UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

ALEF JUAN FERREIRA DA SILVA

WRITESAFE: UMA APLICAÇÃO WEB PARA REGISTRO DE OBRAS AUTORAIS BASEADA EM TECNOLOGIA BLOCKCHAIN

GUARAPUAVA

2025

ALEF JUAN FERREIRA DA SILVA

WRITESAFE: UMA APLICAÇÃO WEB PARA REGISTRO DE OBRAS AUTORAIS BASEADA EM TECNOLOGIA BLOCKCHAIN

WriteSafe: A Web Application for Copyright Registration Based on Blockchain Technology

Trabalho de Conclusão de Curso de Graduação apresentado como requisito para obtenção do título de Tecnólogo em Tecnologia em Sistemas para Internet do Curso Superior em Tecnologia em Sistemas para Internet da Universidade Tecnológica Federal do Paraná.

Orientador: Prof. Roni Fabio Banaszewski

GUARAPUAVA

2025

4.0 Internacional

Esta licença permite compartilhamento, remixe, adaptação e criação a partir do trabalho, mesmo para fins comerciais, desde que sejam atribuídos créditos ao(s) autor(es). Conteúdos elaborados por terceiros, citados e referenciados nesta obra não são cobertos pela licença.

RESUMO

Este trabalho descreve o desenvolvimento e a implementação do **WriteSafe**, uma aplicação web funcional que oferece prova digital de autoria para músicas, letras e poesias. A solução implementada integra a blockchain *Polygon* para o registro de anterioridade, o armazenamento descentralizado *IPFS* via Pinata, e o Cloudinary como sistema de armazenamento intermediário. Essa arquitetura cria uma evidência de autoria imutável, datada e publicamente verificável, respondendo à demanda por meios ágeis e acessíveis de proteção autoral. O fluxo de registro foi implementado com foco na experiência do usuário, abstraindo a complexidade da Web3. O processo inicia-se com o envio da obra e uma simulação de pagamento via cartão de crédito, utilizando o ambiente de testes do Mercado Pago. Após a confirmação, o sistema realiza o upload para o IPFS, obtém o Content Identifier (CID) e o registra em um Smart Contract na rede Polygon através de uma carteira institucional. Como resultado, é gerado um certificado digital em formato PDF, contendo os metadados da obra, os hashes de comprovação e um QR Code que direciona para a transação pública no Polygonscan. O desenvolvimento seguiu a estratégia de Produto Mínimo Viável (MVP), que resultou em uma plataforma funcional com as funcionalidades essenciais para os perfis de **criador** e **administrador**, incluindo um painel administrativo para monitoramento de estatísticas da plataforma.

Palavras-chave: registro autoral; blockchain; ipfs; proteção autoral; segurança digital.

ABSTRACT

This paper describes the development and implementation of **WriteSafe**, a functional web application that provides digital proof of authorship for songs, lyrics, and poetry. The implemented solution integrates the *Polygon* blockchain for prior art registration, decentralized storage on *IPFS* via Pinata, and Cloudinary as an intermediate storage system. This architecture creates immutable, timestamped, and publicly verifiable evidence of authorship, addressing the demand for agile and accessible copyright protection methods. The registration workflow was implemented with a focus on user experience, abstracting the complexity of Web3. The process begins with the submission of the work and a simulated credit card payment, utilizing the Mercado Pago test environment. Upon confirmation, the system uploads the work to IPFS, obtains the Content Identifier (CID), and registers it on a custom Smart Contract on the Polygon network via an institutional wallet. As a result, a digital PDF certificate is generated, containing the work's metadata, evidentiary hashes, and a QR code that links to the public transaction on Polygonscan. The development followed a Minimum Viable Product (MVP) strategy, which resulted in a functional platform featuring the essential functionalities for the **creator** and **administrator** profiles, including an administrative dashboard for monitoring platform statistics.

Keywords: copyright registration; blockchain; ipfs; copyright protection; digital security.

LISTA DE FIGURAS

Figura 1 – Interface da plataforma MyWrites com destaque para os balões de ajuda que orientam o criador durante o registro 17
Figura 2 – Representação dos perfis de usuários da plataforma WriteSafe. 33
Figura 3 – Processo de registro de conteúdo na blockchain 34
Figura 4 – Logotipo do WriteSafe 41
Figura 5 – Tela de criação de conta 44
Figura 6 – Tela de login 44
Figura 7 – Tela de onBoarding 45
Figura 8 – Tela de upload de arquivos (em inglês) 46
Figura 9 – Tela de listagem 47
Figura 10 – Modelagem inicial do banco de dados da aplicação 49
Figura 11 – Modelagem de dados final implementada no MVP 50
Figura 12 – Cabeçalho do WriteSafe 53
Figura 13 – Sobre nós 53
Figura 14 – Contato 54
Figura 15 – Homepage do WriteSafe 58
Figura 16 – Página de criação de conta do WriteSafe 59
Figura 17 – Página de login do WriteSafe 60
Figura 18 – Interface de registro de obra, incluindo campos de pagamento 63
Figura 19 – Carteira da plataforma WriteSafe na MetaMask 68
Figura 20 – Interface da plataforma Pinata, a hospedeira dos arquivos IPFS 71
Figura 21 – Exemplo de registro da transação imutável na rede da Polygon, na Polygonscan 71
Figura 22 – Configuração e log de Webhooks no Mercado Pago, evidenciando o recebimento bem-sucedido (Status 200) das notificações de teste no backend 72
Figura 23 – Interface administrativa do WriteSafe 74
Figura 24 – Listagem de obras registradas no WriteSafe 74

LISTA DE TABELAS

Tabela 1 – Comparação entre plataformas de registro autoral com blockchain . . . 19

LISTA DE ABREVIATURAS E SIGLAS

Siglas

API	Application Programming Interface (Interface de Programação de Aplicações)
BEC	Blockchain Ethereum Classic
BN	Biblioteca Nacional
CID	Content Identifier (Identificador de Conteúdo)
CRUD	Create, Read, Update and Delete (Criar, Ler, Atualizar e Excluir)
ECAD	Escritório Central de Arrecadação e Distribuição
HTML	HyperText Markup Language (Linguagem de Marcação de Hipertexto)
HU	História de Usuário (User Story)
ICP-Brasil	Infraestrutura de Chaves Públicas Brasileira
IPFS	InterPlanetary File System (Sistema Interplanetário de Arquivos)
JSON	JavaScript Object Notation (Notação de Objeto JavaScript)
JS	JavaScript
JWT	JSON Web Token (Token Web JSON)
MVP	Minimum Viable Product (Produto Mínimo Viável)
NFT	Non-Fungible Token (Token Não Fungível)
ORM	Object-Relational Mapping (Mapeamento Objeto-Relacional)
PDF	Portable Document Format (Formato de Documento Portátil)
PK	Primary Key (Chave Primária)
FK	Foreign Key (Chave Estrangeira)
REST	Representational State Transfer (Transferência Representacional de Estado)
RPC	Remote Procedure Call (Chamada Remota de Procedimento)
SHA-256	Secure Hash Algorithm 256 (Algoritmo Seguro de Hash 256 bits)
UI	User Interface (Interface de Usuário)

UX	User Experience (Experiência do Usuário)

SUMÁRIO

1 INTRODUÇÃO . . . 11

1.1 Objetivos . . . 12

1.1.1 Objetivo geral . . . 12

1.1.2 Objetivos específicos . . . 12

1.2 Estrutura do trabalho . . . 13

2 TRABALHOS RELACIONADOS . . . 14

2.1 Authora Digital . . . 14

2.2 InspireIP . . . 15

2.3 MyWrites . . . 16

2.4 Ujo Music . . . 18

2.5 Considerações . . . 19

3 REFERENCIAL TEÓRICO . . . 20

3.1 Processos de Desenvolvimento . . . 20

3.1.1 Scrum . . . 20

3.1.2 Kanban . . . 20

3.1.3 MoSCoW . . . 21

3.2 Ferramentas de Desenvolvimento . . . 21

3.2.1 Git e GitFlow . . . 21

3.2.2 GitHub . . . 21

3.2.3 GitHub Projects . . . 22

3.2.4 Estratégia de Monorepo . . . 22

3.2.5 Docker . . . 22

3.3 Tecnologias do Lado do Cliente . . . 23

3.3.1 Typescript . . . 23

3.3.2 Angular . . . 23

3.3.3 Tailwind . . . 23

3.3.4 Flowbite . . . 23

3.4 Tecnologias do Lado do Servidor . . . 24

3.4.1 Node.js . . . 24

3.4.2 Nest.js . . . 24

3.4.3 Hardhat e Ethers.js . . . 24
3.4.4 REST, JSON e JWT . . . 25
3.4.5 Gateway de Pagamento (Mercado Pago) . . . 25
3.4.6 Webhooks . . . 26
3.4.7 ORM e Prisma . . . 26
3.4.8 MySQL . . . 26
3.4.9 Cloudinary . . . 26
3.5 Fundamentação Conceitual da Plataforma *WriteSafe* . . . 27
3.5.1 Direitos Autorais e Prova de Anterioridade Digital . . . 27
3.5.2 Blockchain: Funcionamento, Imutabilidade e Timestamp . . . 27
3.5.3 IPFS: Armazenamento Descentralizado de Arquivos . . . 28
3.5.4 Hashing e sua Aplicação em Registros Digitais . . . 28
3.5.5 Certificados Digitais e Validade Jurídica . . . 29
4 MATERIAIS E MÉTODOS . . . 30
4.1 Materiais . . . 30
4.1.1 Stack de Tecnologias Web (Web2) . . . 30
4.1.2 Stack de Tecnologias Web3 e Serviços Externos . . . 30
4.1.3 Integração de Pagamentos e Webhooks . . . 31
4.1.4 Ambiente de Desenvolvimento . . . 31
4.2 Métodos . . . 31
5 ANÁLISE DO SISTEMA . . . 32
5.1 Perfis de Usuários . . . 32
5.2 Funcionalidades Essenciais . . . 35
5.2.1 Criação de Conta (HU01) . . . 35
5.2.2 Envio de Obra (HU02) . . . 35
5.2.3 Geração de Hash (HU03) . . . 36
5.2.4 Registro na Blockchain (HU04) . . . 36
5.2.5 Armazenamento no IPFS (HU05) . . . 37
5.2.6 Certificado Digital (HU06) . . . 37
5.2.7 Validação Pública (HU07) . . . 37
5.2.8 Consulta por Validador (HU08) . . . 38
5.2.9 Visualização por Validador (HU09) . . . 38

5.2.10 Painel Administrativo (HU10) 39
5.3 Considerações 39
6 PROJETO DO SISTEMA 40
6.1 Identidade Visual 40
6.1.1 Paleta de Cores 40
6.1.2 Tipografia 40
6.1.3 Logotipo 41
6.2 Planejamento da Sprint 0 e da Sprint 1 41
6.2.1 Preparação de Ambiente (Sprint 0) 42
6.2.2 Criação de Conta e Login (Sprint 1) 42
6.3 Prototipação de Telas 43
6.3.1 Telas da Primeira Sprint 43
6.3.2 Telas Principais das Sprints Futuras 45
6.4 Modelagem de Dados 47
7 DESENVOLVIMENTO DO SISTEMA 51
7.1 Sprint 0: Fundação do Projeto e Estruturação do Ambiente de Desenvolvimento 51
7.2 Sprint 1: UI/UX Inicial e Identidade Visual 52
7.3 Sprint 2: Backend de Autenticação e Usuários (HU01) 54
7.4 Sprint 3: Fluxo de Autenticação Completo e Segurança 56
7.5 Sprint 4: Frontend do Registro de Obra (HU02) 60
7.6 Sprint 5: Integração de Armazenamento Externo (HU03, HU05) 63
7.7 Sprint 6: Configuração e Deploy do Smart Contract (HU04) 65
7.8 Sprint 7: Orquestração do Fluxo de Pagamento e Registro (HU02 a HU07) 68
7.9 Sprint 8: Dashboards, Finalização do MVP(HU06, HU07, HU10) 72
8 CONCLUSÃO 75
REFERÊNCIAS 76
APÊNDICE A FUNCIONALIDADES FUTURAS 80
A.1 Evoluções no Fluxo de Registro e Pagamento 80
A.2 Funcionalidades para o Criador 81
A.3 Funcionalidades para o Validador Público 81

A.4 Funcionalidades para o Administrador 82

1 INTRODUÇÃO

A criação artística, seja na forma de músicas, letras ou poesias, é uma expressão essencial da cultura e identidade humana. No entanto, com a expansão das tecnologias digitais, garantir a autoria e os direitos sobre essas obras se tornou um desafio crescente. A facilidade de compartilhamento e reprodução de conteúdo online aumentou significativamente as vulnerabilidades relacionadas ao plágio e à apropriação indevida de criações artísticas (JUNIOR; NEJAIM; NEJAIM, 2024).

