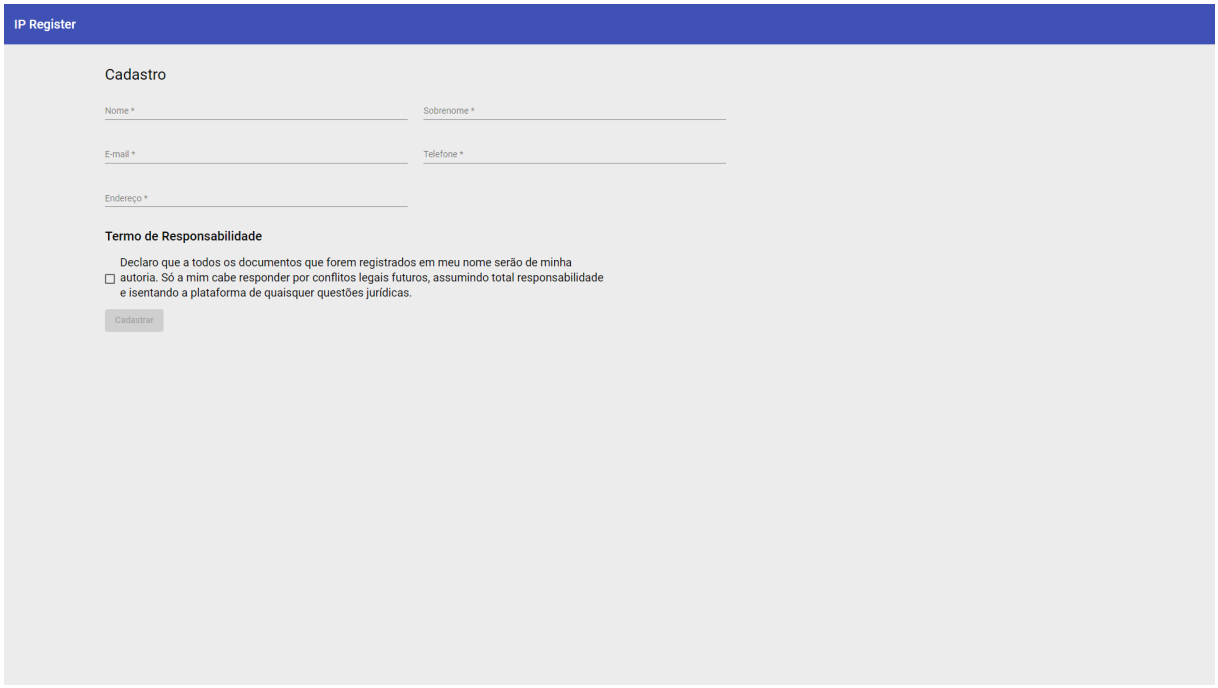


Figura 4.3: Tela de cadastro.

- Meus Registros (Seção 4.2.5);
- Registrar Documentos (Seção 4.2.6);

Esse componente também é responsável por monitorar a atividade da carteira do usuário por ser o único que está presente em todo o projeto. existem eventos de carteira como troca de conta ou troca de rede que ocorrem fora do controle da aplicação. O serviço de *WalletService* (seção 4.4.3) é declarado em seu construtor permitindo que os eventos da carteira sejam detectados a qualquer momento.

4.2.4 Editar Dados

Esse componente permite que o usuário veja os dados que cadastrou na tela de cadastro (Seção 4.2.2), editá-los e adicionar ou remover endereços de carteiras associados a sua conta. Essa tela pode ser constatada na Figura 4.4.

4.2.5 Meus Registros

Esse componente permite a visualização dos documentos registrados pelo usuário na carteira na rede a qual ele estiver conectado no momento. No topo da página está o endereço do contrato utilizado para registrar esses arquivos.

A exibição dos registros é em forma de uma tabela composta por 5 colunas em ordenados pelo *id*, dispostos no sentido da esquerda para direita. Cada documento é exibido com a ima-

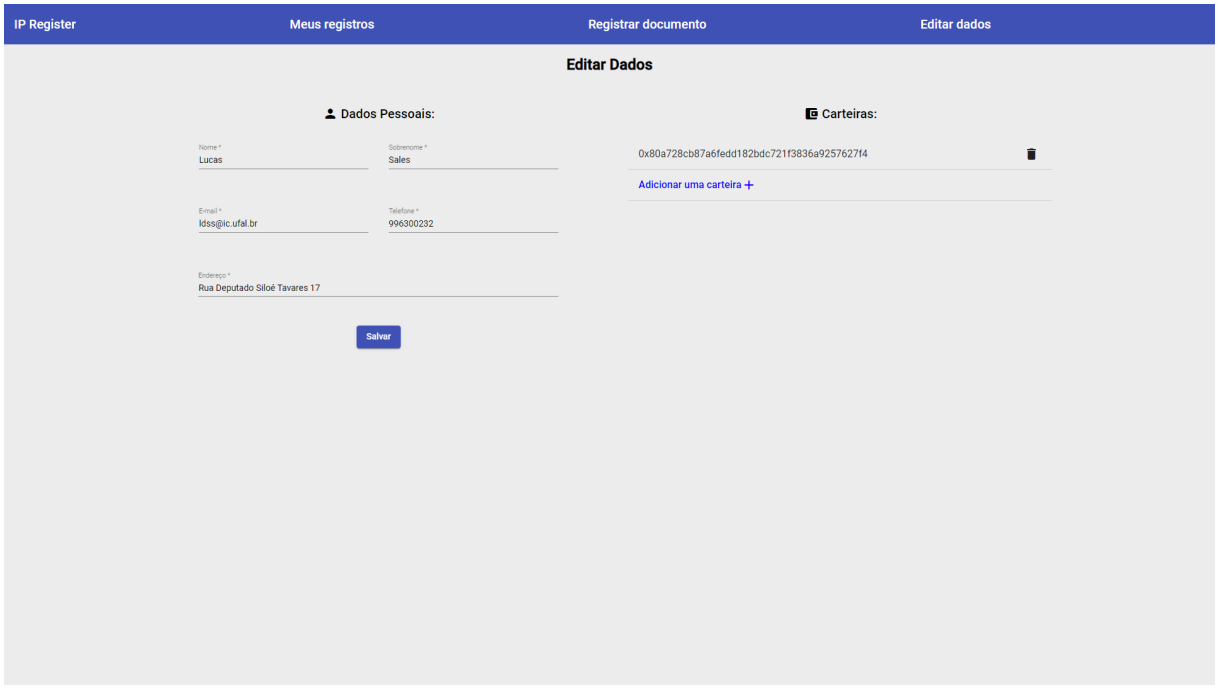


Figura 4.4: Tela de dados do usuário.

gem de sua capa e o título registrado no formulário de metadados (Seção 4.2.6). A Figura 4.5 demonstra a tela em questão.

Ao clicar em algum item listado, um modal é aberto exibindo a imagem da capa no lado esquerdo e no lado direito os metadados cadastrados (Seção 4.2.6) e o *id* do *token* registrado na rede.

A capa pode ser clicada, assim abrindo uma nova guia no navegador onde o documento é aberto através do link de acesso para o mesmo no IPFS através do *gateway Piñata*. Também são exibidas as informações sobre o número do bloco em que o registro de encontra e o *timestamp* associado a ele. A Figura 4.6 retrata a visualizações dos dados citados na aplicação.

4.2.6 Registrar Documentos

Esse componente é o responsável pelo "coração" do projeto, o registro da NFT na *blockchain*. Essa tela possui um componente de *stepper* com 8 passos, cada passo está associado a um componente filho desse, eles são:

- *Upload* do Arquivo: componente *file-uploader* (Figura 4.7);
- Dados da Obra: componente *intelectual-work-form* (Figura 4.9);
- Dados dos Requerentes: componente *applicant-form* (Figura 4.10);
- Representante Legal: componente *legal-representative-form* (Figura 4.11);
- Adaptação ou Tradução: componente *adaptation-or-translation-form* (Figura 4.12);

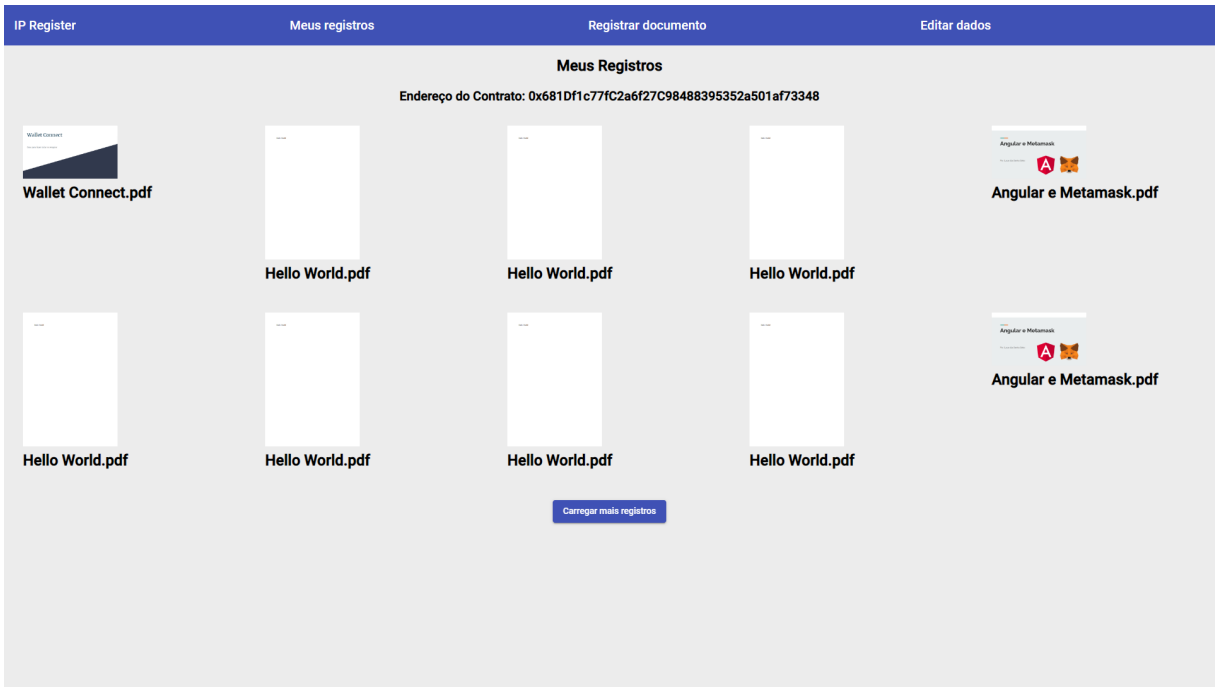


Figura 4.5: Tela de registros do usuário.

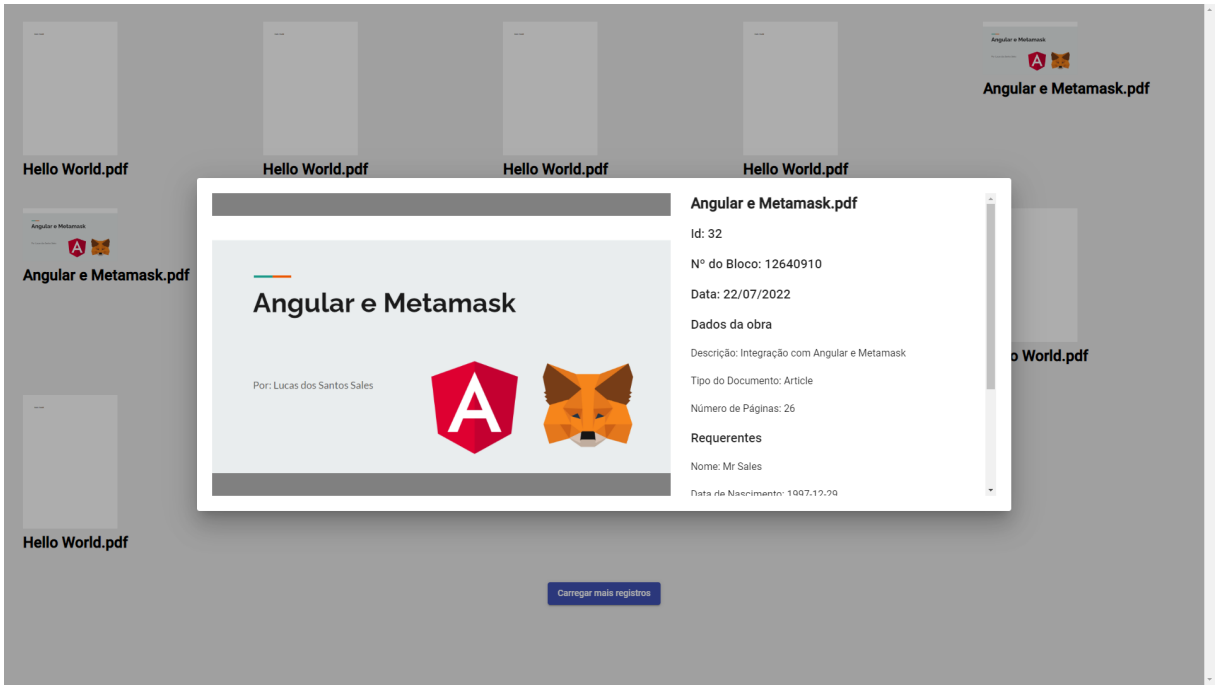


Figura 4.6: Detalhes de um registro específico.