No Brasil, a principal instituição responsável pelo registro de direitos autorais é a Biblioteca Nacional (BN), cujo processo pode ser realizado de forma presencial ou online, via plataforma Gov.br (Governo Federal, 2023). Embora o registro não seja obrigatório para o reconhecimento legal da autoria, ele funciona como uma importante prova de anterioridade, sendo fundamental em casos de disputas jurídicas. O processo exige a apresentação de uma cópia física ou digital da obra. No caso de músicas, recomenda-se a submissão da letra acompanhada da partitura, pois isso garante uma representação mais completa da composição.

Este processo tradicional apresenta entraves como custo elevado, exigência técnica (como a submissão de partituras) e prazos longos para emissão do certificado, que pode levar de 30 a 90 dias (Governo Federal, 2023). Essas barreiras são especialmente sentidas por artistas independentes, que muitas vezes não dominam a linguagem musical formal ou não têm acesso a profissionais que realizem a transcrição da melodia. Além disso, o valor da taxa (R$ 40,00 por submissão em 2025) pode ser um obstáculo para aqueles que registram obras com frequência (Fundação Biblioteca Nacional, 2025).

Como alternativa, muitos autores recorrem a métodos informais na tentativa de proteger suas criações. Entre elas estão o registro em cartório de títulos e documentos, o envio por e-mail para si mesmo e o envio pelos Correios em envelope lacrado. Apesar de populares, essas práticas apresentam fragilidades significativas. O envelope lacrado, por exemplo, pode ser violado ou ter sua integridade questionada. O registro em cartório fornece apenas uma data de protocolo, sem garantir a originalidade da obra. Já no caso do e-mail, há o risco de a plataforma utilizada ser descontinuada ou perder os dados — como já ocorreu com antigos serviços como o Terra Mail ou Yahoo Grupos — o que inviabilizaria o uso da mensagem como prova. Essas limitações comprometem a efetividade dessas estratégias em contextos legais, especialmente quando confrontadas com disputas formais de autoria (JusBrasil, 2023).

Diante desse cenário, uma solução tecnológica emergente é o uso da blockchain, que permite o registro imutável e auditável de informações, garantindo autenticidade, integridade e marca temporal. Por ser descentralizada e resistente a fraudes, a blockchain possibilita que os próprios autores realizem o registro de suas obras sem intermediações, de maneira segura, rápida e com custo significativamente reduzido em comparação aos métodos tradicionais.

Além disso, o uso combinado de blockchain e IPFS (InterPlanetary File System) permite incluir, no mesmo registro, a letra, a partitura e o arquivo de áudio, oferecendo uma prova téc-

nica de integridade e anterioridade. Embora tais registros ainda não gozem da presunção legal conferida às assinaturas ICP-Brasil pela Medida Provisória n.º 2.200-2/2001, eles podem ser utilizados como meio de prova em juízo e já vêm sendo reconhecidos em decisões judiciais brasileiras e estrangeiras. Estudos setoriais apontam adoção crescente da tecnologia em mais de 170 países, reforçando seu valor probatório complementar (Authora Digital, 2024; PATENTES, 2023).

Neste contexto, este trabalho propõe uma aplicação web que utiliza blockchain e IPFS para gerar prova de anterioridade de obras musicais, oferecendo aos artistas uma alternativa moderna, acessível e juridicamente robusta para comprovar a integridade e a data de criação de suas composições. A solução destina-se sobretudo a músicos independentes e demais criadores que encontram barreiras nos procedimentos tradicionais, contribuindo para democratizar a proteção intelectual no ambiente digital.

1.1 Objetivos

Nesta seção serão apresentados o objetivo geral e os objetivos específicos do presente trabalho.

1.1.1 Objetivo geral

Desenvolver uma aplicação web que permita o registro autoral seguro de músicas, letras e poesias por meio da tecnologia blockchain, garantindo prova de anterioridade imutável e acessível, com foco na proteção dos direitos autorais.

1.1.2 Objetivos específicos

- Permitir o cadastro e autenticação de usuários via sistema tradicional (e-mail e senha), para que cada criador possa acessar um ambiente seguro e personalizado para seus registros.
- Possibilitar o upload de diferentes tipos de obras autorais (letras em texto, áudios em MP3/WAV e partituras em PDF), para abranger diferentes formas de expressão artística e permitir o registro completo da obra.
- Disponibilizar a opção de armazenar os arquivos no IPFS, para garantir o armazenamento distribuído e imutável dos documentos, mantendo a obra acessível mesmo fora da plataforma.

- Realizar o registro do hash da obra na blockchain utilizando a carteira digital da plataforma para garantir a autenticidade, integridade e a marca temporal do registro, com validade jurídica.
- Gerar ao criador um certificado digital de registro, contendo os dados principais da obra (hash IPFS, data de registro, transação na blockchain), para que ele possa comprovar a autoria em disputas ou negociações.
- Fornecer um histórico pessoal de registros realizados para que o criador possa acompanhar, revisar ou recuperar facilmente seus certificados e obras registradas.
- Oferecer uma interface pública de consulta de registros, para que terceiros interessados (como editores, produtores ou juízes) possam verificar a validade de uma obra a partir de seu hash ou título.

1.2 Estrutura do trabalho

Este trabalho está estruturado em **oito** capítulos. O **Capítulo 2** apresenta uma revisão de trabalhos relacionados, analisando soluções existentes que utilizam tecnologias, especialmente blockchain, para o registro de autoria, destacando suas limitações e contribuições. O **Capítulo 3** traz o referencial teórico, com uma introdução às tecnologias e processos que serão aplicados no sistema, priorizados segundo a metodologia MoSCoW. No **Capítulo 4**, são descritos os materiais e métodos utilizados, contextualizando a integração das tecnologias para o desenvolvimento da aplicação.

O **Capítulo 5** aborda a Análise do Sistema, detalhando os perfis de usuários, as funcionalidades essenciais, e considerações que fundamentaram as decisões iniciais do projeto. O **Capítulo 6** apresenta o Projeto do Sistema, incluindo a identidade visual, prototipação de telas, planejamento das sprints, modelagem do banco de dados e as considerações técnicas que orientaram a implementação.

O **Capítulo 7** detalha o **Desenvolvimento do Sistema**, abordando a implementação prática das funcionalidades principais, a orquestração das integrações com as APIs externas (como Pinata, Mercado Pago e Cloudinary) e a interação efetiva com a blockchain.

Por fim, o **Capítulo 8** apresenta a **Conclusão**, com reflexões sobre o projeto, os resultados alcançados em comparação aos objetivos propostos e as perspectivas para trabalhos futuros.

2 TRABALHOS RELACIONADOS

Com o avanço das tecnologias digitais e a crescente demanda por métodos eficazes de proteção de propriedade intelectual, diversas plataformas e soluções vêm sendo desenvolvidas com o objetivo de facilitar o registro e a comprovação de autoria de obras artísticas musicais utilizando a tecnologia *blockchain*. Entre essas soluções, destacam-se a *Authora Digital*, *InspireIP*, *MyWrites* e *Ujo Music*, cada uma com abordagens específicas voltadas para o ecossistema musical.

2.1 Authora Digital

A *Authora Digital* é uma plataforma brasileira voltada ao registro de obras autorais digitais, como textos, imagens, músicas e outros tipos de conteúdo. O principal objetivo da plataforma é fornecer uma forma acessível e prática de comprovação da anterioridade de uma obra, utilizando carimbo de data e hora (*timestamp*) registrado na **blockchain Ethereum Classic (BEC)** (Authora Digital, 2024).

Diferentemente de plataformas que armazenam arquivos ou utilizam IPFS, a Authora não realiza upload dos conteúdos enviados pelos criadores. O registro ocorre por meio da geração de um *hash* local no dispositivo do autor, que é então registrado na blockchain. Essa abordagem garante a confidencialidade das obras, mas também implica que o criador é o único responsável pela guarda e integridade dos arquivos originais. Se o arquivo for alterado ou perdido, o registro correspondente pode se tornar inválido como prova de autoria.

Embora ofereça uma solução acessível e de rápida utilização, a *Authora Digital* adota um modelo centralizado de operação. Nesse formato, os registros são realizados por meio de uma carteira digital controlada pela própria plataforma, e não diretamente pelo criador. Como consequência, o nome do autor não é associado diretamente à transação na blockchain, o que pode reduzir a autonomia e o controle do criador sobre seu registro. Por outro lado, esse modelo simplifica o processo para usuários leigos, que não precisam lidar com a criação de carteiras digitais, aquisição de criptomoedas ou configuração de redes descentralizadas.

A plataforma busca oferecer uma alternativa prática e juridicamente válida ao modelo tradicional de registro autoral, com foco na acessibilidade, na usabilidade e na redução de custos. Atualmente, a *Authora Digital* opera com um sistema de créditos, em que cada crédito permite realizar um registro na blockchain. O plano básico está disponível a partir de R$ 14,90, valor que concede ao criador o direito a um registro, com validade de até um ano para utilização do crédito adquirido. Embora não ofereça um plano gratuito, o custo de entrada é relativamente acessível em comparação ao método tradicional da Biblioteca Nacional, que cobra uma taxa fixa por submissão (Authora Digital, 2024).

A seguir, são apresentadas as principais vantagens e desvantagens observadas na utilização da plataforma Authora Digital.

Vantagens:

- Registro rápido com marca temporal imutável na blockchain Ethereum Classic;
- Interface acessível em português, voltada ao público brasileiro;
- Certificado digital com *timestamp* imutável que pode ser verificado publicamente.
- Preserva a confidencialidade da obra, pois o arquivo não é enviado à plataforma;

Desvantagens:

- Não possui integração direta com órgãos de arrecadação de direitos (como o ECAD (Escritório Central de Arrecadação e Distribuição));
- O criador precisa manter o arquivo original inalterado; se for perdido ou modificado, o registro pode se tornar inválido em caso de contestação;
- Não utiliza armazenamento descentralizado (como o IPFS);
- O registro é feito com a carteira da própria empresa, e não do criador.
- Dependência da plataforma para validação do autor em caso de disputa.
- Não permite múltiplos arquivos por registro (ex.: letra + áudio + partitura).

2.2 InspireIP

A *InspireIP* é uma plataforma nacional voltada ao registro e à certificação de obras digitais por meio da tecnologia blockchain. Seu foco é oferecer uma solução acessível e juridicamente segura para a proteção de propriedade intelectual, documentos e evidências digitais. A plataforma integra armazenamento descentralizado via IPFS e emite certificados com carimbo de tempo (ICP-Brasil((Infraestrutura de Chaves Públicas Brasileira))), conferindo validade jurídica aos registros realizados (InspireIP, 2024).

O processo é todo digital e finalizado em poucos minutos, sem a necessidade de conhecimento técnico sobre blockchain. Os registros são realizados por uma carteira da própria plataforma, o que simplifica o uso para autores leigos, mas limita a autonomia do criador sobre a titularidade da transação na rede.

A InspireIP permite o registro de arquivos em qualquer formato, incluindo imagens, vídeos, áudios, códigos-fonte, planilhas, PDFs e até conversas e e-mails, caracterizando uma plataforma versátil tanto para artistas quanto para advogados, startups e empresas de tecnologia.

A plataforma oferece dois modelos principais: registro avulso e planos de assinatura. O registro individual tem custo de R$ 79,00, enquanto os planos mensais variam entre R$ 99,00

e R$ 120,00, com 1 crédito incluso por mês e possibilidade de adquirir registros adicionais por R$ 59,00 cada.

A seguir, são apresentadas as principais vantagens e desvantagens observadas na utilização da plataforma.

Vantagens:

- Registro rápido (processo concluído em até 5 minutos);
- Armazenamento descentralizado no IPFS;
- Certificados com *timestamp* (ICP-Brasil) e QR Code de validação;
- Suporte a diversos tipos de arquivos e segmentos profissionais.

Desvantagens:

- O registro é feito pela carteira da plataforma, não garantindo vinculação direta do autor na blockchain;
- Custo elevado para registros avulsos (R$ 79,00 cada), tornando planos mais vantajosos apenas para criadores frequentes;
- Ausência de vínculo com órgãos de arrecadação de direitos autorais, como o ECAD.

2.3 MyWrites

A *MyWrites* é uma plataforma digital voltada à proteção de direitos autorais, com foco no registro de composições musicais e textos literários por meio da tecnologia blockchain. Diferentemente de plataformas que apenas oferecem o certificado com hash da obra, o criador pode realizar o upload da letra ou do áudio e gerar um certificado digital contendo os dados do registro (MYWRITES, 2025).

A plataforma é voltada principalmente para compositores independentes, não exigindo que o autor possua ou configure uma carteira blockchain própria. Os registros são realizados por uma carteira institucional gerida pela própria plataforma, o que simplifica o uso para usuários leigos. No entanto, esse modelo também implica que o nome do autor não é vinculado diretamente na transação blockchain. Ao final do processo, é gerado um certificado digital contendo o título da obra, a data do registro, os dados do autor e um link de verificação na rede blockchain utilizada.

Apesar de oferecer um plano gratuito, este não permite a realização de registros. Para efetuar qualquer registro de obra, o criador deve adquirir o plano anual, atualmente comercializado por R$ 478,80.

Embora a proposta da plataforma seja prática, a interface apresenta certas limitações quanto à usabilidade. A navegação depende fortemente de balões de ajuda e notificações, e não

há suporte contextual claro durante as etapas do processo, o que pode representar uma barreira para criadores menos familiarizados com a ferramenta. Essa deficiência pode ser observada na Figura 17.

Figura 1 – Interface da plataforma MyWrites com destaque para os balões de ajuda que orientam o criador durante o registro

A seguir, são apresentadas as principais vantagens e desvantagens observadas na utilização da plataforma.

Vantagens:

- Foco específico no registro de composições musicais e letras, atendendo um público bem definido;
- Processo de registro em blockchain com emissão de certificado digital;
- Dispensa do uso de carteira digital por parte do criador;

Desvantagens:

- Plano gratuito extremamente limitado — não permite realizar registros;
- Custo elevado do plano anual (R$ 478,80), dificultando o acesso a autores iniciantes;
- Interface pouco intuitiva, com excesso de mensagens flutuantes e ausência de suporte passo a passo;
- O registro é feito pela carteira da plataforma, o que reduz a autonomia jurídica direta do autor.
- A plataforma não armazena os arquivos originais das obras.

2.4 Ujo Music

A *Ujo Music* foi uma das primeiras plataformas a explorar o uso da tecnologia blockchain na indústria musical, com foco na automação da distribuição de royalties através de *smart contracts*. Apesar de ter sido um projeto inovador, a plataforma foi descontinuada em 2019 (MUSIC, 2019).

A *Ujo Music* buscava transformar a indústria musical por meio da tecnologia blockchain. Desenvolvida sobre a rede Ethereum, sua proposta era permitir que músicos independentes distribuíssem suas obras diretamente aos consumidores, recebendo pagamentos automatizados através de contratos inteligentes (*smart contracts*) (MUSIC, 2019).

O diferencial da Ujo Music estava no uso de contratos programáveis para divisão de royalties. Cada vez que uma música era vendida ou ouvida, os lucros eram automaticamente repartidos entre os colaboradores da obra (como letristas, intérpretes e produtores), de acordo com uma porcentagem previamente definida no contrato inteligente. Essa estrutura eliminava intermediários tradicionais da cadeia fonográfica, como editoras e distribuidoras, promovendo maior transparência e autonomia para os artistas.

Apesar de sua visão inovadora, a plataforma enfrentou diversos desafios técnicos e de adoção no mercado. O uso de blockchain e criptomoedas ainda era incipiente na indústria musical, e a complexidade técnica envolvida para artistas configurarem seus contratos e carteiras digitais limitou sua popularização.

A seguir, são apresentadas as principais vantagens e desvantagens observadas na utilização da plataforma.

Vantagens:

- Eliminação de intermediários na distribuição de receitas musicais;
- Transparência e automação no repasse de royalties;
- Pagamentos instantâneos e distribuição proporcional entre os colaboradores da obra.

Desvantagens:

- Plataforma atualmente descontinuada (encerrada em 2019);
- Elevada complexidade técnica, dificultando o uso por artistas sem conhecimento em blockchain;
- Modelo não adaptado à realidade de mercados como o brasileiro, com foco predominante no mercado internacional.

2.5 Considerações

A análise das plataformas Authora Digital, InspireIP, MyWrites e Ujo Music permite identificar avanços significativos no uso da tecnologia blockchain como alternativa moderna para o registro de autoria e a proteção de obras intelectuais. Cada solução apresenta diferentes abordagens, com variações nos modelos de uso, grau de descentralização e público-alvo.

A Tabela 1 apresenta uma comparação entre essas plataformas, destacando critérios como tipo de registro, armazenamento, modelo de carteira, validade jurídica e status atual.

Tabela 1 – Comparação entre plataformas de registro autoral com blockchain

Critério	Authora Digital	InspireIP	MyWrites	Ujo Music
Foco principal	Obras gerais	Obras e provas digitais	Letras e músicas	Royalties musicais
Tipo de registro	Hash na blockchain	Hash + IPFS	Hash na blockchain	Smart contracts
Carteira do autor	Não	Não	Não	Sim
Armazenamento descentralizado	Não	Sim (IPFS)	Não	Parcial
Certificado digital	Sim	Sim (ICP-Brasil)	Sim	Não aplicável
Validade jurídica	Sim (MP 2.200-2)	Sim	Sim (Lei 9.610)	Sem foco jurídico
Preço inicial (2025)	R$ 14,90 por registro	R79*avulso/R* 99-120 mês	R$ 478,80 por ano	Descontinuada
Status atual	Ativa	Ativa	Ativa	Encerrada (2019)

De um modo geral, plataformas como a **InspireIP** se destacam pela combinação de registro em blockchain com armazenamento no IPFS e certificação com timestamp ICP-Brasil, oferecendo uma solução robusta e juridicamente amparada. Já a **Authora Digital** e a **MyWrites** priorizam a simplicidade e o baixo custo, mas não armazenam os arquivos originais, o que exige maior cuidado por parte do autor quanto à preservação da obra. A **Ujo Music**, apesar de descontinuada, representou um marco no uso de *smart contracts* para distribuição de royalties e inspirou modelos posteriores com foco em autonomia financeira para artistas.

3 REFERENCIAL TEÓRICO

Para o desenvolvimento da plataforma *WriteSafe*, destinada ao registro e proteção de obras autorais no meio digital, é essencial compreender os conceitos fundamentais que embasam sua proposta. Este capítulo abordará não apenas temas como direitos autorais, blockchain, IPFS, hashing e validade jurídica digital, que formam a base conceitual da solução, mas também as metodologias, ferramentas e tecnologias que serão utilizadas no processo de desenvolvimento da aplicação. A construção de uma aplicação web eficiente e confiável demanda a escolha criteriosa de linguagens de programação, bibliotecas e frameworks, além do emprego de boas práticas de programação e de metodologias de desenvolvimento. Esses aspectos são essenciais para garantir a qualidade do produto final, otimizar o tempo de implementação e criar bases sólidas para futuras melhorias. Adicionalmente, serão apresentados os processos e ferramentas que facilitarão o planejamento e a execução do projeto, considerando desde a organização do código até a manutenção da plataforma ao longo do tempo. A abordagem adotada neste capítulo busca fornecer uma visão ampla e integrada, que alicerça o desenvolvimento da solução proposta.

3.1 Processos de Desenvolvimento

3.1.1 Scrum

O Scrum é um framework ágil amplamente adotado em projetos de desenvolvimento de software. Ele divide o trabalho em sprints — ciclos curtos, time-boxed e iterativos—que viabilizam inspeção frequente, adaptação rápida a mudanças e entrega incremental de valor. Cada sprint começa com um planejamento de metas claras, segue com reuniões diárias de acompanhamento (Daily Scrum) e termina com revisão e retrospectiva, fases em que o produto é inspecionado e o processo de trabalho, ajustado. Ao incentivar comunicação transparente, feedback contínuo e priorização de funcionalidades que geram maior benefício ao cliente, o Scrum eleva a colaboração da equipe e contribui para o aumento da qualidade do produto final. (SCHWABER; SUTHERLAND, 2020).

3.1.2 Kanban

Kanban é uma abordagem visual para o gerenciamento de fluxo de trabalho, onde tarefas são representadas em um quadro dividido em colunas que indicam os diferentes estados do processo, como "A Fazer", "Em Progresso"e "Concluído". Ele permite que equipes visualizem gargalos, organizem prioridades e mantenham um fluxo constante de entregas (ANDERSON, 2010).

3.1.3 MoSCoW

O MoSCoW é uma técnica de priorização de requisitos desenvolvida para auxiliar equipes de projeto a alcançarem um entendimento comum sobre a importância de cada funcionalidade em um determinado prazo. O método é fundamental para gerenciar o escopo e as expectativas, garantindo que o foco permaneça nos itens de maior valor para o produto final (DSDM Consortium, 2014).

O método organiza os requisitos em quatro categorias principais, representadas pelo acrônimo:

- **Must Have (Deve Ter):** Refere-se às funcionalidades que são essenciais para o funcionamento mínimo e para a viabilidade do projeto. Sem elas, o produto não pode ser entregue na data acordada.
- **Should Have (Deveria Ter):** Abrange funcionalidades importantes, mas não vitais. O sistema pode funcionar sem elas, mas sua ausência representa uma perda de valor significativa. Devem ser incluídas se possível.
- **Could Have (Poderia Ter):** São funcionalidades desejáveis, mas de menor prioridade. São consideradas itens de menor impacto que podem ser incluídos se houver tempo e recursos disponíveis, sem comprometer os outros requisitos.
- **Won't Have This Time (Não Terá Desta Vez):** Corresponde aos requisitos que foram explicitamente acordados como fora do escopo para o período de tempo atual, mas que podem ser reconsiderados em versões ou projetos futuros.

3.2 Ferramentas de Desenvolvimento

3.2.1 Git e GitFlow

Git é um sistema de controle de versões distribuído que permite rastrear alterações no código e colaborar de forma eficiente (CHACON; STRAUB, 2014). GitFlow é uma extensão de fluxo de trabalho para Git, que define uma estrutura de ramificação organizada, facilitando o gerenciamento de funcionalidades, correções e lançamentos de forma estruturada (NVIE, 2010).

3.2.2 GitHub

GitHub é uma plataforma de hospedagem de código que utiliza o sistema de controle de versões Git. Ele facilita a colaboração em projetos de software, permitindo que várias pessoas

trabalhem simultaneamente no mesmo código. Oferece funcionalidades como pull requests, revisão de código e integração com diversas ferramentas de desenvolvimento (GITHUB, 2023a).

3.2.3 GitHub Projects

O **GitHub Projects** é o módulo de gestão de projetos nativo da plataforma GitHub. Ele permite criar quadros *kanban* totalmente personalizáveis que agrupam *issues* e *pull requests* do repositório, oferecendo uma visão única do backlog, do trabalho em andamento e das entregas concluídas.

- **Colunas configuráveis**: *To Do*, *In Progress*, *Done* ou qualquer fluxo definido pela equipe.
- **Automação de cartões**: movimenta tarefas automaticamente conforme o estado do *issue* ou do *pull request*.
- **Campos e filtros**: permitem priorizar tarefas, estimar esforço e monitorar prazos.
- **Gráficos de progresso**: *burndown* e estatísticas de coluna integrados ao histórico do repositório.