- Observações: componente *observations-form* 4.13;
- Disposições Finais: componente *final-dispositions* (Figura 4.14);
- Final: componente *mint-end* (Figura 4.15);

O componente pai, *mint-nft*, possui a variável *registerMetadata*, do tipo *FormGroup*, para armazenar os metadados necessários para o registro. Ele faz o papel do formulário de registro da Biblioteca Nacional, tendo os mesmos campos pedidos para os requerentes escreverem os dados associados a obra e aos envolvidos.

A cada passo, os componentes filhos utilizam um evento de *Output* para evocar a função *setInMetadata* que recebe um pedaço do formulário vindo dos componentes filhos e utiliza um *patchValue* para armazená-los na chave equivalente do formulário principal.

O primeiro passo é onde o usuário insere o arquivo que será registrado na operação. Ele possui uma área onde é possível arrastar o arquivo para dentro dela ou clicar para procurar pelo explorador dentro computador. Uma vez que o arquivo seja adicionado o nome dele é exibido logo acima da área e abaixo dela é possível ter uma visualização do documento inserido, sendo possível navegar entre as páginas dele.

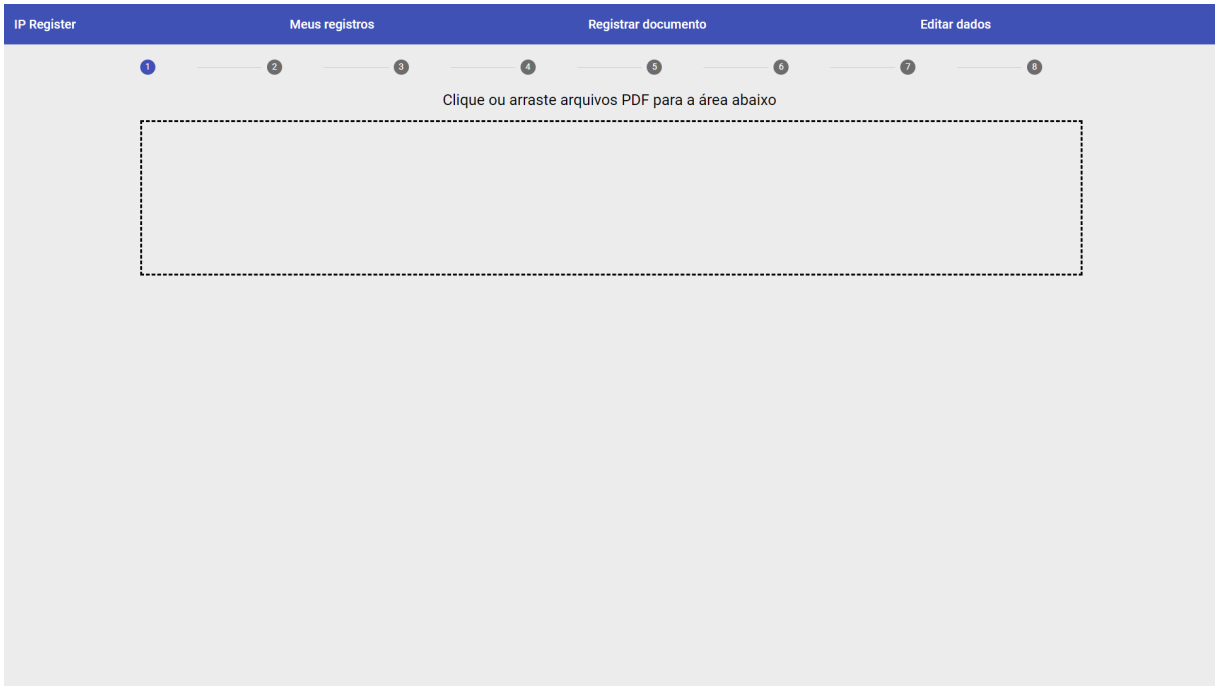


Figura 4.7: Tela para submeter a obra.

Uma vez que o documento esteja selecionado, é possível prosseguir para o próximo passo apertando o botão de "Enviar", o qual irá iniciar o envio do documento para o IPFS, a função *sendFile()* evoca o método *getCover* para recuperar a imagem da capa do PDF na visualização do *canvas*.

Após isso, os métodos *sendDocument* e *sendImage*, que usam o *IpfsService* para enviar o arquivo para o *Piñata*, são chamadas. Cada uma retorna um *contentId* associado a cada arquivo e ambos são emitidos de volta ao componente pai para serem salvos no formulário principal nas chaves *document* e *image*.

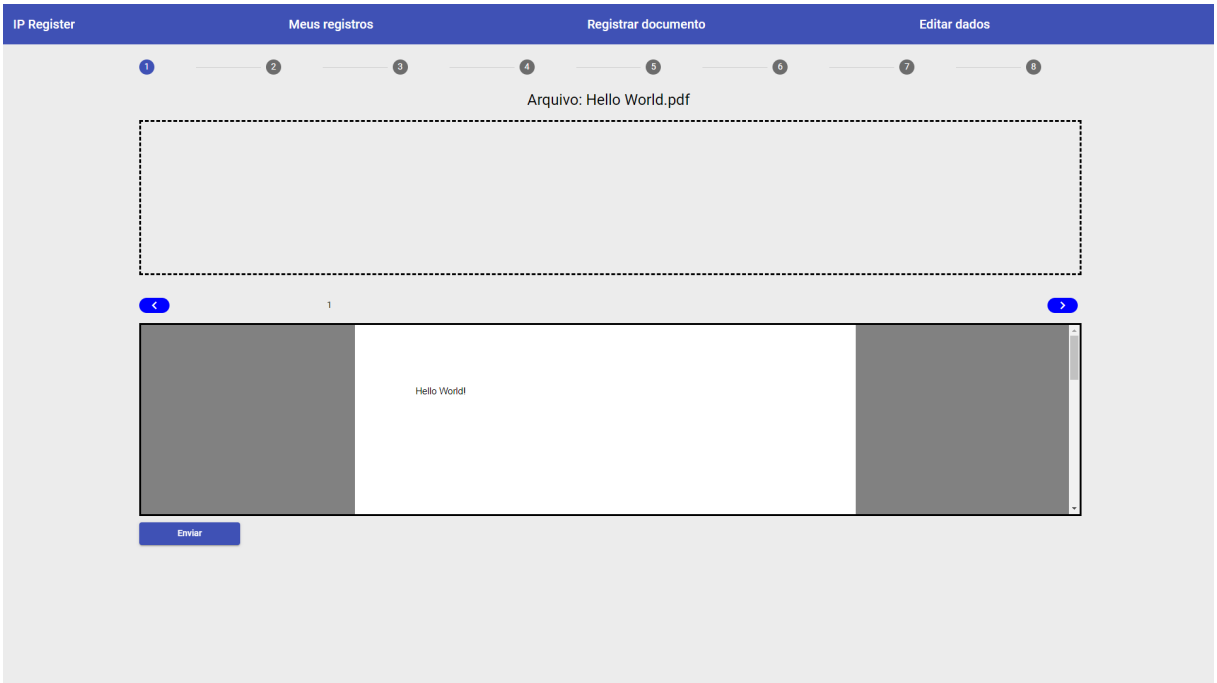


Figura 4.8: Tela de registrar documento exibindo pdf.

O segundo passo apresenta um formulário onde o usuário deve inserir os dados associados obra. Os campos requeridos são: nome da obra, gênero, número de páginas e resumo. Também há um campo ara marcar se obra em questão é "Publicada" ou "Inédita". Caso o usuário selecione a opção "Publicada", uma nova seção será exibida pedindo os dados relacionados a editora responsável pela publicação. Os dados pedidos da Editora são: editor(a), gráfica, número da edição, ano, local da publicação e volume/série.

Os dados referentes ao nome do documento e ao número de páginas são preenchidos automaticamente mas o usuário, se preferir, pode atribuir outro nome ao NFT, o qual será utilizado para a exibição no componente de Meus registros (Seção 4.2.5). O campo de gênero possui um *select* que contém as mesmas opções demonstradas no formulário da Biblioteca Nacional, como constatados na Figura 2.1, sendo possível marcar mais de uma opção.

O terceiro passo é onde os dados dos requerentes são registrados. Existe um botão de + nas abas que permite adicionar mais requerentes ao registro e, caso exista mais de um requerente e a aba dele esteja ativa, um ícone de lixeira é exibido que permite remover o requerente. Cada aba exibe um formulário com os campos: nome, RG, órgão expedidor, CPF/CNPJ, data de nascimento, naturalidade, nacionalidade, pseudônimo/nome artístico, ocupação, grau de instrução, nome da mãe, endereço, bairro, município, UF, CEP, telefone, e-mail, site e vínculo com a obra.

Os dados que forem em comum com os de cadastro são automaticamente preenchidos no

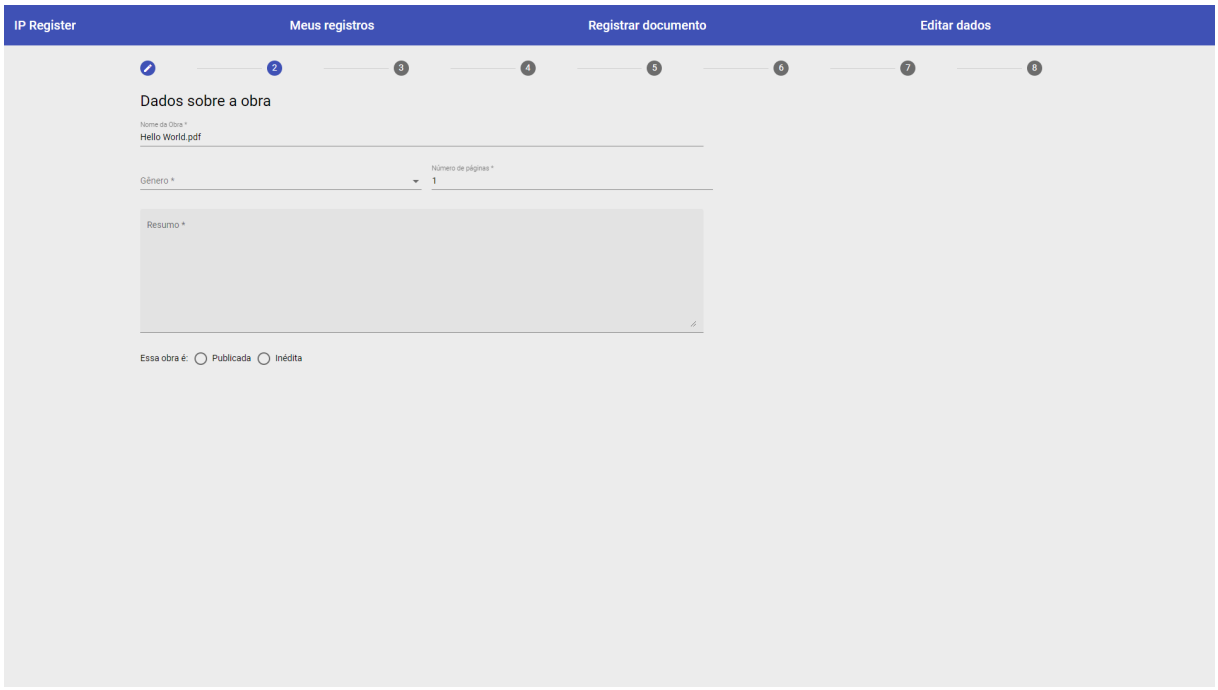


Figura 4.9: Formulário de Dados da Obra.