Ao concentrar código-fonte e acompanhamento de tarefas em um mesmo ambiente, o GitHub Projects facilita a coordenação entre desenvolvedores, reduz a dependência de ferramentas externas e reforça a rastreabilidade das decisões de projeto (GITHUB, 2023b).

3.2.4 Estratégia de Monorepo

No desenvolvimento deste projeto, será adotada a estratégia de monorepo, na qual todos os componentes do sistema estarão armazenados em um único repositório. Essa abordagem facilita o compartilhamento de código entre diferentes partes do sistema, simplifica o gerenciamento de dependências e integrações, e proporciona maior visibilidade do progresso geral do projeto. Além disso, a estrutura de monorepo será combinada com as práticas do GitFlow, garantindo organização e controle no desenvolvimento colaborativo.

3.2.5 Docker

O Docker é uma plataforma de virtualização baseada em contêineres que permite empacotar aplicações e suas dependências em ambientes isolados e portáteis. Diferente das máquinas virtuais tradicionais, os contêineres compartilham o mesmo kernel do sistema operacional, o que reduz significativamente o consumo de recursos e o tempo de inicialização (DOCKER, 2023).

Para simplificar o gerenciamento de múltiplos contêineres que compõem uma mesma aplicação — como serviços de backend, banco de dados e ferramentas auxiliares —, o Docker Compose pode ser utilizado. Essa ferramenta permite definir, em um único arquivo de configuração (docker-compose.yml), todos os serviços necessários, suas redes, volumes e variáveis de ambiente, automatizando o processo de inicialização do ambiente completo. (DOCKER, 2023).

3.3 Tecnologias do Lado do Cliente

3.3.1 Typescript

TypeScript é uma linguagem de programação baseada em JavaScript, mas que adiciona tipagem estática opcional e outros recursos avançados. Ele ajuda a identificar erros antes do tempo de execução e melhora a manutenção e escalabilidade de aplicações (MICROSOFT, 2023).

3.3.2 Angular

Angular é um framework de desenvolvimento front-end baseado em TypeScript, projetado para a construção de aplicações web dinâmicas. Ele oferece ferramentas poderosas, como dois sentidos de ligação de dados (two-way data binding), injeção de dependência e um sistema de componentes altamente modular (GOOGLE, 2023).

3.3.3 Tailwind

Tailwind é um framework CSS utilitário que fornece classes pré-definidas para estilização direta no HTML. Ele permite criar interfaces rápidas e personalizáveis sem a necessidade de escrever CSS do zero, tornando o desenvolvimento mais ágil e organizado (TEAM, 2023b).

3.3.4 Flowbite

Flowbite é uma biblioteca de componentes UI construída sobre Tailwind, oferecendo elementos prontos como botões, tabelas e modais. Ele combina design moderno e responsivo com facilidade de integração, acelerando o desenvolvimento de interfaces (THEMESBERG, 2023).

3.4 Tecnologias do Lado do Servidor

3.4.1 Node.js

Node.js é uma plataforma de execução de JavaScript no lado do servidor, baseada no motor V8 do Chrome. Ele permite o desenvolvimento de aplicações rápidas e escaláveis, especialmente em arquiteturas orientadas a eventos e de entrada/saída não bloqueante (TILKOV; VINOSKI, 2010).

3.4.2 Nest.js

Nest.js é um framework progressivo de desenvolvimento de aplicações Node.js que utiliza TypeScript e segue os princípios da programação modular e orientada a objetos (MYŚLIWIEC, 2023).

3.4.3 Hardhat e Ethers.js

Enquanto o NestJS gerencia a API web, um conjunto separado de ferramentas é necessário para interagir com a blockchain.

O Hardhat é um ambiente de desenvolvimento voltado para contratos inteligentes (smart contracts) escritos em Solidity. Ele fornece ferramentas integradas para compilação, teste, implantação e depuração de contratos em diferentes redes blockchain. Entre seus recursos estão a simulação de rede local, o gerenciamento de artefatos de compilação e a integração com bibliotecas JavaScript (FOUNDATION, 2023).

O Ethers.js, por sua vez, é uma biblioteca JavaScript voltada à interação entre aplicações e a blockchain Ethereum (ou redes compatíveis, como a Polygon). Ela permite conectar-se a nós da rede, assinar transações e invocar funções de contratos inteligentes, oferecendo uma interface moderna e segura para aplicações descentralizadas (ETHERS, 2023).

Além dessas ferramentas, o ambiente de desenvolvimento de contratos inteligentes geralmente envolve o uso combinado de tecnologias complementares. A linguagem *Solidity* é a base para a escrita dos contratos, enquanto o serviço *Alchemy* atua como provedor de nós RPC, permitindo a conexão entre o ambiente Hardhat e a rede blockchain. A carteira digital *MetaMask* é empregada para o gerenciamento das chaves privadas e credenciais utilizadas nas transações, assegurando a assinatura segura dos contratos. Em conjunto, essas ferramentas possibilitam a implantação de contratos inteligentes em redes compatíveis com a Ethereum, como a *Polygon*, sendo a rede de testes Amoy uma das opções frequentemente utilizadas em ambientes de desenvolvimento e validação.

3.4.4 REST, JSON e JWT

As tecnologias REST, JSON e JWT podem ser usadas para garantir comunicação padronizada, transporte leve de dados e autenticação sem estado.

O REST (Representational State Transfer) foi proposto por Fielding (2000), REST define um conjunto de restrições para sistemas distribuídos baseados em hipermídia. Seus princípios centrais incluem: comunicação *stateless*, identificação clara de recursos por URI, operações padronizadas do protocolo HTTP (`GET`, `POST`, `PUT`, `DELETE`) e representações transferidas de forma uniforme. A adoção dessas restrições aumenta a interoperabilidade entre clientes e servidores independentes da plataforma.

O JSON (JavaScript Object Notation) é um formato leve, textual e independente de linguagem para troca de dados estruturados, padronizado pela ECMA-404 (2023). Sua sintaxe baseada em pares `chave-valor` e listas ordenadas favorece tanto a leitura humana quanto o processamento automatizado, o que explica seu amplo uso como corpo de requisições e respostas em serviços RESTful.

O JWT (JSON Web Token) (Jones, 2015) é um padrão aberto (RFC 7519) que descreve um método compacto, seguro e independente de estado para transmitir informações entre partes. Um token é composto por *header*, *payload* e *signature*, codificados em Base64URL e concatenados por pontos. A assinatura assegura integridade; opcionalmente, pode ser empregada criptografia para confidencialidade. Por incluir declarações (*claims*) no próprio token, o JWT permite autenticação e autorização sem necessidade de sessão persistente no servidor.

3.4.5 Gateway de Pagamento (Mercado Pago)

Um gateway de pagamento é um serviço responsável por intermediar transações financeiras entre uma aplicação e as instituições bancárias ou operadoras de cartão. Ele atua como um canal seguro de comunicação, realizando a autenticação, a autorização e a liquidação dos pagamentos de forma automatizada. Esses serviços geralmente oferecem APIs e SDKs que permitem integrar sistemas web ou móveis a meios de pagamento eletrônicos, garantindo criptografia dos dados sensíveis, conformidade com padrões de segurança (como PCI DSS) e monitoramento de fraudes (PAGO, 2023).

Entre os principais provedores disponíveis no mercado brasileiro estão o Mercado Pago, PagSeguro e Stripe, que se diferenciam pela facilidade de integração, custo por transação e suporte a diferentes formas de pagamento. A adoção de um gateway desse tipo é fundamental para aplicações que envolvem monetização de serviços ou cobrança por uso, viabilizando operações financeiras de forma segura, rastreável e escalável.

3.4.6 Webhooks

Os webhooks são um padrão de arquitetura utilizado para permitir a comunicação assíncrona entre aplicações web. Diferentemente do modelo tradicional de polling, em que uma aplicação precisa consultar periodicamente outra para verificar alterações, o webhook adota uma abordagem reativa: sempre que um evento ocorre em um sistema, ele envia uma requisição HTTP POST contendo os dados do evento a um endpoint previamente configurado em outro serviço (FYI, 2023).

Essa técnica é amplamente utilizada em integrações entre APIs modernas, pois reduz o consumo de recursos, aumenta a eficiência e permite a automação de fluxos de trabalho em tempo real. Plataformas de pagamento, hospedagem, autenticação e notificações frequentemente utilizam webhooks para sinalizar eventos como confirmações de transação, atualizações de status ou novos registros em sistemas externos.

3.4.7 ORM e Prisma

ORM (Object-Relational Mapping) é uma técnica que mapeia objetos em código para tabelas em bancos de dados, facilitando a manipulação de dados (AMBLER, 2003). Prisma é uma ferramenta ORM moderna que simplifica a interação com bancos de dados, fornecendo uma interface intuitiva para operações CRUD e suporte a migrações (TEAM, 2023a).

3.4.8 MySQL

MySQL é um sistema de gerenciamento de banco de dados relacional (RDBMS) amplamente utilizado. Ele é conhecido por sua confiabilidade, desempenho e simplicidade, sendo adequado para armazenar e gerenciar grandes volumes de dados em aplicações web (CORPORATION, 2023).

3.4.9 Cloudinary

Cloudinary é uma plataforma baseada em nuvem para gerenciamento e entrega de arquivos multimídia, como imagens, vídeos e documentos. Ela oferece ferramentas robustas para armazenamento, otimização e entrega dinâmica de arquivos, garantindo rapidez e eficiência em aplicações que demandam consumo frequente de mídia (CLOUDINARY, 2023).

3.5 Fundamentação Conceitual da Plataforma *WriteSafe*

Este tópico apresenta os conceitos fundamentais que sustentam a plataforma, incluindo direitos autorais, blockchain, IPFS, hashing e validade jurídica digital. Esse conteúdo serve como base para a compreensão dos aspectos técnicos e jurídicos relacionados ao registro de obras autorais no meio digital.

3.5.1 Direitos Autorais e Prova de Anterioridade Digital

A proteção aos direitos autorais garante ao criador o controle sobre o uso de sua obra, incluindo o direito de ser reconhecido como autor e de obter remuneração por sua exploração. Tais prerrogativas estão previstas em normas como a Convenção de Berna (World Intellectual Property Organization, 1979) e na Lei Brasileira nº 9.610/1998 (LEI. . . , 1998), que afirma que o direito de autoria independe de registro, sendo reconhecido aquele que se identifica como autor, salvo prova em contrário.

Apesar disso, a obtenção de um registro oficial, como o fornecido pelo Escritório de Direitos Autorais (EDA) da Biblioteca Nacional, pode ser crucial para comprovar a anterioridade da criação em disputas legais. No entanto, esse processo é muitas vezes limitado por burocracia, custos e falta de acessibilidade — especialmente para artistas independentes (NACIONAL, 2025). Alguns métodos tradicionais como registro em cartório, envio por e-mail ou Correios têm sido usados como prova, mas apresentam limitações quanto à integridade, originalidade e agilidade.

Com a digitalização dos processos criativos e a facilidade de compartilhamento online, torna-se mais difícil garantir a autenticidade de obras intelectuais. Nesse contexto, surgem alternativas digitais que buscam fornecer provas confiáveis de autoria, como o uso de blockchain e armazenamento descentralizado.

3.5.2 Blockchain: Funcionamento, Imutabilidade e Timestamp

A tecnologia blockchain é uma estrutura de dados descentralizada e criptograficamente segura, composta por blocos que armazenam transações encadeadas de forma imutável. Cada bloco contém um identificador único, chamado de *hash*, e um *timestamp* — registro da data e hora — o que permite comprovar a existência de determinada informação em um momento específico (NAKAMOTO, 2008).

A imutabilidade e a transparência dessa tecnologia tornam o blockchain uma ferramenta promissora para registros de autoria digital. Ao registrar o hash de um arquivo em uma rede pública como a Polygon, é possível criar uma prova de anterioridade permanente e auditável com reconhecimento internacional. Além disso, o uso de contratos inteligentes (*smart contracts*)

pode automatizar tarefas como distribuição de royalties e validação de registros, oferecendo eficiência e confiança adicionais ao processo (BUTERIN, 2014).