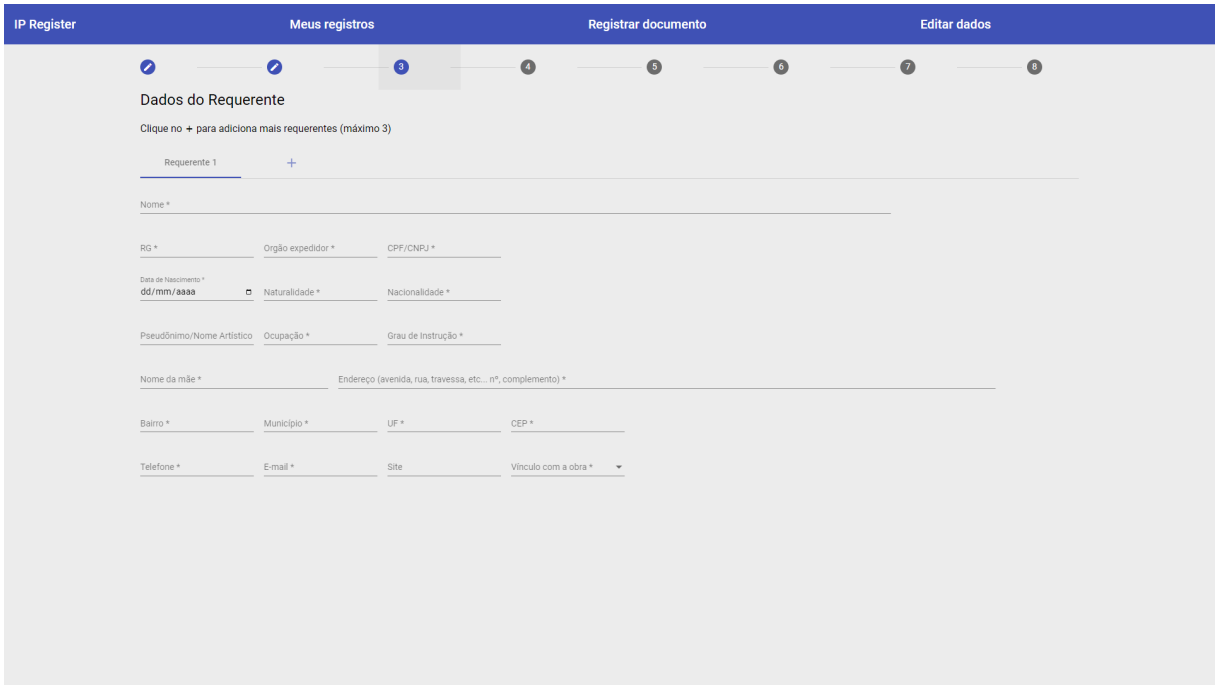


Figura 4.10: Formulário de Requerentes.

campo do Requerente 1, mas ainda é possível edição caso necessário. Os campos de site e pseudônimo são optativos e o campo vinculo da obra é um *select* que apresenta as mesmas opções da Biblioteca Nacional.

O Quarto e o Quinto passo possui um comportamento semelhante, ambos possuem um botão *toggle* que, ao ativar, desabilita os formulários tornando desnecessário o preenchimento deles. No Quarto passo é pedido os dados do Representante Legal, caso o requerente seja maior de idade, é só ativar o *toggle* de "Sou maior de 18 anos" que o requerente declara ser maior de idade. No Quinto passo, que pede os dados de se a obra é uma Tradução ou Adaptação de uma obra já existente, ao apertar o *toggle* "A obra que estou registrando não é uma adaptação ou tradução de outra obra", ele declara que sua obra é original e esse passo não será mais necessário ser preenchido;

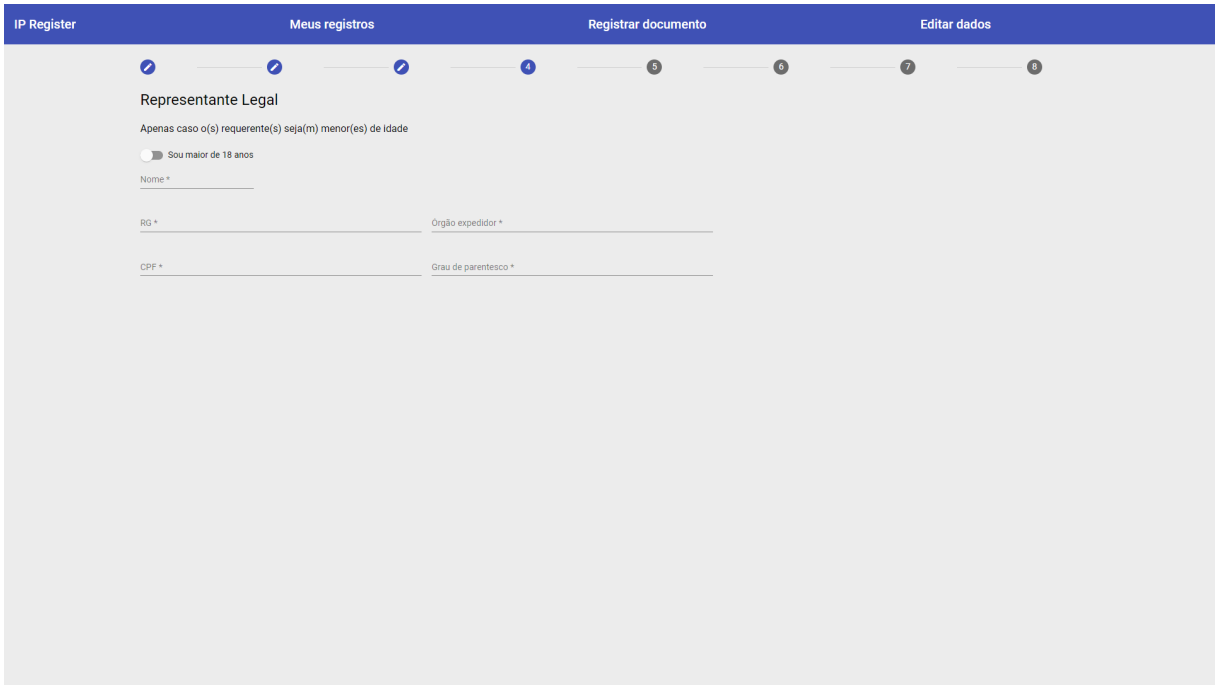


Figura 4.11: Formulário de Dados do Representante Legal.

Caso o requerente seja menor de idade, é necessário cadastrar os dados do seu Representante Legal informando: nome, RG, órgão expedidor, CPF e grau de parentesco. Caso a obra seja Tradução e/ou Adaptação de outra obra, é necessário declarara a qual das duas categorias ela pertence, necessitando que pelo menos um dos *checkboxes* associados seja marcado, sendo possível marcar ambos, e que o nome e o autor da obra original seja informado.

O sexto passo consiste em uma *textarea* para Observações e o sétimo passo são duas *checkboxes* com as Disposições Finais, contendo as mesmas declarações que o formulário da Biblioteca Nacional. Ao concordar com os termos, o requerente declara que a obra é de sua autoria e mantém apenas para si quaisquer responsabilidades judiciais por ela, isentando a plataforma das mesmas. Ao serem aceitos, caso o formulário seja validado, a função *setInMetadata* chama o método *sendMetadata* que realiza o envio dos metadados para o IPFS e salvando o CID retor-

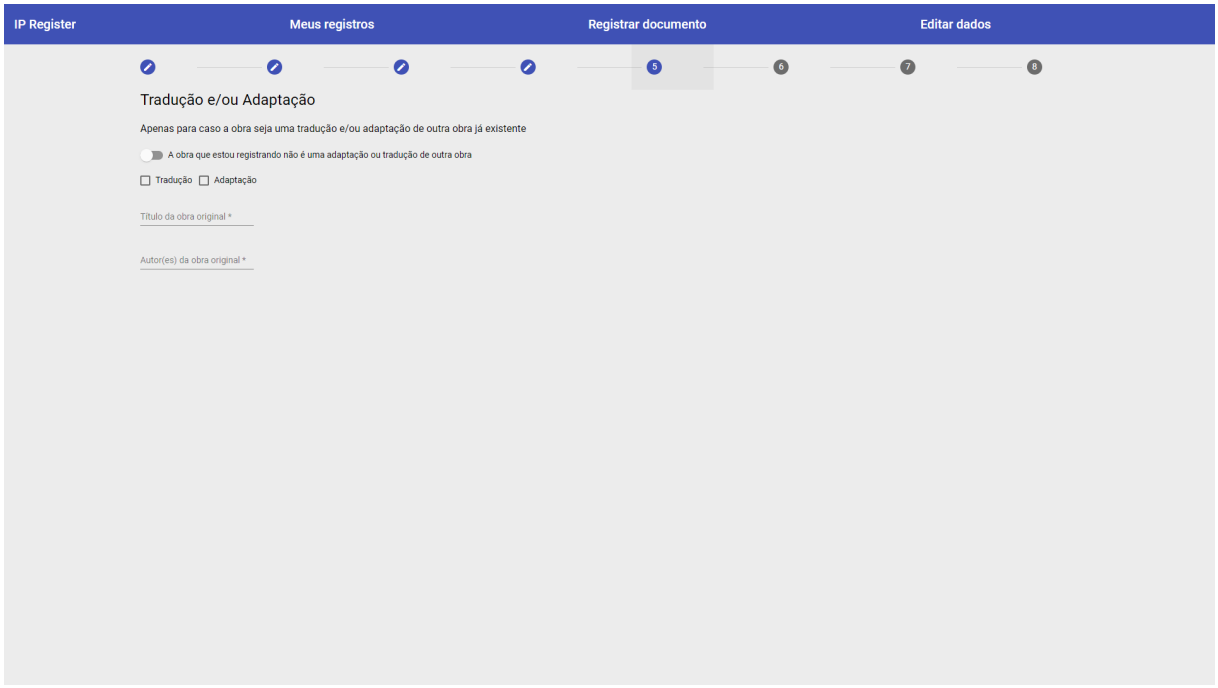


Figura 4.12: Formulário de Dados da obra original.

nado no banco.

No passo final, a plataforma requisita para a carteira conectada uma transação e o usuário deve prestar atenção quando o *pop-up* da extensão aparecer ou, caso esteja conectado pelo celular, abrir o aplicativo e confirmar por la a transação. Ao ser confirmada, um objeto de transação é gerado recebendo como dados o endereço da carteira conectada de remetente, o endereço do contrato correspondente a rede como destinatário, um valor mínimo de *gas*, um *nonce* equivalente ao *id* do próximo NFT a ser gerado e um *data* que recebe da ABI do contrato o acesso ao método *mintNFT*.

Após enviada, a transação retorna uma *hash* para armazenar no banco e exibir na tela, em caso de sucesso, na forma de um link para a página do *scan* correspondente a rede. Ao acessar o link, é possível acompanhar a situação para saber se já foi concluída para enfim poder exibi-lo na tela de Meus Registros (4.2.5). Em caso de falha, é indicado pela tela que ocorreu um erro, sendo necessário reiniciar o processo. Uma vez finalizado o processo, o NFT já pode ser visualizado na tela de Meus registros e adicionada a carteira *mobile* do usuário. Também é possível selecionar a opção de "Realizar outro registro" para reiniciar o processo.

4.3 Integração com a Web3

Para atingir os objetivos propostos no capítulo 1, deve haver uma comunicação entre a aplicação, que roda em um navegador ainda seguindo os protocolos da Web2, e os sistemas descentralizados da rede *Ethereum*, que já operam com as ideias da chamada Web3. Para isso foram utilizadas as bibliotecas de *Javascript* Web3.js e um sistema chamado *Alchemy*, para auxiliarem

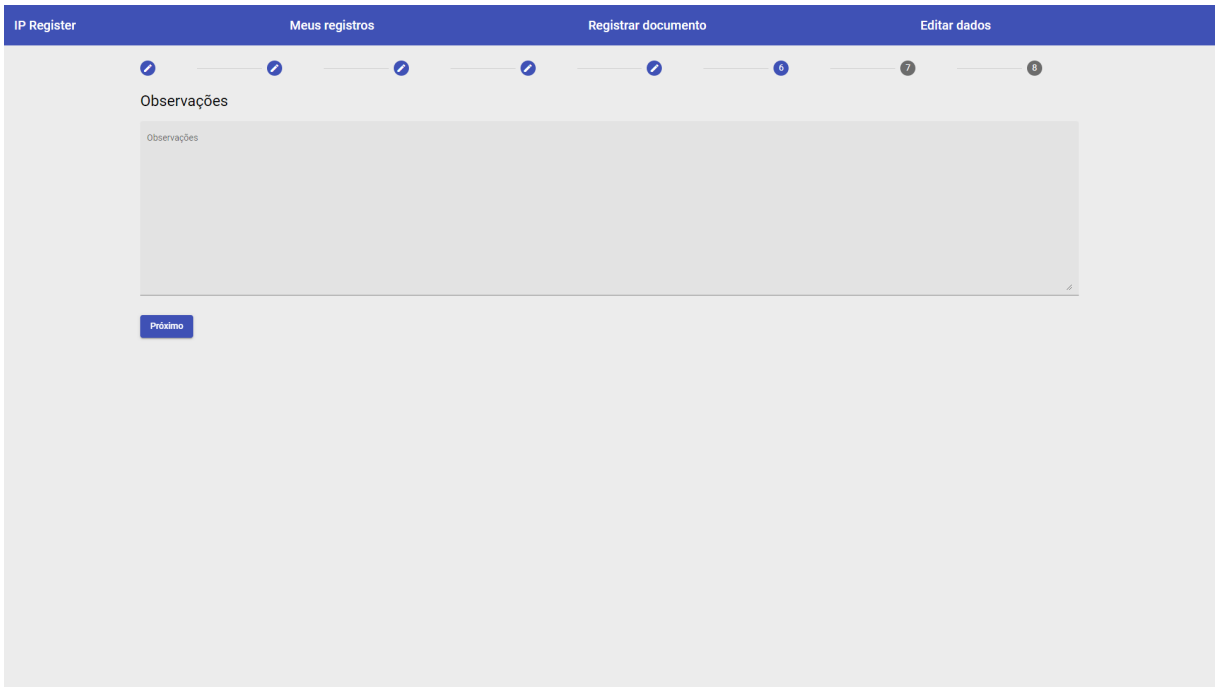


Figura 4.13: Formulário de Dados da Obra.

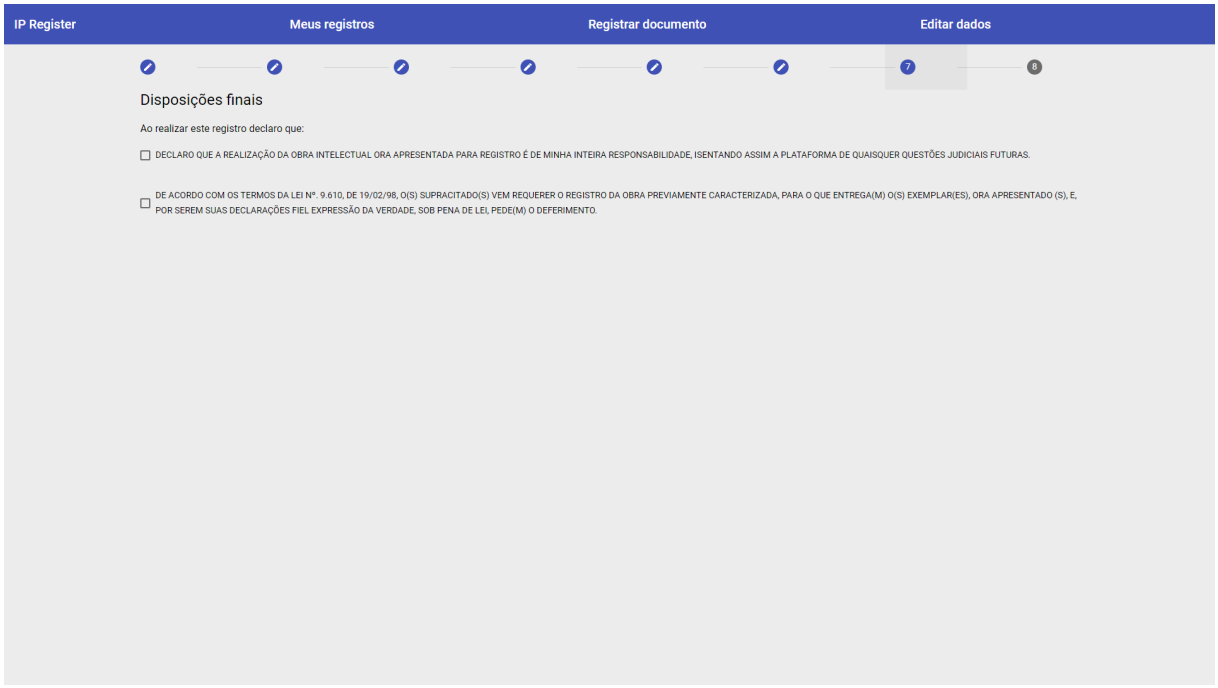


Figura 4.14: Formulário de Dados da Obra.

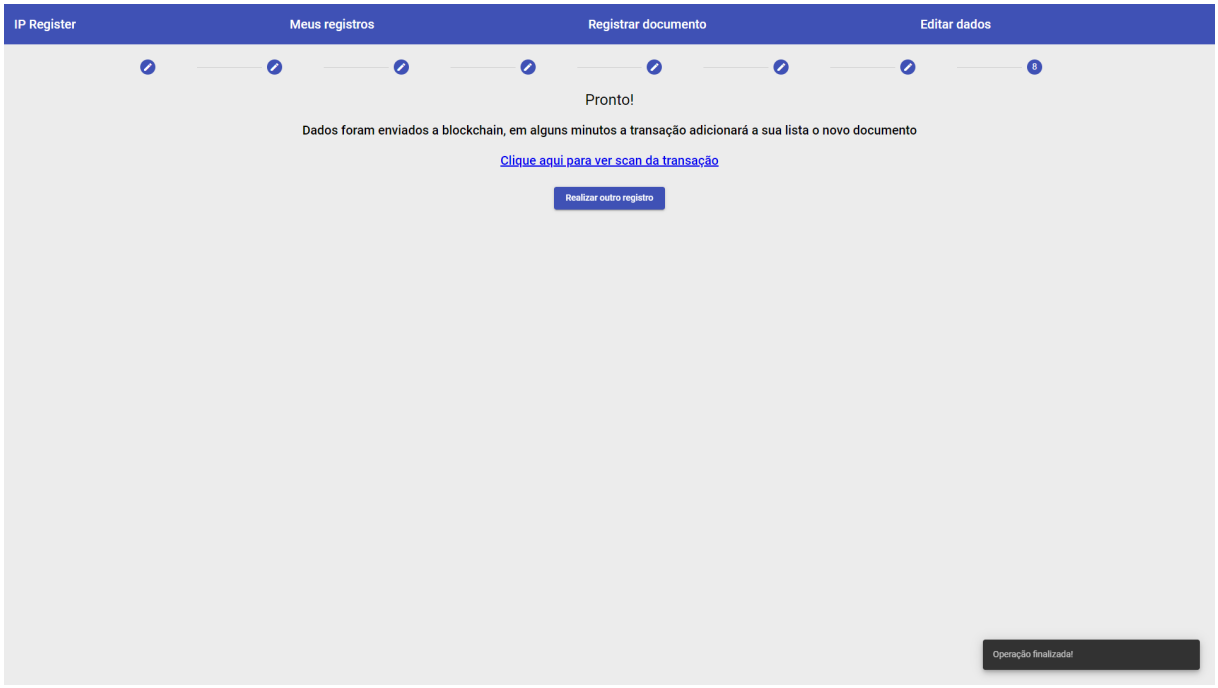


Figura 4.15: Tela de sucesso ao registrar NFT.

na comunicação necessária com as blockchains testadas no projeto.

4.3.1 Alchemy

É uma plataforma web3 focada em auxiliar o desenvolvimento voltado para *blockchain*, para deixá-lo mais fácil para os desenvolvedores. Ele atua como um *Node provider*, onde ao invés de a aplicação ter que se preocupar em ter um nó na rede, ele fornece as infraestruturas necessárias para manter as comunicações com a rede, tirando essa preocupação do desenvolvedor, que passa a se focar apenas em codificar o projeto.

Para utilizar, basta ir no site deles, criar uma conta gratuitamente, gerar as chaves de API, montar um projeto e baixar a biblioteca *@alch/alchemy-web3*. Para iniciar a conexão, basta importar o comando *createAlchemyWeb3* e passar como argumento a URL de acesso para o projeto previamente criado e salvando seu retorno em uma variável *web3*. Após isso, é só começar a utilizar as funções disponíveis na documentação do Web3.js.

4.4 Integração com carteiras

Para realizar a interação entre a aplicação web e o contrato inteligente, é necessário utilizar conexão de uma carteira de criptomoedas com suporte as redes onde o contrato teve seu *deploy* feito. Duas formas foram abordadas no desenvolvimento deste projeto. Uma foi a de conectar a uma extensão da *Metamask*, que pode ser instalada em qualquer navegador compatível, e a outra foi por carteiras mobile utilizando uma ferramenta chamada *Wallet Connect*.

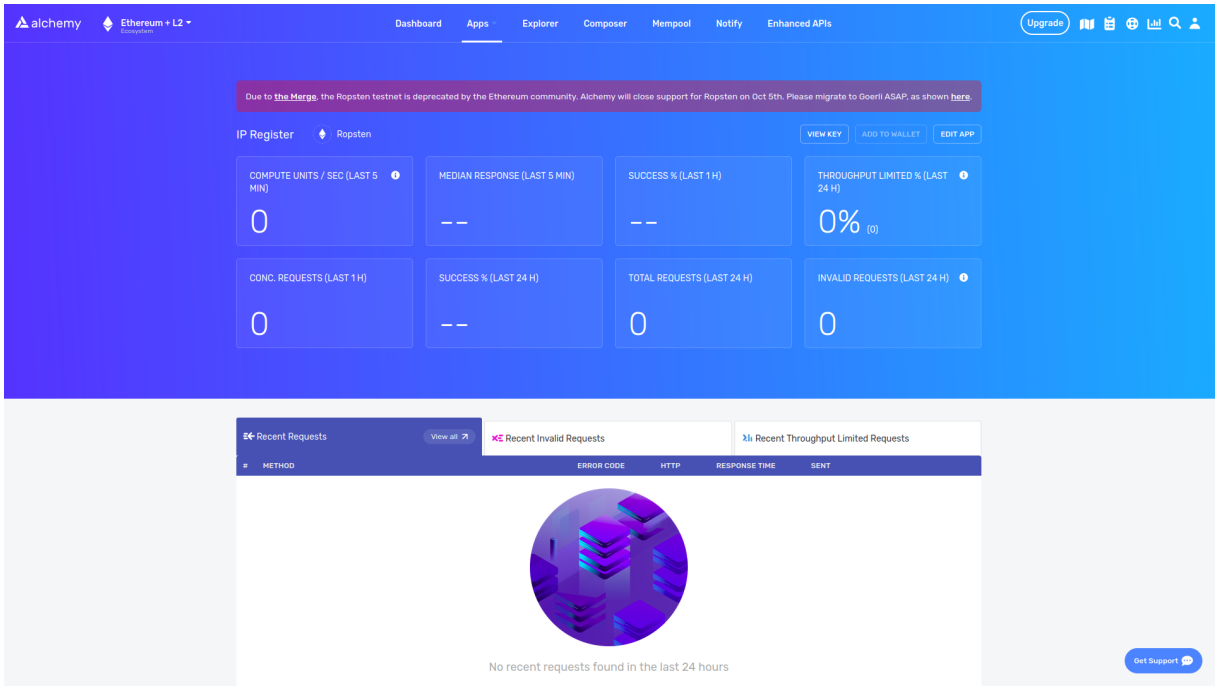


Figura 4.16: Dashboard da Alchemy para o projeto

4.4.1 Extensão da Metamask

A carteira *Metamask* pode ser facilmente utilizada a partir de extensões para navegadores que podem ser encontradas nas lojas de extensões para seu navegador. Uma vez que ela seja adicionada, ela adiciona a chave *ethereum* ao objeto de *window* do *Javascript*, a qual permite que uma aplicação web consiga acessar a carteira e requisitar métodos para ela.

4.4.2 Wallet Connect

Segundo a sua documentação, é descrito como um protocolo aberto para realizar comunicações seguras entre carteiras de criptomoedas e aplicações Web3 que queiram se conectar a elas. A conexão é feita de forma remota utilizando um "servidor ponte" para a transmissão de dados encriptados simetricamente por uma chave compartilhada entre os dois dispositivos da conexão (*payloads*).

A conexão é feita através de um código QR exibido pela aplicação assim que a opção "Conectar *Wallet Connect*" for clicada na tela inicial, o usuário então deve abrir sua carteira *mobile* e escanear com ela o código para realizar a conexão. Caso a carteira seja suportada pelo *Wallet Connect*, aparecerá uma tela pedindo a confirmação do usuário para realizar a conexão da carteira com a aplicação. Se for aceita, o programa realiza a conexão e com isso será possível utilizar a aplicação.

4.4.3 WalletService

É o principal serviço para a comunicação entre as carteiras e a aplicação. Serve como um adaptador em que ele recebe em seu construtor os serviços *WalletConnectService* e *MetamaskService* e, a depender da forma utilizada na conexão, ele escolhe qual deles utilizar.

A decisão de criar esse serviço foi a de abstrair melhor o uso dos serviços de carteiras no código utilizando um serviço apenas com as funções de enviar transações, conectar e desconectar carteira e por trás, o serviço decide qual a função exata que ele utilizar, sendo a que utiliza a extensão ou a *Wallet-connect*. As implementações das funções operam de formas distintas mas possuem o mesmo resultado, permitindo assim que esse serviço conseguisse atuar de forma mais "genérica" durante a execução do programa.

Os métodos presentes nesse serviço são:

connect

- Recebe uma *string* correspondente a forma de conexão. valores aceitos são:
  - *Metamask* - para conectar com a extensão;
  - *WalletConnect* - para conectar pelo celular.
- Atribui a variável *mainService* qual dos serviços deve ser utilizado quando os métodos forem chamados e chama o método de *connect* correspondente;
- Não possui retorno.

sendTransation

- Recebe uma *string* correspondente ao *contenId* dos metadados no IPFS;
- Chama o método de *sendTransation* correspondente ao serviço atribuído no *mainService*;
- Retorna uma *Promise*.

disconnect

- Não possui parâmetros;
- Chama o método de *disconnect* correspondente ao serviço atribuído no *mainService*;
- Não possui retorno.

4.4.4 MetamaskService

Cuida das operações utilizando a extensão *Metamask* instalada no navegador do usuário. Seus métodos são invocados pelo *WalletService* quando ele está conectado por este método. Ele se utiliza da propriedade *Ethereum* no objeto *window* do *Javascript*, adicionado quando o navegador possui uma extensão de carteira *Ethereum*, para requisitar métodos Web3.