3.5.3 IPFS: Armazenamento Descentralizado de Arquivos

Com o crescimento exponencial da produção de conteúdo digital e a necessidade de preservar a integridade e a disponibilidade dos dados, surgiram alternativas ao modelo tradicional de armazenamento centralizado. Nesse contexto, o *InterPlanetary File System* (IPFS) desponta como uma das principais soluções de armazenamento descentralizado e distribuído da atualidade.

O IPFS é um protocolo peer-to-peer (P2P) que conecta computadores em uma rede global para compartilhar e armazenar arquivos de forma colaborativa, sem depender de um servidor central. Diferentemente dos sistemas convencionais, que localizam arquivos por meio de um endereço em um servidor específico, o IPFS utiliza um identificador único baseado no conteúdo do arquivo — um hash criptográfico conhecido como CID (*Content Identifier*) (BENET, 2014).

O CID é gerado a partir do conteúdo do arquivo, e qualquer alteração, por menor que seja, resulta em um novo identificador, garantindo unicidade e imutabilidade. Essa característica assegura que o identificador permaneça consistente enquanto o conteúdo do arquivo não for alterado, independentemente de sua localização na rede. Essa abordagem não apenas preserva a integridade dos dados, mas também facilita a recuperação e a verificação confiável do conteúdo.

No contexto do registro de direitos autorais, o IPFS, aliado ao uso de CIDs, permite armazenar músicas, letras, partituras e outros materiais de forma acessível, segura e resistente a alterações. Isso elimina a dependência de servidores centralizados e aumenta a durabilidade e a confiabilidade do registro, tornando-o uma solução robusta e moderna para criadores autorais.

3.5.4 Hashing e sua Aplicação em Registros Digitais

O hashing é uma técnica que converte qualquer conteúdo digital em uma sequência fixa de caracteres, chamada hash. Essa sequência funciona como uma impressão digital do arquivo: qualquer modificação mínima no conteúdo gera um hash completamente diferente.

No processo de registro de autoria digital, o hash do conteúdo é gerado e registrado na blockchain. Mesmo que o arquivo original não seja armazenado publicamente, o hash serve como prova criptográfica de que aquele conteúdo existia em determinado momento e permanece inalterado desde então. Essa técnica permite manter a confidencialidade da obra, ao mesmo tempo em que assegura sua autenticidade.

3.5.5 Certificados Digitais e Validade Jurídica

A validade jurídica de documentos eletrônicos no Brasil é reconhecida pela Medida Provisória nº 2.200-2/2001 (BRASIL, 2001), que instituiu a Infraestrutura de Chaves Públicas Brasileira (ICP-Brasil). Essa medida garante autenticidade, integridade e validade jurídica a documentos eletrônicos assinados digitalmente com certificados emitidos por autoridades certificadoras credenciadas. No entanto, embora a ICP-Brasil seja exigida em alguns contextos formais, como atos notariais e administrativos, a crescente digitalização de processos levou à consideração de outras tecnologias seguras e auditáveis como meios válidos de comprovação jurídica.

Nesse cenário, o uso da tecnologia blockchain vem se consolidando como uma alternativa moderna e eficaz para a autenticação de registros digitais, especialmente no campo da propriedade intelectual. Plataformas que utilizam redes públicas como Ethereum e Polygon permitem o registro de hashes de arquivos, gerando certificados digitais contendo marca temporal imutável, identificador da transação blockchain e link público para verificação (NAKAMOTO, 2008; BUTERIN, 2014). Esses certificados funcionam como provas complementares de autoria, com crescente aceitação em disputas judiciais, tanto no Brasil quanto no exterior (JusBrasil, 2023; PATENTES, 2023).

Internacionalmente, a validade probatória da blockchain já é reconhecida em diversos países. Na China, por exemplo, o Supremo Tribunal Popular determinou que dados registrados em blockchain podem ser admitidos como prova legal em litígios civis e comerciais (CAO, 2018). Na União Europeia, o Regulamento eIDAS (UNION, 2014) estabelece as bases legais para serviços de confiança digital, permitindo que tecnologias emergentes como blockchain sejam utilizadas para validar documentos e transações, desde que observados princípios de integridade e autenticidade. Nos Estados Unidos, embora não exista legislação federal específica sobre blockchain, tribunais estaduais como os de Vermont e Arizona já reconheceram expressamente a validade de evidências registradas em redes distribuídas (VERMONT, 2017; LEGISLATURE, 2018).

Assim, ainda que a solução proposta neste trabalho não utilize certificação pela ICP-Brasil, ela se fundamenta em princípios amplamente aceitos de imutabilidade, rastreabilidade e transparência proporcionados pela blockchain pública, oferecendo respaldo técnico e jurídico adequado para a proteção de obras autorais no meio digital.

4 MATERIAIS E MÉTODOS

4.1 Materiais

O desenvolvimento da plataforma WriteSafe combinou tecnologias de duas gerações distintas da web — Web2 e Web3 —, refletindo a transição entre arquiteturas centralizadas e descentralizadas.

A Web2 caracteriza-se por aplicações cliente-servidor tradicionais, com servidores centralizados responsáveis pelo armazenamento e processamento de dados. Já a Web3 representa uma evolução que incorpora princípios de descentralização, autenticação por carteiras digitais e uso de blockchain para registro e validação de informações.

4.1.1 Stack de Tecnologias Web (Web2)

A base da aplicação foi construída sobre uma pilha de tecnologias Web2 robusta e moderna. No *front-end*, foi utilizado o framework **Angular** em sua versão *standalone*, escolhido por sua robustez e arquitetura baseada em componentes. A linguagem **TypeScript** foi adotada em todo o projeto, garantindo tipagem estática para prevenir erros e facilitar a manutenção. Para o design visual, foi utilizado o **Tailwind CSS**, um framework *utility-first* que permitiu a criação de interfaces responsivas de forma ágil. A biblioteca **Flowbite** foi integrada para componentes de UI interativos, e o **NGX-Mask** para a formatação de campos de formulário, como o CPF.

O *back-end* foi desenvolvido com **Node.js**, utilizando o framework **NestJS**. A escolha do *NestJS* foi estratégica, pois sua arquitetura modular inspirada no Angular facilitou a separação de responsabilidades (Módulos, Controladores, Serviços). A comunicação seguiu o padrão **REST**, utilizando **JSON** para transferência de dados. A autenticação foi implementada com **JSON Web Tokens (JWT)**, com o `@nestjs/passport`, garantindo o controle de acesso às rotas protegidas.

Para o gerenciamento do banco de dados, foi utilizado o **MySQL** (v5.7). A interação com o banco foi realizada através do **Prisma**, um ORM (*Object-Relational Mapping*) moderno que simplificou as consultas e garantiu a segurança de tipos.

4.1.2 Stack de Tecnologias Web3 e Serviços Externos

A principal inovação do *WriteSafe* reside na integração de serviços Web3 e APIs externas para orquestrar o fluxo de registro.

Para o armazenamento de arquivos, foi adotada uma arquitetura híbrida. O **Cloudinary** foi utilizado como plataforma de armazenamento intermediário, permitindo o upload rápido e seguro do arquivo do usuário enquanto se aguardava a confirmação do pagamento. Após a

confirmação, os arquivos são então enviados para o **IPFS** (InterPlanetary File System) através da API do **Pinata**, garantindo o armazenamento descentralizado e permanente da prova.

Para a integração com a blockchain, foi utilizado o **Hardhat** como ambiente de desenvolvimento para escrever, compilar e testar o Smart Contract `WriteSafeRegistry.sol`, escrito em **Solidity**. A biblioteca **Ethers.js** foi integrada ao *NestJS* para permitir que o backend se conectasse à rede **Polygon** (através de um nó RPC da **Alchemy**) e chamasse as funções do contrato publicado.

4.1.3 Integração de Pagamentos e Webhooks

Para simular a monetização do serviço, foi integrado o gateway de pagamento **Mercado Pago** em seu ambiente de *sandbox* (testes). O frontend utilizou o SDK JS v2 para a tokenização segura dos dados do cartão de crédito (Checkout Transparente), e o backend utilizou o SDK do Mercado Pago para processar a cobrança.

A confirmação do pagamento foi implementada de forma assíncrona através de **Webhooks**. A ferramenta **ngrok** foi utilizada durante o desenvolvimento para criar um túnel público, permitindo que os servidores do Mercado Pago enviassem notificações de pagamento em tempo real para a aplicação NestJS rodando localmente.

4.1.4 Ambiente de Desenvolvimento

O ambiente de desenvolvimento foi composto pelo **Visual Studio Code** como IDE principal. O **Docker** e o **Docker Compose** foram utilizados estritamente para a instanciação do contêiner do banco de dados **MySQL**, garantindo um ambiente de banco de dados limpo e isolado, enquanto o frontend e o backend foram executados manualmente para agilizar a depuração.

4.2 Métodos

A metodologia adotada neste trabalho fundamentou-se em práticas ágeis de desenvolvimento de software, com foco em entregas iterativas e validação contínua. O processo foi inspirado no framework Scrum, estruturado em ciclos curtos denominados sprints, e complementado pela técnica MoSCoW para priorização das funcionalidades.

O controle de versionamento foi realizado no GitHub, seguindo o fluxo GitFlow, enquanto o gerenciamento das tarefas utilizou o GitHub Projects em formato Kanban, permitindo acompanhar visualmente o progresso e a conclusão das atividades.

A cada sprint foram realizadas revisões internas e testes de integração, garantindo a evolução incremental do produto e a rastreabilidade das entregas.

5 ANÁLISE DO SISTEMA

Este capítulo apresenta a análise do **WriteSafe** — uma plataforma web destinada a gerar prova digital de autoria para obras musicais, letras e partituras. A solução combina duas tecnologias centrais: (i) a *blockchain* (rede Polygon), onde é gravado um *hash* criptográfico imutável da obra; (ii) o *InterPlanetary File System* (IPFS), utilizado para armazenar o arquivo original de forma distribuída.

Todos os registros são executados por uma *carteira institucional* da aplicação, dispensando que o criador configure a própria carteira. Ao término do processo, a plataforma emite um certificado digital com título, hash, data e link público de verificação.

Os requisitos funcionais foram capturados como **Histórias de Usuário**, redigidas no formato:

COMO ⟨perfil⟩, *EU QUERO* ⟨ação ou necessidade⟩ *PARA* ⟨benefício ou propósito⟩.

Essa técnica, alinhada ao framework ágil **Scrum**, facilitou a priorização e o acompanhamento iterativo do trabalho. Cada história recebeu um identificador sequencial **HU$$**, sendo associada a um dos três perfis principais: **Criador**, **Validador Público** e **Administrador**.

Para ordenar o backlog, adotou-se o método **MoSCoW**. O escopo desta análise definiu um *Produto Mínimo Viável (MVP)*, contendo apenas as histórias marcadas como *Must Have*. Funcionalidades classificadas como *Should*, *Could* ou *Won't* foram planejadas para versões futuras, permitindo uma evolução incremental.

5.1 Perfis de Usuários

Inicialmente, a aplicação previa três perfis de usuários com funções distintas. Contudo, ao longo do desenvolvimento, o perfil Validador Público foi reformulado: em vez de representar um usuário autenticado, passou a atuar como uma função estendida do sistema, destinada à consulta e validação pública dos registros. A Figura 2 ilustra visualmente esses papéis e na sequência, a descrição individual de cada um.

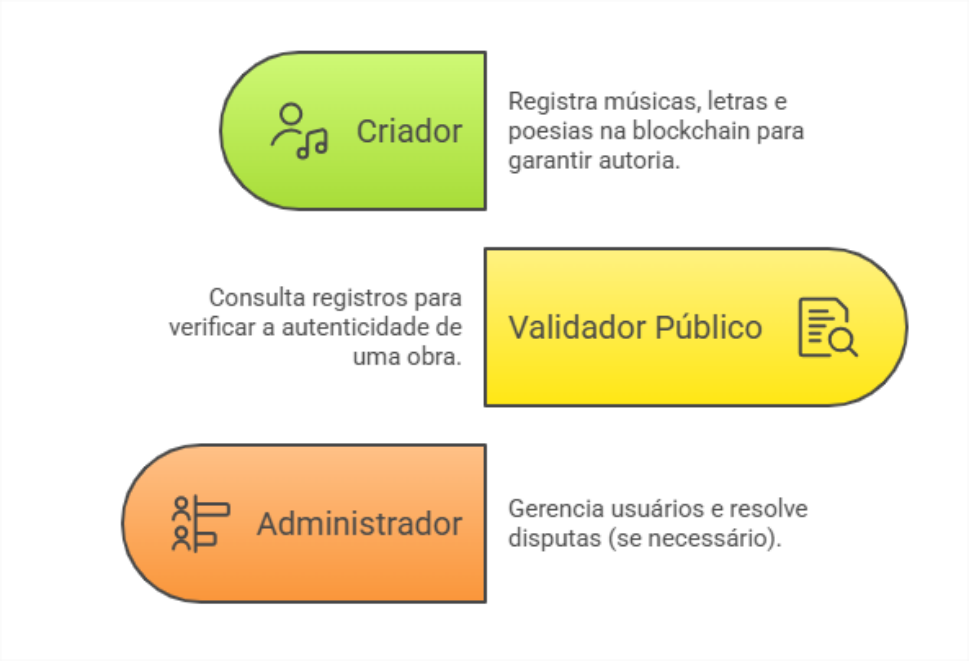

Figura 2 – Representação dos perfis de usuários da plataforma WriteSafe.

- **Criador (Compositor/Autor/Músico)**: usuário principal da plataforma. É o responsável pelo upload das obras e pela solicitação de registro autoral. Busca segurança, praticidade e validade jurídica no processo de comprovação de autoria.

  A Figura 3 apresenta o fluxo de interação do Criador com a plataforma, desde o momento do login até a obtenção do certificado digital de registro. O diagrama permite visualizar a sequência das etapas automatizadas envolvidas no processo, incluindo o envio da obra, o armazenamento opcional no IPFS e a entrega eficiente dos arquivos por meio do Cloudinary.

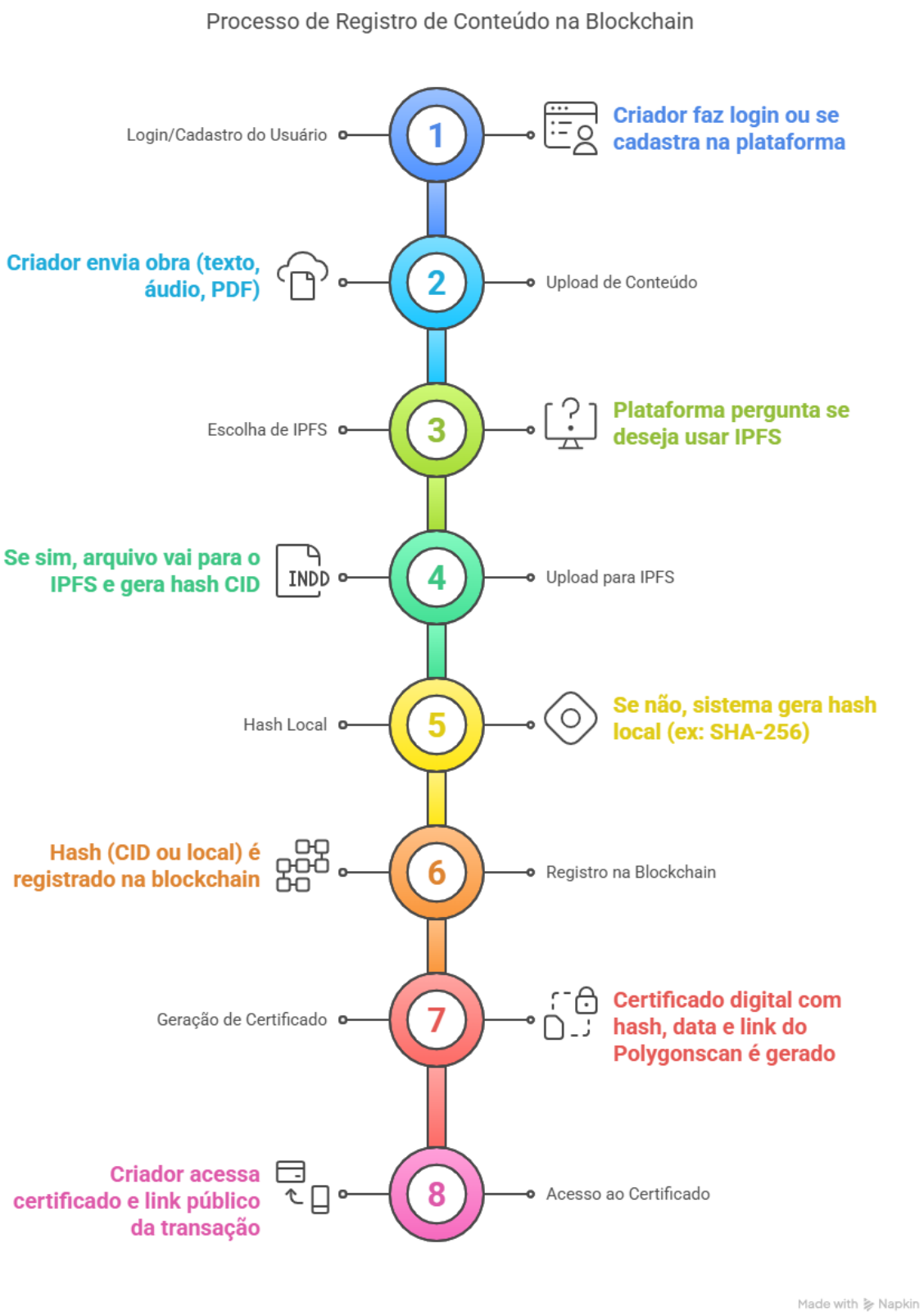

Figura 3 – Processo de registro de conteúdo na blockchain

- **Validador Público**: na concepção inicial, esse perfil representava qualquer pessoa interessada em verificar a autenticidade de uma obra registrada. Atuaria na consulta pública dos registros e na validação das informações associadas às criações cadastra-

das. No entanto, com a evolução do projeto, o Validador Público deixou de ser tratado como um usuário autenticado e passou a constituir uma extensão da própria plataforma, responsável por oferecer a verificação pública de autenticidade das obras. Essa funcionalidade redireciona o usuário para a transação correspondente na Polygonscan, permitindo a validação direta dos dados registrados na blockchain.

- **Administrador**: responsável pela manutenção da plataforma, gestão de criadores e visualização de métricas.

5.2 Funcionalidades Essenciais

A seguir, são apresentadas as principais histórias de usuário de cada perfil, conforme definidas na fase de análise. Essas histórias detalham o que cada tipo de usuário poderia realizar no sistema e foram organizadas e priorizadas utilizando o método MoSCoW. A estrutura permitiu um acompanhamento claro das etapas de implementação e do impacto esperado de cada funcionalidade.

O escopo do projeto está detalhado no Apêndice A, o qual destaca os principais componentes do sistema, os fluxos de interação entre os usuários e as funcionalidades essenciais.

5.2.1 Criação de Conta (HU01)

- **História do Criador:** *Como criador, quero criar uma conta com e-mail e senha, para acessar a plataforma de forma segura.*
- **Critérios de Aceitação:**
  - O criador deve poder preencher um formulário com e-mail e senha.
  - O sistema deve validar se o e-mail já está cadastrado.
  - A senha deve atender aos critérios mínimos de segurança.
  - Após o cadastro, o criador deve receber um e-mail de confirmação.
  - O acesso à plataforma só será liberado após confirmação do e-mail.
- **Nota de Implementação:** As três primeiras funcionalidades foram implementadas com validação via bcrypt e controle de duplicidade de e-mail. O fluxo de confirmação por e-mail foi despriorizado no MVP, permitindo login imediato após o cadastro.

5.2.2 Envio de Obra (HU02)

- **História do Criador:** *Como criador, quero fazer upload de letras, músicas ou partituras, para registrar minha obra digitalmente.*

- **Critérios de Aceitação:**
  - O criador deve poder escolher e enviar arquivos suportados.
  - O sistema deve validar o formato e o tamanho dos arquivos.
  - O sistema deve solicitar metadados como título e tipo de obra.
  - O sistema deve gerar um hash da obra após o upload.
  - Uma mensagem de sucesso deve ser exibida após o envio.
- **Nota de Implementação:** Todos os critérios foram implementados com suporte a arquivos MP3, TXT e PDF, limite de 10 MB e exibição de mensagens personalizadas. A HU foi expandida para incluir um fluxo de pagamento simulado via Mercado Pago, com coleta de CPF e nome do titular antes do registro final.

5.2.3 Geração de Hash (HU03)

- **História do Criador:** *Como criador, quero que a plataforma gere um hash da obra, para garantir a imutabilidade do conteúdo.*
- **Critérios de Aceitação:**
  - O sistema deve gerar um hash único com algoritmo seguro (ex.: SHA-256).
  - O hash deve ser armazenado junto aos metadados da obra.
  - O hash deve ser exibido ao usuário como comprovante.
- **Nota de Implementação:** A geração do hash foi implementada no backend com o módulo crypto do Node.js, armazenada com os metadados e exibida no certificado PDF.

5.2.4 Registro na Blockchain (HU04)

- **História do Criador:** *Como criador, quero registrar esse hash na blockchain via carteira da plataforma, para obter uma prova de autoria com validade pública.*
- **Critérios de Aceitação:**
  - O sistema deve registrar o hash na rede Polygon.
  - O criador deve receber o ID da transação como resposta.
  - O status da transação (pendente, sucesso, erro) deve ser registrado.
- **Nota de Implementação:** O registro foi realizado na rede de testes Amoy da Polygon, com salvamento do ID e status da transação no banco de dados.

5.2.5 Armazenamento no IPFS (HU05)

- **História do Criador:** *Como criador, quero escolher se meus arquivos serão armazenados no IPFS, para manter uma cópia acessível e descentralizada.*
- **Critérios de Aceitação:**
  - O criador deve poder escolher entre armazenamento local e IPFS.
  - Caso o criador opte pelo armazenamento no IPFS, os arquivos enviados (PDF,MP3, etc.) serão compactados em um arquivo ZIP antes de serem armazenados. Isso garante maior eficiência e organização no armazenamento distribuído.
  - O sistema deve gerar o CID do arquivo.
  - O link IPFS deve estar disponível no painel do usuário.
- **Nota de Implementação:** O sistema foi ajustado para armazenar automaticamente no IPFS via Pinata, removendo a escolha manual do usuário. A compactação em ZIP foi despriorizada e, no fluxo do MVP, utilizou-se o Cloudinary como armazenamento temporário para o processo de pagamento assíncrono.

5.2.6 Certificado Digital (HU06)

- **História do Criador:** *Como criador, quero receber um certificado digital com as informações do registro, para usar como comprovação da autoria.*
- **Critérios de Aceitação:**
  - O sistema deve gerar um PDF com autor, título, hash, data e ID da transação.
  - O certificado deve conter QR Code com link para o Polygonscan.
  - O criador deve poder baixar o certificado a qualquer momento.
  - O sistema deve buscar a obra para ser visualizada pelo criador a partir do Cloudinary.
- **Nota de Implementação:** A geração do certificado foi implementada no backend com pdfkit, contendo QR Code e link para o Polygonscan. A visualização das obras é feita na página “Minhas Obras” com acesso direto ao arquivo via Cloudinary.

5.2.7 Validação Pública (HU07)

- **História do Criador:** *Como criador, quero acessar o link da transação no Polygonscan, para validar publicamente meu registro na blockchain.*

- **Critérios de Aceitação:**
  - O link da transação deve ser exibido após o registro.
  - O criador deve conseguir acessar os dados da transação via Polygonscan.
  - O link deve estar disponível no painel do criador e no certificado.
- **Nota de Implementação:** Funcionalidade totalmente implementada com exibição e persistência do link de verificação pública.

5.2.8 Consulta por Validador (HU08)

- **História do Validador Público:** *Como validador público, quero consultar uma obra usando o hash do IPFS ou o endereço do autor, para verificar a autenticidade do registro.*
- **Critérios de Aceitação:**
  - O sistema deve oferecer filtros por hash, autor e carteira.
  - A consulta deve retornar dados completos da obra.
  - Se não encontrado, o sistema deve informar a ausência de registros.
- **Nota de Implementação:** Esta HU foi classificada como *Won't Have This Time* e despriorizada do MVP para focar no fluxo do criador.