Os métodos presentes nesse serviço são:

connect

- Não possui parâmetros;
- O método recupera a propriedade *ethereum* do objeto *window* e verifica se pertence a uma extensão da Metamask, se não for ele dispara um aviso de "instale a Metamask", caso seja ele detecta um *provider* da *Ethereum* e faz uma requisição para o método *eth_requestAccounts*, o qual irá disparar a extensão da Metamask no navegador e aguardará o usuário se conectar;
- Caso o usuário se conecte a uma conta, a requisição retornará uma lista de contas e o primeiro item dela será utilizando como sendo a conta conectada. caso usuário não se conecte ele retorna um erro;
- O método *authUser* valida se o usuário já está cadastrado, caso esteja os dados de requisição são salvos em *localStorage*, a página é recarregada e o id da rede é requisitado pela requisição ao método *eth_chainId*. caso não esteja será redirecionado a tela de cadastro e conexão abortada;
- O valor do *id* da rede vem em hexadecimal então é convertido para decimal para manter padronização dos ids e salvo em *localStorage*;
- Não possui retorno.

setEvents

- Não possui parâmetros;
- Serve para programar o que o código deve fazer ao detectar alguns eventos da carteira;
- *accountsChanged* - evento que detecta que ele mudou a conta conectada, caso ela retorne uma lista vazia, significa que ele desconectou, então a função *disconnect* é chamada;
- *chainChanged* - evento que detecta que a rede foi alterada, retorna o novo id da rede e ele é convertido em decimal e depois armazenado em *localStorage* e substituído na variável global;

- Não possui retorno.

sendTransation

- Recebe uma *string* correspondente ao *contenId* dos metadados no IPFS;
- Monta um objeto de transação e o envia utilizando o método *eth_sendTransaction*;
- Retorna uma *Promise*.

disconnect

- Não possui parâmetros;
- Remove do *localStorage* todas variáveis necessárias para manter a conexão e recarrega a página;
- Não possui retorno.

4.4.5 WalletConnectService

Cuida das operações utilizando o *Wallet Connect* para acessar carteiras no celular. Seus métodos são invocados pelo *WalletService* quando ele está conectado por este método. Ele se utiliza de um objeto *connector* gerado pelo *WalletConnect* que possui métodos que facilitam a manipulação dos eventos.

Os métodos presentes nesse serviço são:

connect

- Não possui parâmetros;
- Verifica se já possui uma conexão, caso haja recupera a conta, caso não chama o método *createSession* para iniciar uma sessão abrindo um QR *code*;
- Configura um evento de conexão onde o valor da conta é recuperado, utilizando a primeira da lista, e enviado para o método *authUser* que valida se o usuário já está cadastrado.
- Caso o usuário esteja cadastrado, os dados de conexão são salvos em *localStorage* e a página é recarregada;
- Caso o usuário não esteja cadastrado, a conexão é desfeita com método *killSession*, e o usuário é redirecionado a tela de cadastro;
- Não possui retorno.

setEvents

- Não possui parâmetros;
- Serve para programar o que o código deve fazer ao detectar alguns eventos da carteira;
- session_update - evento que detecta mudanças na sessão, utilizada para atualizar o *id* da rede, recupera do objeto de *payload* esse valore e reatribui e salva em *localStorage*;
- *disconnect* - evento que detecta desconexão, chama a função *disconnect*;
- Não possui retorno.

sendTransation

- Recebe uma *string* correspondente ao *contenId* dos metadados no IPFS;
- Monta um objeto de transação e o envia utilizando o método *sendTransaction* do *connector*;
- Retorna uma *Promise*.

disconnect

- Não possui parâmetros;
- Utiliza o método *killSection* do *connector* para encerra a conexão entre o aparelho e a plataforma e remove do *localStorage* todas variáveis necessárias para manter a conexão e recarrega a página;
- Não possui retorno.

4.5 Comunicação com IPFS

Como foi proposta, a comunicação é feita utilizando a API do *Piñata*. Para facilitar seu uso em código, foi criado um serviço chamado *IpfsPinataService* para conter as funções necessárias, que serão abordadas mais a frente, e realizar as configurações necessárias para o funcionamento das requisições.

4.5.1 IpfsPinataService

O serviço contém os atributos *baseUrl*, uma *string* que armazena a base da URL de *endpoint* da API, *axios*, uma biblioteca JS que realiza as requisições para a API, e *headers*, sendo um

cabeçalho para as chamadas que precisa de valores para autorizar a operação cujo valor é configurado no construtor.

O *Piñata* trabalha com duas formas de autenticação, uma utilizando *Bearer token*, com um JWT gerado para a conta, e a outra com cabeçalho customizável que usa as chaves da API e a chave secreta geradas pela conta. Ambas as formas foram testadas e funcionaram normalmente, então foi escolhido seguir, para esse trabalho, com o *Bearer Token*. O *token* utilizado foi salvo como uma variável de ambiente do programa.

O serviço possui dois métodos básicos: *sendFile*, para enviar arquivos, e *sendJSON*, para enviar JSONs. também há um método de *testConnection* para realizar testes de conexão para segurar que a comunicação com o gateway está ocorrendo normalmente.

sendFile

- Parâmetros: objeto do tipo File contendo o documento a ser enviado;
- Utiliza uma requisição POST para o *endpoint* 'pinning/pinFileToIPFS';
- Retorna uma *Promise*;

sendJSON

- Parâmetros: arquivo JSON contendo os metadados;
- Utiliza uma requisição POST para o *endpoint* 'pinning/pinJSONToIPFS';
- Retorna uma *Promise*;

4.6 Testes com outras carteiras

Durante o desenvolvimento, a carteira escolhida como foco foi a Metamask, porém, graças ao *Wallet Connect* que possui integrações com diversas carteiras mobile, foi possível realizar testes em carteiras distintas. Infelizmente, nenhum das carteiras disponíveis possuía conexão com as redes de testes, apenas com as redes principais da Ethereum ou *Binance*.

Os testes foram realizados utilizando a rede Ethereum nas seguintes carteiras mobile: *Rainbow* (Figura 4.17), *Trust* (Figura 4.18), *WallETH* (Figura 4.19), *Argent* (Figura 4.20), *Crypto.com* (Figura 4.22) e *imToken* (Figura 4.23). Em todas o mesmo roteiro foi seguido:

- Procurar o *scan* de código QR no aplicativo;
- Conectar com a aplicação pelo *Wallet Connect*;
- Realizar o registro de um documento;

- Verificar se no passo final a carteira exibe a transação;

O roteiro de testes não chega a concluir a transação devido à falta de fundos nas carteiras, pois seria necessário investir dinheiro real para obter ETH ou minerar, o que não foi considerado interessante para esse caso. A maioria conseguiu chegar até o último passo, com exceção apenas da carteira *Argent*, que, durante seu teste de transação, foi disparado no *console* da aplicação um erro, que está registrado na Figura 4.21. Não foi descoberto se foi devido à falta de fundos ou se foi algum problema com a conexão com o aplicativo.

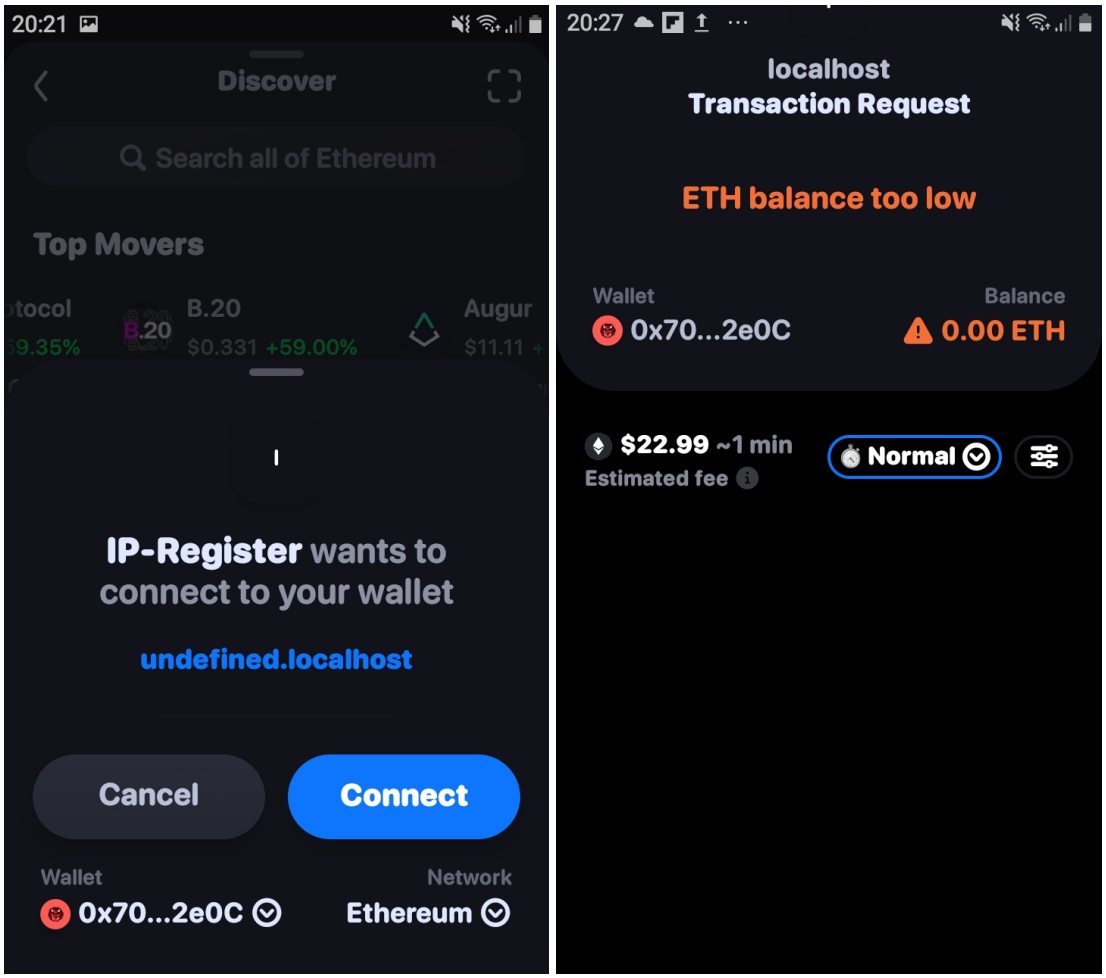


Figura 4.17: Conexão (esquerda) e transação (direita) na carteira Rainbow.

Outra carteira utilizada foi a *Coinbase*, mas não via *Wallet Connect*, pois o mesmo não oferece suporte para ela. O teste foi realizado via extensão, porém ainda foi necessária a integração com aplicativo para confirmar a conexão e as transações. A conexão se deu de forma "acidental" ao iniciar uma conexão pela opção "Conectar com Metamask" e ambas '*Coinbase* e *Metamask* abriram. É possível visualizar isso na Figura 4.24. Após fechar a janela da Metamask e seguir pela janela da Coinbase, foi possível conectar a aplicação.

A Coinbase possui suporte a rede de teste *Ropsten*, então um valor em ETH foi transferido-lhe através de um *Faucet* e o roteiro de registro de NFT foi posto em prática. Ao chegar no passo

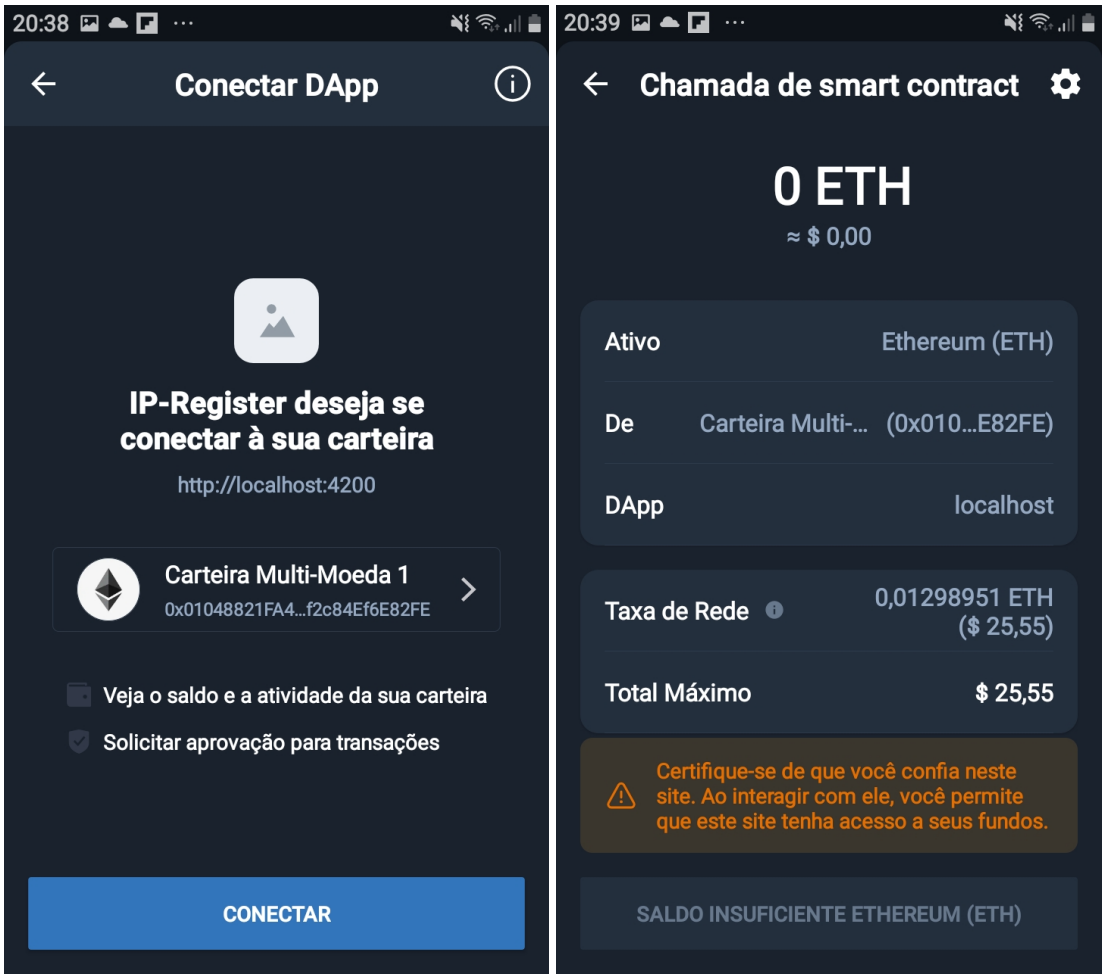


Figura 4.18: Conexão (esquerda) e transação (direita) na carteira Trust.