5.2.9 Visualização por Validador (HU09)

- **História do Validador Público:** *Como validador público, quero visualizar dados como a data, o autor e o hash, para confirmar a autoria da obra registrada.*
- **Critérios de Aceitação:**
  - O sistema deve exibir todos os metadados da obra.
  - Os dados exibidos não devem ser editáveis.
  - O validador deve poder exportar um relatório de verificação.
- **Nota de Implementação:** Esta HU foi classificada como *Won't Have This Time* e despriorizada do MVP.

5.2.10 Painel Administrativo (HU10)

- **História do Administrador:** *Como administrador, quero visualizar um painel com estatísticas básicas da plataforma, para acompanhar o número de registros e criadores ativos.*
- **Critérios de Aceitação:**
  - O painel deve exibir número total de registros, criadores e obras por período.
  - Os dados devem ser atualizados em tempo real ou com atualização automática periódica.
- **Nota de Implementação:** A funcionalidade foi implementada com um sistema de permissões (Roles) no token JWT e um `AdminGuard` dedicado no frontend e backend.

5.3 Considerações

Durante o ciclo de desenvolvimento iterativo, algumas dessas HUs foram **adaptadas** e **funcionalidades não previstas inicialmente foram adicionadas** para atender a desafios técnicos e melhorar o fluxo do produto. Por exemplo, a **HU05** foi modificada para sempre utilizar o IPFS, eliminando a escolha do usuário para garantir a robustez da prova. Além disso, foram adicionadas integrações complexas não detalhadas nas HUs originais, como um **gateway de pagamento** (Mercado Pago) e um **armazenamento temporário** (Cloudinary), que se mostraram essenciais para viabilizar o fluxo de registro assíncrono. As HUs **HU08** e **HU09**, focadas no Validador Público, foram despriorizadas do MVP para focar na entrega do fluxo principal do Criador e do Administrador.

6 PROJETO DO SISTEMA

Com base na análise do sistema detalhada no capítulo anterior, este capítulo descreve o projeto técnico do WriteSafe. O objetivo é traduzir as histórias de usuário e as necessidades dos perfis de Criador, Validador Público(descontinuado) e Administrador em uma estrutura tangível, abordando desde a concepção visual da interface até a arquitetura de dados relacional que sustentará o sistema.

6.1 Identidade Visual

O sistema WriteSafe foi desenvolvido para proporcionar uma navegação fluida entre as funcionalidades. As telas serão implementadas com atenção especial à usabilidade e ao apelo visual, utilizando uma paleta de cores moderna e condizente com o propósito do sistema.

6.1.1 Paleta de Cores

A paleta de cores escolhida para o WriteSafe foi cuidadosamente definida para transmitir confiança, criatividade e conveniência. Os principais tons são:

- **Background do sistema**: Branco para tema light e Azul-marinho para tema dark (despriorizado).
- **Texto e Ícones**: Azul Claro e Cinza.
- **Botões**: Azul, Preto e Cinza.
- **Alertas ou Notificações**: Vermelho (despriorizado).
- **Fontes**: Branco, Preto e Azul.

Essas escolhas asseguram uma identidade visual coesa e profissional, proporcionando uma experiência agradável e eficiente aos usuários.

6.1.2 Tipografia

A tipografia escolhida para o WriteSafe é a fonte **Roboto**, reconhecida por sua modernidade, legibilidade e versatilidade. Essa escolha reforça o caráter profissional e acessível da plataforma, garantindo uma experiência visual clara e agradável para os usuários em diferentes dispositivos.

6.1.3 Logotipo

O logotipo do WriteSafe está apresentado na Figura 4. Ele foi criado com a intenção de transmitir segurança ao criador, desde sua construção até as cores. É Apresentado um cadeado com forma de escudo para refletir a proteção e defesa da aplicação, e, além disso, uma caneta de pena para lembrar a criação de obras.

Figura 4 – Logotipo do WriteSafe
Autoria própria (2025).

6.2 Planejamento da Sprint 0 e da Sprint 1

O planejamento inicial do projeto WriteSafe está dividido em duas etapas principais: a Sprint 0, foi dedicada à preparação do ambiente de desenvolvimento, e a Sprint 1, focada

em entregar a funcionalidade inicial de criação e autenticação de contas, conforme descrito na História de Usuário HU01.

6.2.1 Preparação de Ambiente (Sprint 0)

A Sprint 0, com duração estimada de 5 dias, será destinada à configuração do ambiente de desenvolvimento. Essa etapa, embora não entregue valor direto ao cliente, é essencial para garantir a produtividade e organização das próximas sprints. Durante a Sprint 0, serão realizadas as seguintes atividades:

- Configuração do ambiente local de desenvolvimento, incluindo a instalação de ferramentas como editores de código, frameworks e bibliotecas necessárias.
- Configuração do sistema de versionamento utilizando Git e integração com um repositório remoto (ex.: GitHub).
- Definição da arquitetura inicial do banco de dados.
- Configuração de sistemas de integração contínua e automação de testes.

Ao final da Sprint 0, o ambiente estava preparado para suportar o desenvolvimento ágil e colaborativo.

6.2.2 Criação de Conta e Login (Sprint 1)

A Sprint 1, com duração de duas semanas, foi a primeira a entregar valor direto ao cliente. O objetivo principal dessa sprint é implementar a funcionalidade de cadastro e login, com foco na segurança e usabilidade, conforme descrito na História de Usuário HU01. As principais atividades planejadas foram:

- Implementação da funcionalidade de criação de contas, incluindo validação de dados e armazenamento seguro das informações do usuário.
- Configuração de autenticação utilizando JWT (JSON Web Tokens), garantindo que os usuários tenham sessões seguras.
- Implementação da lógica de hash de senhas para armazená-las de forma segura no banco de dados.
- Desenvolvimento da interface de login e cadastro, garantindo uma experiência intuitiva para o usuário.
- Testes de integração para verificar a funcionalidade de cadastro e login.

Ao final da Sprint 1, entregou-se uma funcionalidade demonstrável e funcional que permitirá aos usuários criar contas e acessar o sistema com segurança.

6.3 Prototipação de Telas

Para a etapa de prototipação das interfaces, foi utilizada a ferramenta de design colaborativo **Figma**.

6.3.1 Telas da Primeira Sprint

As telas de login e criação de conta são elementos fundamentais do sistema WriteSafe, sendo responsáveis por garantir o acesso seguro e personalizado para cada usuário. Essas telas têm como objetivo proporcionar uma experiência de usuário clara e intuitiva, priorizando a segurança e a usabilidade.

A tela de criação de conta permite ao usuário registrar suas informações essenciais, como nome, e-mail e senha. Já a tela de login é a porta de entrada para o sistema, onde os usuários, após a validação de suas credenciais, terão acesso às funcionalidades específicas do seu perfil.

Essas interfaces foram inicialmente projetadas com base nos princípios de design responsivo, garantindo uma experiência consistente em diferentes dispositivos. As Figuras 5 e 6 apresentam os protótipos dessas telas.

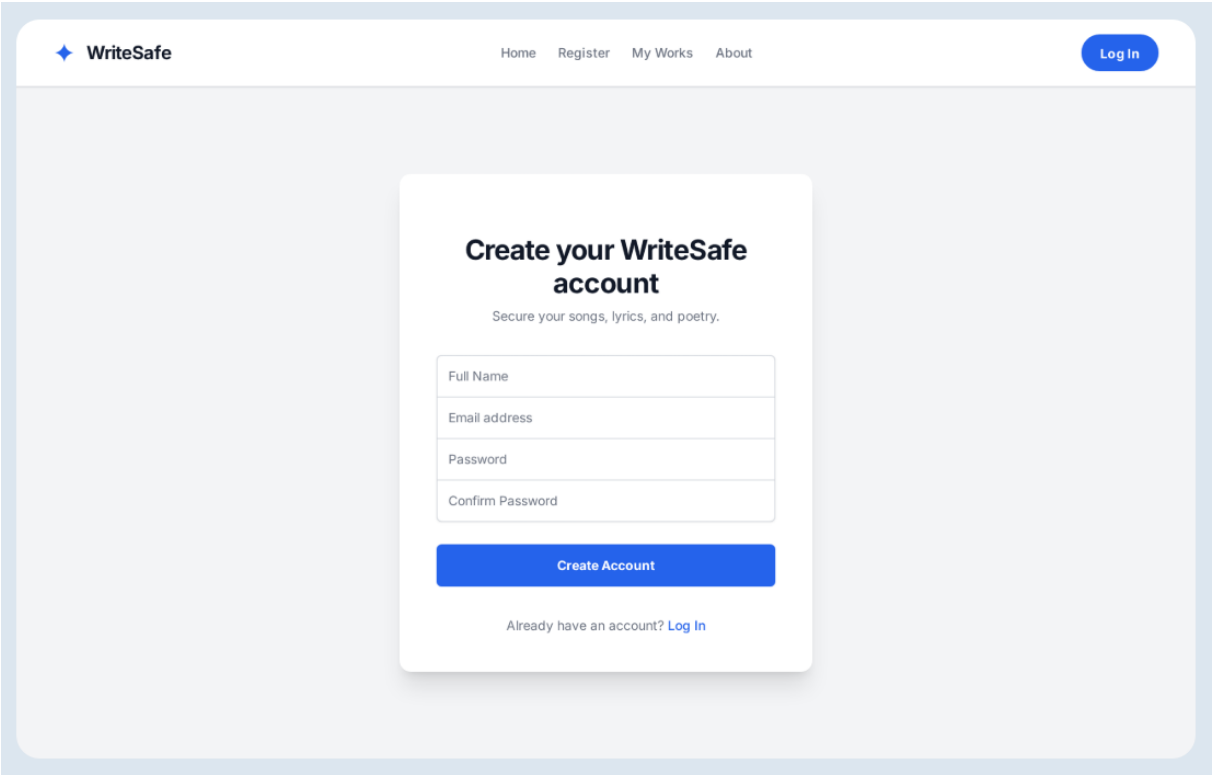

Figura 5 – Tela de criação de conta
Autoria própria (2025).

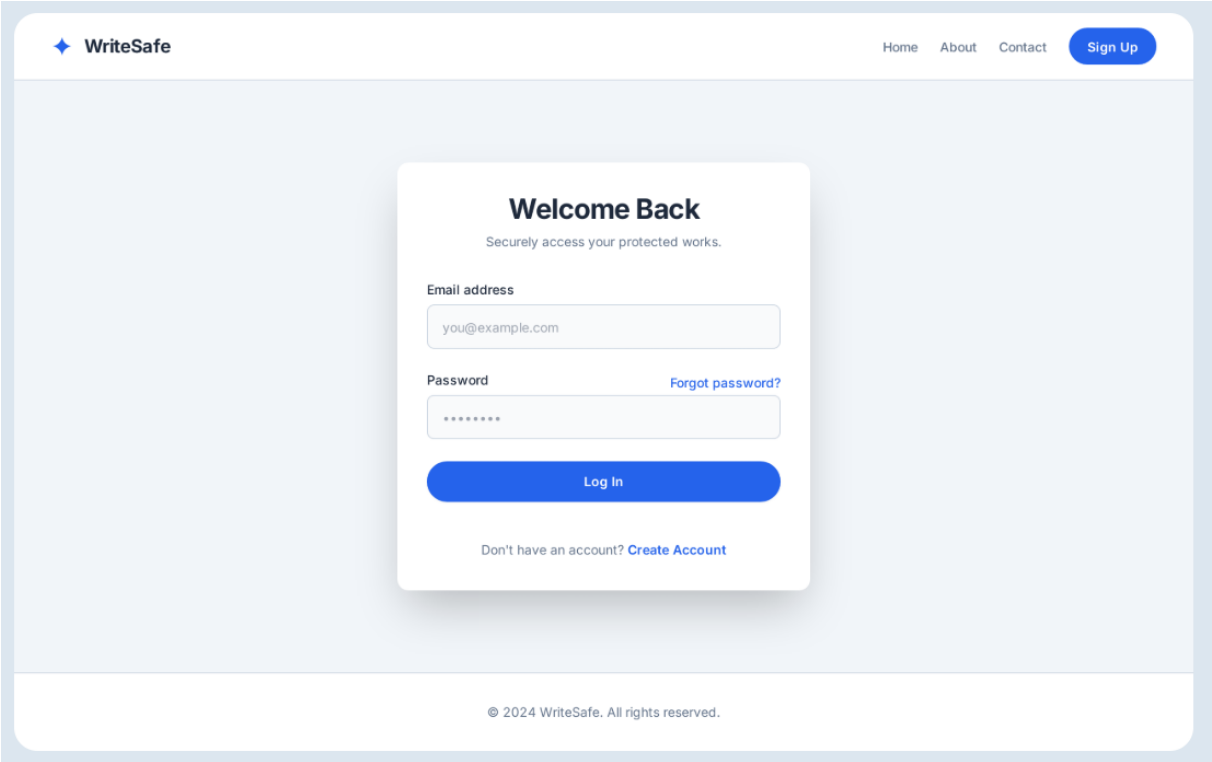

Figura 6 – Tela de login
Autoria própria (2025).