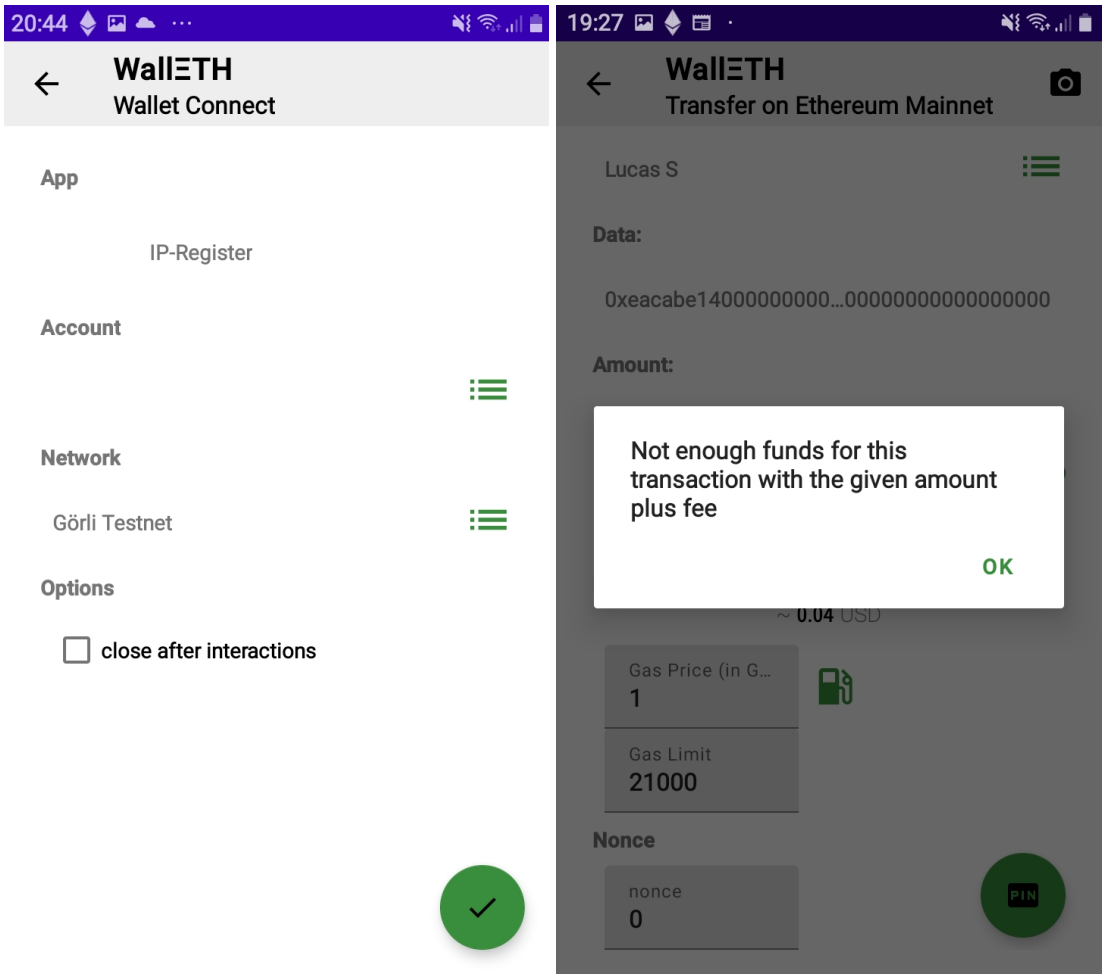


Figura 4.19: Conexão (esquerda) e transação (direita) na carteira WallETH.

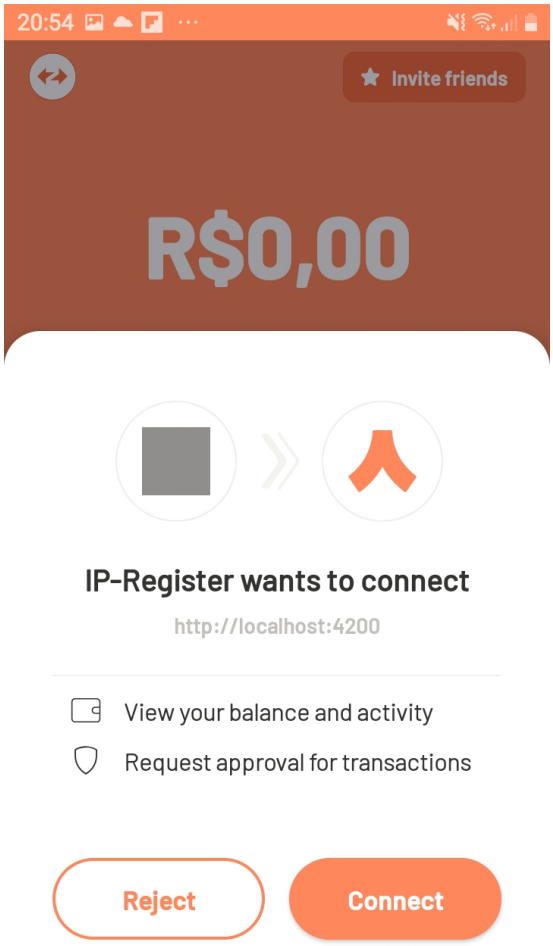


Figura 4.20: Conexão na carteira Argent.

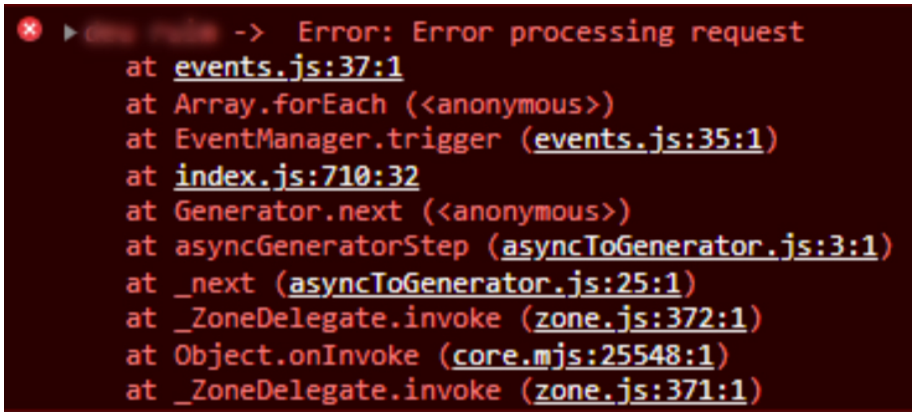


Figura 4.21: Erro recebido na transação com a carteira Argent.

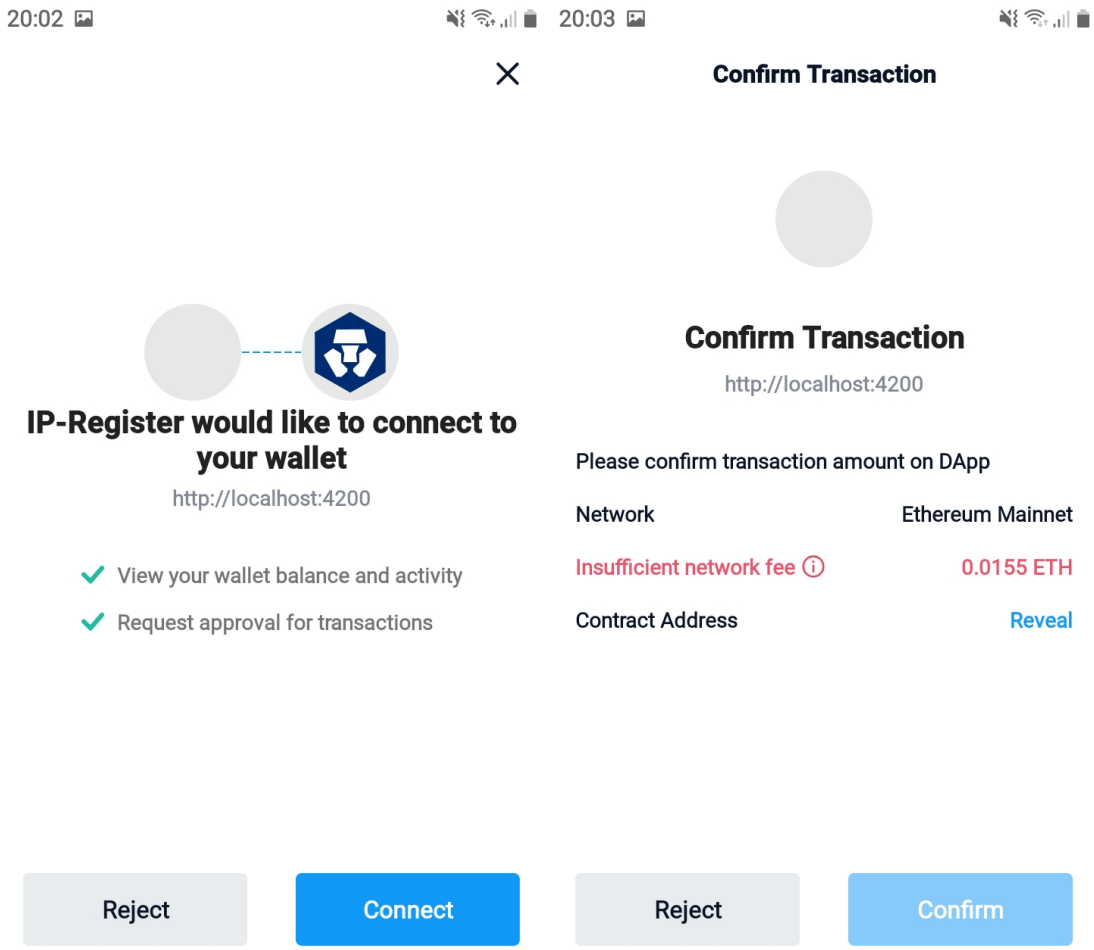


Figura 4.22: Conexão (esquerda) e transação (direita) na carteira Crypto.com.

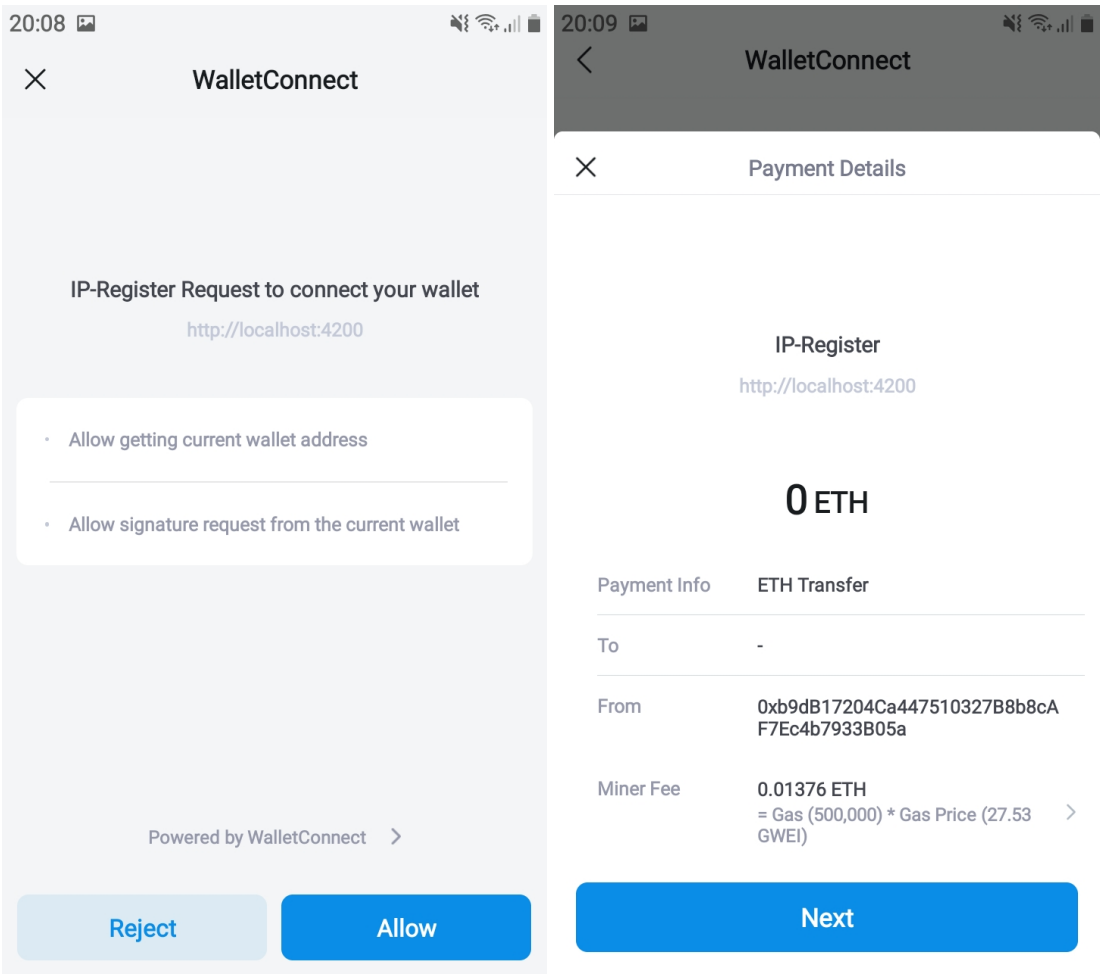


Figura 4.23: Conexão (esquerda) e transação (direita) na carteira imToken.

final, a tela de confirmar transação apareceu no aplicativo e assim foi possível concluir o registro de uma NFT na *blockchain* pela Coinbase. A Figura 4.25 mostra a conexão e a transação na visão do aplicativo, enquanto a Figura 4.26 mostra a transação realizada no *scan* da *Ropsten*.

Apesar do sucesso ao realizar o *mint* do NFT com a Coinbase, com a extensão ativada foi percebido que houve algum tipo de interferência com a extensão da *Metamask*. Por exemplo, o evento de desconexão que recarrega a página quando detecta que a carteira foi desconectada da aplicação foi perdido. Ao desativar a extensão da *Coinbase* os eventos voltaram a funcionar normalmente. Quando a extensão da *Metamask* foi desativada, a opção nem abria a tela de conectar a *Coinbase*, pois o código detecta a existência de uma carteira *Metamask* no navegador para poder ativar seus eventos.

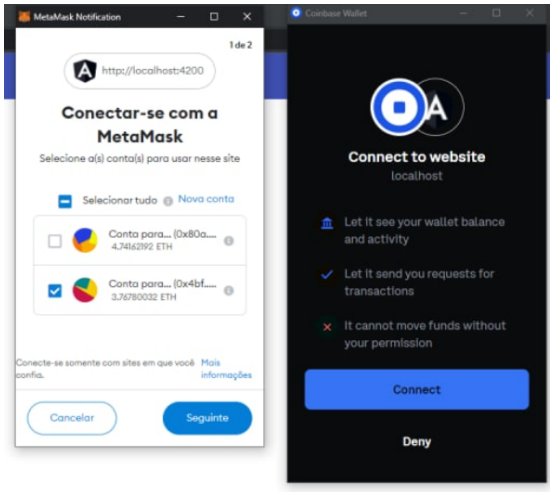


Figura 4.24: Conexão na carteira *Coinbase Wallet* pelo navegador.

4.7 Binance Testnet

O *deploy* para a Rede de Teste da *Binance* foi feito utilizando a *Remix IDE* e o endereço do contrato foi adicionado a aplicação. Para se adequar ao novo paradigma de contratos em outras redes, foram feitas alterações para recuperar o *id* da rede atualmente conectada na carteira, codificar eventos para detectar a mudança de rede e armazenar o valor novo dos *id* e um código de *switch* case para retornar o endereço do contrato correspondente ao *id* recebido pela chamada da função *getContract*.

Na conexão com *Metamask* o retorno possui apenas uma lista de contas conectadas, é necessária uma requisição com o método *eth_chainId* que retorna o valor do *id* correspondente a rede em hexadecimal. Na *Wallet Connect*, todos os eventos possuem um objeto *payload* que possui uma chave *params* que é uma lista cujo primeiro item já possui a chave *chainId*, porém o valor dela está em decimal. Para fins de padronização, foi optado pelo uso dos valores em decimais para os códigos que precisem do id da rede, então foi utilizada uma função de conversão do

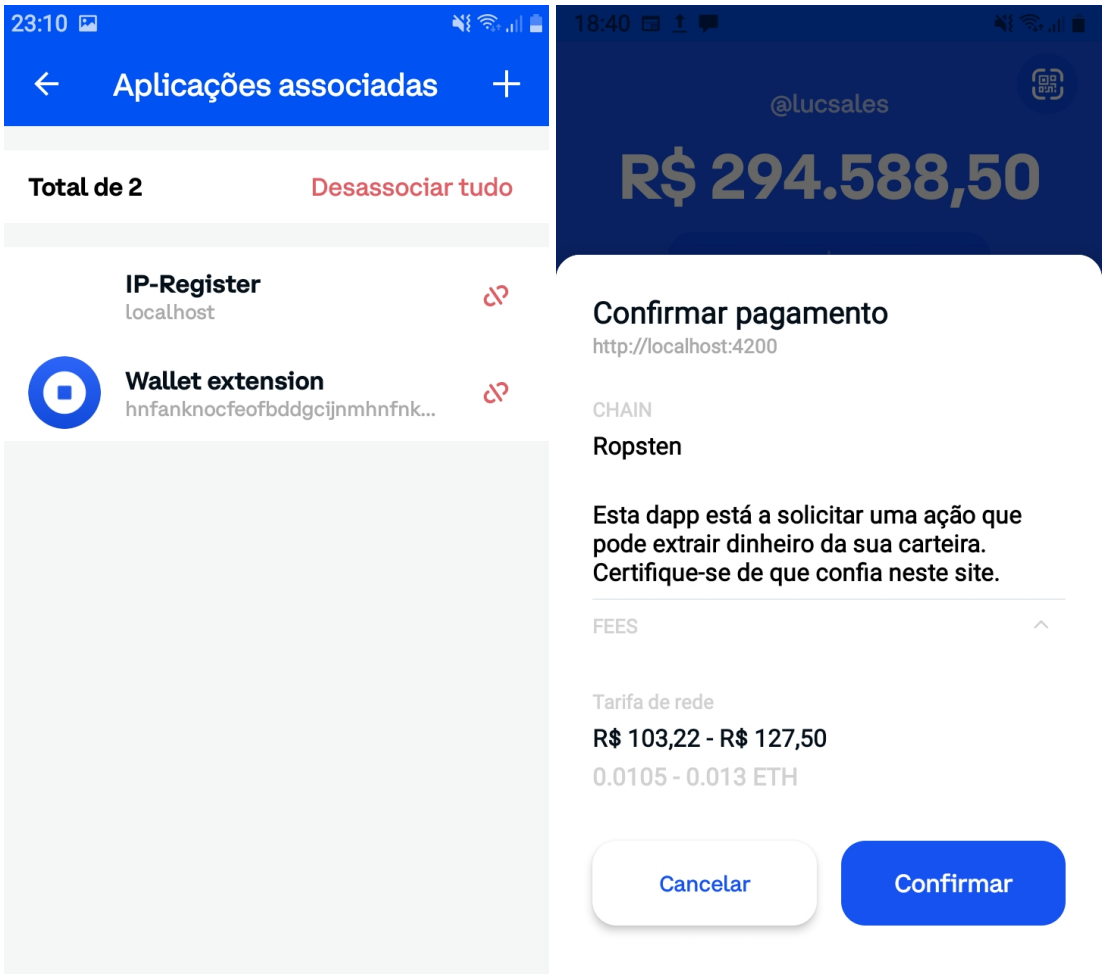


Figura 4.25: Conexão (esquerda) e transação (direita) na carteira *Coinbase Wallet*.

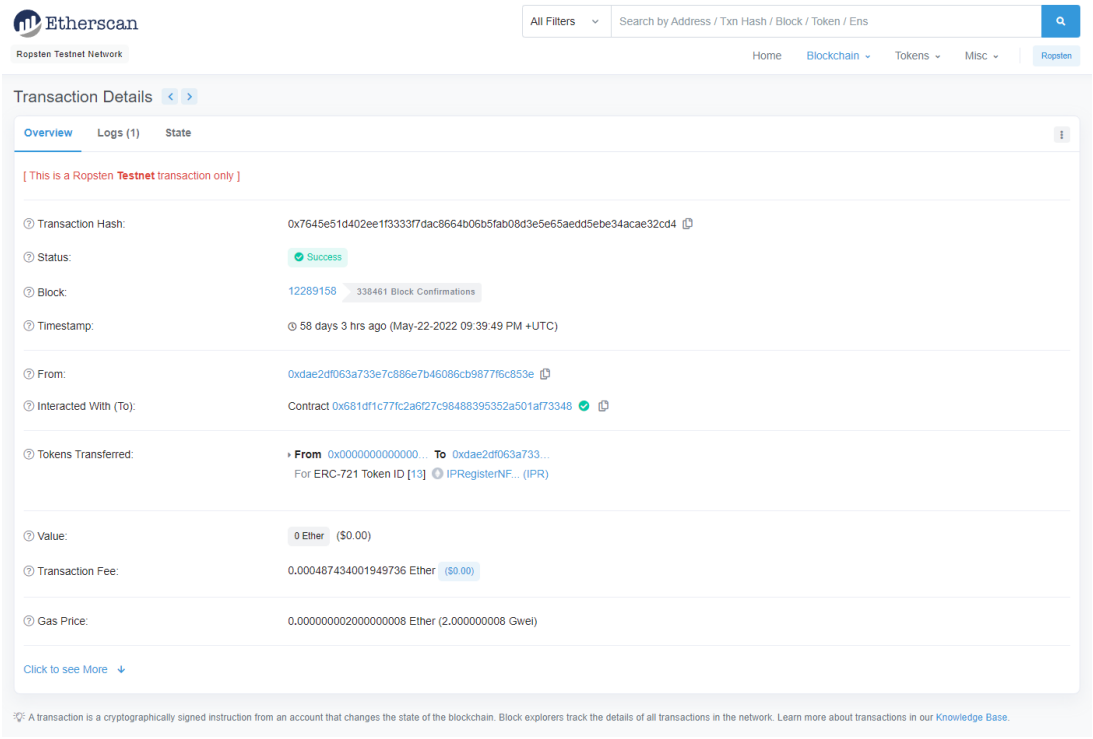


Figura 4.26: *Scan* da transação realizada utilizando a carteira da *Coinbase*.

Web3.utils para converter o *id* retornado pela *Metamask* de hexadecimal para decimal (função *hexToNumber*).

Na hora de testar as requisições para o contrato da *Binance* a carteira *Metamask* no navegador conseguiu sem problemas gerar uma transação e ela foi registrada no *scan* da *Binance Testnet*, entretanto os testes com a *Wallet Connect* não foram muito bem sucedidos.

Conectando pela *Wallet Connect*, as requisições eram geradas, um *hash* de transação era feito, porém ao acessar no *scan* a transação não era encontrada na rede e nem constava no histórico do contrato. A rede *Ropsten* também foi verificada para o caso da transação ter ido para o contrato errado mas nada foi encontrado.

Após algumas pesquisas, foi descoberto que a implementação da *Wallet Connect*, roda apenas em redes da Ethereum e é necessário realizar algumas configurações para que opera com a *Binance*. Infelizmente o tutorial que demonstra métodos para realizar isso era de difícil compreensão e não foi possível implementar a solução para essa conexão.

5

Conclusão

Foi desenvolvido um Contrato Inteligente capaz de "mintar" um NFT em uma rede de testes da *Ethereum*. O *deploy* foi realizado com sucesso tanto na rede *Ropsten* quanto na *BSC testnet*. O contrato herda funções necessárias para a listagem dos *tokens* registrados pela carteira que solicita e retorna para o usuário seu saldo e os *ids* relacionados.

Foi desenvolvida uma API capaz de realizar CRUD de usuários, carteiras e registros. Fazendo vínculos entre as tabelas utilizando o *id* do usuário como uma chave estrangeira e permite o controle de quem tem acesso à plataforma e dos registros que estão sendo realizados.

Foi desenvolvida uma aplicação web capaz de se conectar com a carteira *Metamask*, tanto *mobile* quanto a extensão, cadastrar usuários, permitir edição de dados pessoais, gerar e visualizar registros de autorias feitos em *blockchain* utilizando NFTs. O processo de registro conta com um *upload* de uma arquivo em formato PDF e reproduz o formulário de registro de autoria utilizado na Biblioteca Nacional. O registro é feito utilizando a carteira *Metamask* do usuário que realiza uma transação para o contrato Inteligente que gera o NFT correspondente ao registro realizado.

Foram realizados testes em outras carteiras e redes e foram bem sucedidos até onde era possível. A *Wallet-Connect* dificultou o uso da rede *Binance* via *mobile* mas a extensão funciona tranquilamente. Somente a carteira da *Argent* não foi possível chegar ao último estágio do roteiro de testes e nenhum NFT foi gerado por elas devido à falta de fundos. Entretanto, os testes feitos pela *Coinbase* demonstraram ser possível utilizar outras carteiras para gerar registros com essa aplicação.

5.1 Trabalhos Futuros

5.1.1 Marketplace interno

Permitir que na plataforma seja possível a venda ou troca dos Registros de propriedade Intelectual feitos entre os usuários. A movimentação desses ativos digitais tem o intuito de gerar lucro para os artistas que desejem vender os direitos de propriedade de suas criações através de contratos inteligentes que possam permitir renda passiva aos criadores originais.

5.1.2 Geração de certificado

Geração de um certificado de propriedade apenas para exibição como forma visual de demonstrar que o usuário que possuir seu acesso é o atual dono do registro.

5.1.3 Plataforma Administrativa

Com o possível crescimento desse projeto, torna-se necessário um método maior de controle da aplicação, dos usuários e de novas *features*. Sendo assim uma plataforma para uso administrativo com uma conta maior que possa consultar os dados armazenados e realizar seu controle torna-se necessário.

Junto a isso, vem melhorias de controle de acesos dentro da API com autenticação baseada em *Roles* de usuário que diferencia um administrador de um usuário padrão na hora de acessar e requisitar para a mesma.

5.1.4 Recuperar registros incompletos

A versão atual da aplicação apenas gera o formulário de metadados e o envia para o IPFS. Não há nenhuma persistência desses dados em bancos locais, pois não era interessante manter esses registros localmente já que seriam salvos na *blockchain*. Entretanto se um usuário começar um processo de registro, o banco já salva o início do registro. caso o usuário não chegue a finalizar o processo, os campos de *metadataCid* e, especialmente, *txHash* ficam vazios, o que já indica que a operação deve ter sido abortada.

Se o usuário inciar um processo de registro e existir alguma linha associada a seu *id* sem o *hash* da transação salvo, deverá ser requisitado os dados associados até o momento que ele parou para ele poder concluir ou, se ele preferir, descartar a operação.

5.1.5 Criptografia de dados sensíveis

Ao realizar o registro na rede, o formulário de metadados exige dados pessoais como RG, CPF, telefone, endereço, entre outros. Os dados são salvos no IPFS uma vez lá estão espalhados pela

rede descentralizada e qualquer que tenha o *contentId* associado a ele pode acessar tranquilamente e ter acesso aos dados fornecidos, já que a rede é pública.

Deve haver alguma forma de criptografia para manter o sigilo de dados que o criador não queira que fiquem abertos para qualquer um ter acesso, seja para sua segurança ou pelo simples anonimato. A criptografia deve ser pensada de uma forma que o dono dos dados possa recuperá-los e, em simultâneo, a plataforma possa ter acesso, caso seja necessário.

5.1.6 Funcionalidade de Averbação

Os formulários que a Biblioteca Nacional fornecem servem tanto para realizar um registro como para requere alterações a algum registro previamente existente. Como na *blockchain* os registros são imutáveis, é necessário buscar formas de "versionar" os registros, caso o criador ou o atual dono deseje realizar alguma alteração. O IPFS fornece suporte a versionamento de arquivos, mas esse projeto decidiu não seguir por essa vertente até o momento pois não era necessária até então.

Referências bibliográficas

Binance Academy. O que é delegated proof of stake? URL https://academy.binance.com/pt/articles/delegated-proof-of-stake-explained.

Aditya Anand. Breaking down: Sha-1 algorithm. URL https://infosecwriteups.com/breaking-down-sha-1-algorithm-c152ed353de2.

BBVA. A basic dictionary of blockchain: 10 terms you should know. URL https://www.bbva.com/en/basic-dictionary-blockchain-10-terms-know/.

Juan Benet. Stanford seminar - ipfs and the permanent web. URL https://www.youtube.com/watch?v=HUVmypx9HGI.

Isabela Cabral. O que são dapps? apps descentralizados podem revolucionar a internet. URL https://www.techtudo.com.br/noticias/2018/08/o-que-sao-dapps-apps-descentralizados-podem-revolucionar-a-internet.ghtml.

Henrique Centieiro. What's proof of elapsed time. URL https://medium.com/nerd-for-tech/whats-proof-of-elapsed-time-4f67cf3f45b3.

Kerry Clarke-Potter. What is a blockchain 'token'? is it just cryptocurrency? URL https://blockheadtechnologies.com/what-is-a-blockchain-token-is-it-just-cryptocurrency/.

Coinbase. O que é defi? URL https://www.coinbase.com/pt/learn/crypto-basics/what-is-defi.

Whiteboard Crypto. What is proof of stake? how it works (animated) + ethereum 2.0 upgrade!, a. URL https://www.youtube.com/watch?v=x83EVUZ_EWo.

Whiteboard Crypto. What is a cryptographic hashing function? (example + purpose), b. URL https://www.youtube.com/watch?v=gTfNtop9vzM.

Whiteboard Crypto. What are smart contracts in crypto? (4 examples + animated), c. URL https://www.youtube.com/watch?v=pyaIppMhuic.

Whiteboard Crypto. Crypto coin vs token (differences + examples), d. URL https://www.youtube.com/watch?v=422HORNUfkU.

Whiteboard Crypto. What is an nft? (non-fungible tokens explained), e. URL https://www.youtube.com/watch?v=4dkl5O9LOKg.

Whiteboard Crypto. What is web 3.0? (explained with animations), f. URL https://www.youtube.com/watch?v=nHhAEkG1y2U.

Portal da Indústria. O que é propriedade intelectual, registro de marca e concessão de patente. URL https://www.portaldaindustria.com.br/industria-de-a-z/propriedade-intelectual-registro-de-marca-e-concessao-de-patente/.

Câmara do deputados. Lei nº 496, de 1º de agosto de 1898. URL https://www2.camara.leg.br/legin/fed/lei/1824-1899/lei-496-1-agosto-1898-540039-publicacaooriginal-39820-pl.html.

IPFS Docs. How ipfs works. URL https://docs.ipfs.tech/concepts/how-ipfs-works/.

Chris Dupres. How ipfs works and why you need to care. URL https://thechiaplot.net/2022/01/08/how-ipfs-works-and-why-you-need-to-care/.

Euromoney. What is blockchain? URL https://www.euromoney.com/learning/blockchain-explained/what-is-blockchain.

Simply Explained. Proof-of-stake (vs proof-of-work), a. URL https://www.youtube.com/watch?v=M3EFi_POhps.

Simply Explained. Ipfs: Interplanetary file storage!, b. URL https://www.youtube.com/watch?v=5Uj6uR3fp-U.

Jake Frankenfield. Proof of elapsed time (poet), a. URL https://www.investopedia.com/terms/p/proof-elapsed-time-cryptocurrency.asp.

Jake Frankenfield. Crypto tokens definition, b. URL https://www.investopedia.com/terms/c/crypto-token.asp.

Jake Frankenfield. Blockchain wallet, c. URL https://www.investopedia.com/terms/b/blockchain-wallet.asp.

Tainá Freitas. O que é defi e como funcionam as finanças descentralizadas? URL https://www.startse.com/artigos/o-que-e-defi-decentralized-finance/.

Geeksforgeeks. Proof of work (pow) consensus. URL https://www.geeksforgeeks.org/proof-of-work-pow-consensus/.

Gary Gensler. 1. introduction for 15.s12 blockchain and money, fall 2018, a. URL https://www.youtube.com/watch?v=EH6vE97qIP4.

Gary Gensler. 2. money, ledgers & bitcoin, b. URL https://www.youtube.com/watch?v=5auv_xrvoJk.

Gary Gensler. 3. blockchain basics & cryptography, c. URL https://www.youtube.com/watch?v=0UvVOMZqpEA.

Vincent Gramoli. Deconstructing blockchains: A comprehensive survey on consensus, membership and structure. URL https://www.researchgate.net/publication/335337656_Deconstructing_Blockchains_A_Comprehensive_Survey_on_Consensus_Membership_and_Structure.

Johnny Harris. Nfts, explained. URL https://www.youtube.com/watch?v=Oz9zw7-_vhM.

Adam Hayes. Blockchain facts: What is it, how it works, and how it can be used. URL https://www.investopedia.com/terms/b/blockchain.asp#ixzz4CYVN0C4n.

IBM. What are smart contracts on blockchain? URL https://www.ibm.com/topics/smart-contracts.

Ayomide Ibosiola. Web3.0 — the shift from information publishing to reasoning. URL https://njkhanh.com/web3-0-the-shift-from-information-publishing-to-reasoning-p5f3137393833.

Web3 is going great. What is web3? URL https://web3isgoinggreat.com/what.

Joshua. Intro to ethereum. URL https://ethereum.org/pt/developers/docs/intro-to-ethereum/.

Leandro Kovacs. Qual a diferença entre token e coin? [criptomoedas]. URL https://tecnoblog.net/responde/qual-a-diferenca-entre-token-e-coin-criptomoedas/.

Mila Laranjeira. O gasto energético do blockchain | nerdologia, a. URL https://www.youtube.com/watch?v=85g3kOiFo7k.

Mila Laranjeira. Da web cringe à web 3.0 | nerdologia tech, b. URL https://www.youtube.com/watch?v=_nxiWws7CpA.

Hongfang Lu, Kun Huang, Mohammadamin Azimi, and Lijun Guo. Blockchain technology in the oil and gas industry: A review of applications, opportunities, challenges, and risks. *IEEE Access*, pages 41426–41433, 2019.

Moxie Marlinspike. My first impressions of web3. URL https://moxie.org/2022/01/07/web3-first-impressions.html.

Kirsty Moreland. What is blockchain?, a. URL https://www.ledger.com/academy/blockchain/what-is-blockchain.

Kirsty Moreland. What is proof-of-work, b. URL https://www.ledger.com/academy/blockchain/what-is-proof-of-work.

Sumi Mudgil. How to write & deploy an nft (part 1/3 of nft tutorial series). URL https://ethereum.org/en/developers/tutorials/how-to-write-and-deploy-an-nft/.

Fundação Biblioteca Nacional. Apresentação, a. URL https://antigo.bn.gov.br/sobre-bn/apresentacao.

Fundação Biblioteca Nacional. Como solicitar o registro de sua obra, b. URL https://www.gov.br/bn/pt-br/servicos/direitos-autorais-1/como-solicitar-o-registro-de-sua-obra.

Fundação Biblioteca Nacional. Como solicitar o registro de sua obra, c. URL https://antigo.bn.gov.br/servicos/direitos-autorais.

Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Decentralized Business Review*, 2008.

Mitie Okabayashi. Smart contracts: entenda o que são e como funcionam. URL https://blog.bitcointrade.com.br/smart-contract/.

Planalto. Lei nº 9.610, de 19 de fevereiro de 1998. URL http://www.planalto.gov.br/ccivil_03/leis/l9610.htm.

Untung Rahardja, M. A Ngadi, Rahmat Budiarto, Qurotul Aini, Marviola Hardini, and Fitra Putri Oganda. Education exchange storage protocol: Transformation into decentralized learning platform. URL https://www.researchgate.net/publication/356755571_Education_Exchange_Storage_Protocol_Transformation_Into_Decentralized_Learning_Platform.

Sebrae. O que são direitos autorais? URL https://www.sebrae.com.br/sites/PortalSebrae/artigos/o-que-sao-direitos-autorais,9acecdbc74834410VgnVCM1000003b74010aRCRD.

Paulo Silveira, Oliver Hager, Roberta Arcoverde, and Maurício Linhares. Nfts- hipsters ponto tech #259. URL https://www.alura.com.br/podcast/hipsterstech-nfts-hipsters-ponto-tech-259-a1039.

Corwin Smith. Ethereum - redes. URL https://ethereum.org/pt-br/developers/docs/networks/.

Cryptopedia Staff. What are proof of stake and delegated proof of stake? URL https://www.gemini.com/cryptopedia/proof-of-stake-delegated-pos-dpos.

zk Capital. Ipfs: A complete analysis of the distributed web. URL https://medium.com/zkcapital/ipfs-the-distributed-web-e21a5496d32d.