6.3.2 Telas Principais das Sprints Futuras

As telas apresentadas nesta seção foram elaboradas com o intuito de refletir as funcionalidades previstas nas Sprints futuras. Elas representam o planejamento inicial, mas é importante ressaltar que essas telas sofreram ajustes ao longo do desenvolvimento, de acordo com o feedback do cliente e a evolução das necessidades do projeto.

A primeira interação do criador com o sistema ocorre na tela de apresentação da aplicação, conforme ilustrado na Figura 7. Nessa tela, o criador conhece um pouco do sistema e seus benefícios. Na mesma, há a chamada para login ou criação de conta.

Figura 7 – Tela de onBoarding
Autoria própria (2025).

Após o login, o criador é direcionado para o ambiente principal, onde pode registrar suas criações por meio da tela de upload de arquivos, ilustrada na Figura 8. Essa tela permite que o criador envie documentos para registro, garantindo que os metadados essenciais, como título, tipo de obra e descrição, sejam preenchidos.

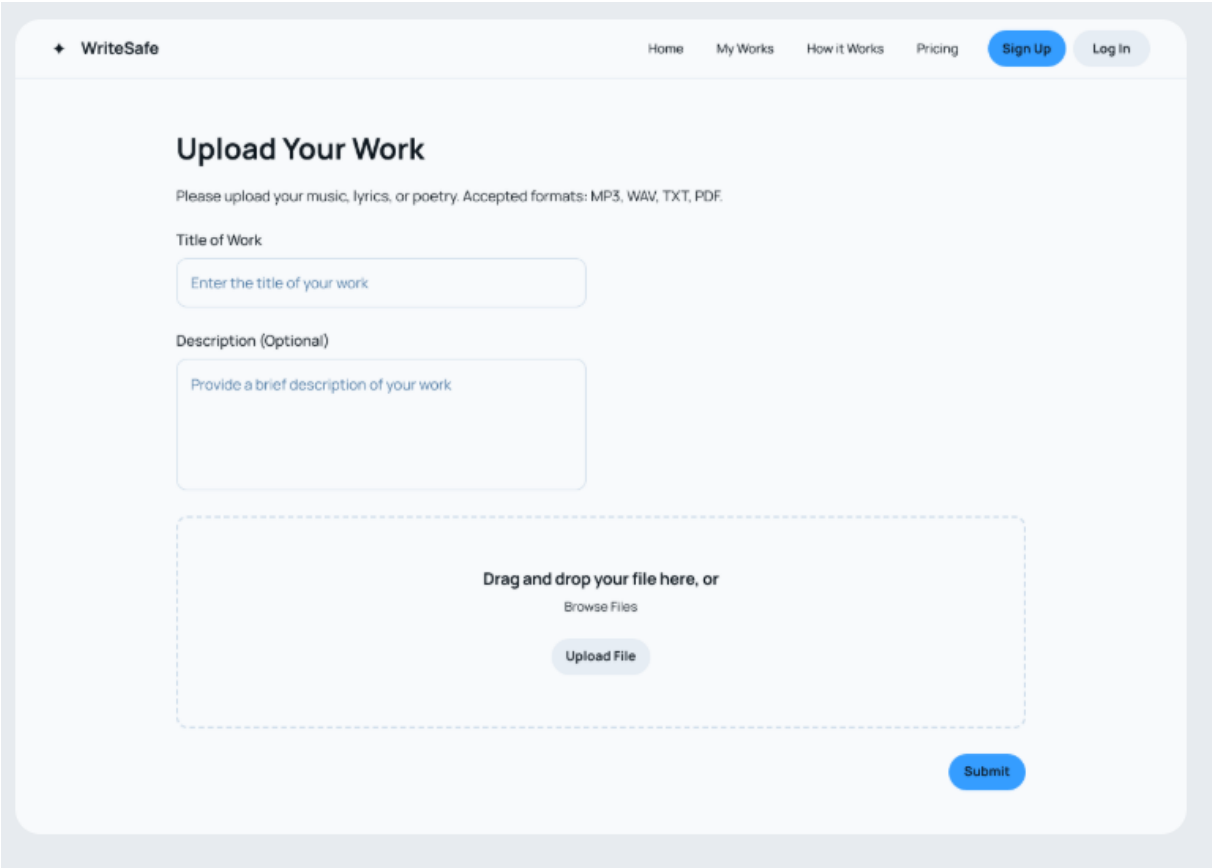

Figura 8 – Tela de upload de arquivos (em inglês)
Autoria própria (2025).

As principais características dessa tela incluem:

- **Campos de Metadados**: Entrada estruturada para informações sobre a obra.
- **Design Minimalista**: Foco em funcionalidade e clareza.

A tela de listagem de registros, exibida na Figura 9, apresenta ao criador uma visão organizada de todas as obras enviadas, incluindo status de validação, data de envio e opções de download. Essa tela foi projetada para facilitar o acompanhamento e o gerenciamento dos registros.

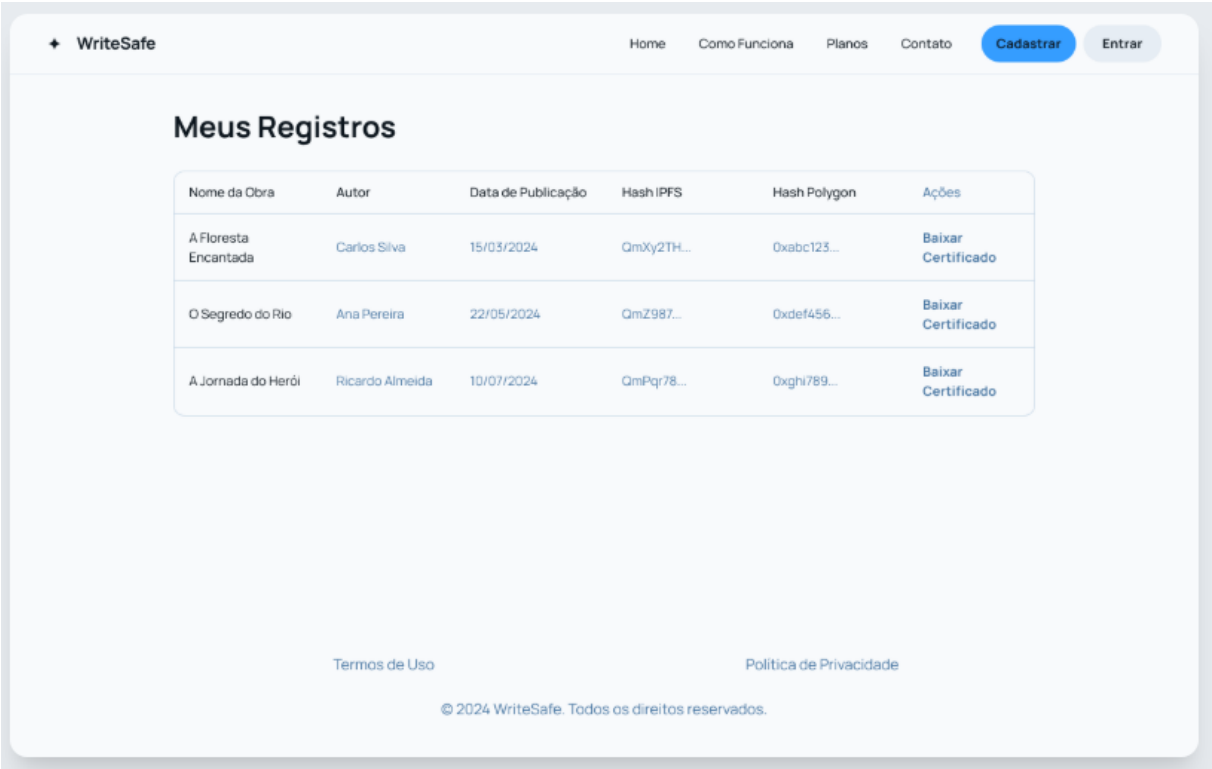

Figura 9 – Tela de listagem
Autoria própria (2025).

Os elementos chave dessa tela incluem:

- **Tabela de Registros**: Organiza os dados em colunas como título, status e ações.
- **Filtros Avançados**: Permitem buscar registros específicos com base em palavras-chave, datas ou tipos de obra.
- **Ação Contextual**: Botões para realizar downloads ou visualizar detalhes de cada registro.

6.4 Modelagem de Dados

A modelagem de dados do **WriteSafe** foi guiada por um objetivo claro: oferecer, em um fluxo único, uma prova de autoria que seja **imutável, pública e datada**, sem prejudicar a agilidade de uma plataforma web moderna. Três decisões de arquitetura nortearam este desenho:

1. **Isolar a cadeia de custódia:** Todos os elementos que comprovam a autoria (hashes criptográficos, CIDs do IPFS e o código da transação em blockchain) foram centralizados na tabela `obras` e não podem ser alterados após o registro.
2. **Separar a camada de experiência:** Arquivos destinados à visualização rápida (como o arquivo original) foram desacoplados da prova. Foi utilizada a plataforma **Cloudi-**

nary como um sistema de armazenamento temporário seguro, garantindo o acesso ao arquivo durante o processamento assíncrono do pagamento.

3. **Garantir escalabilidade:** O esquema foi desenhado para ser extensível, com o uso de chaves primárias do tipo **UUID** para garantir unicidade global.

A Figura 10 apresenta o diagrama de banco de dados planejado inicialmente. A tabela `obras` concentra a evidência principal. Cada registro armazena:

- o `hash_obra_zip`, um hash SHA-256 do arquivo original enviado pelo autor;
- o `cid_ipfs`, que mantém o pacote na rede descentralizada via Pinata;
- o `id_transacao`, retornado pela Polygon quando o Smart Contract grava o registro na blockchain.

O campo `status` (ex: *"AGUARDANDO_PAGAMENTO", "COMPLETO", "ERRO"*) foi essencial para gerenciar o fluxo assíncrono de pagamento e registro.

Para a experiência do usuário, a tabela `arquivos` armazena a URL do arquivo no **Cloudinary** (`url_cloudinary`), permitindo que o sistema finalizasse o registro (enviando para o IPFS e Polygon) somente após a confirmação do pagamento via webhook do Mercado Pago.

A autenticação é tratada na tabela `usuarios`, onde cada conta guarda e-mail único, senha cifrada com *bcrypt* e referencia um registro em `perfis` (ex: Criador, Admin). A tabela `tipos_obra` funciona como um dicionário normalizado (Música, Letra, Partitura, etc.), permitindo que o frontend exiba opções dinâmicas para o usuário.

Quando a obra alcança o status "COMPLETO", o sistema fica apto a gerar um certificado em PDF. A tabela `certificados`, planejada para armazenar informações deste PDF, e a tabela `consultas_validacao`, destinada ao perfil de Validador Público, **não foram implementadas neste MVP**. A geração do certificado foi realizada de forma dinâmica (*on-the-fly*) pelo backend, e as funcionalidades de validação pública (HU08, HU09) foram despriorizadas para focar no fluxo essencial do criador.

A Figura 11 ilustra o esquema de banco de dados final implementado para o MVP, refletindo esta simplificação. Em síntese, o esquema atendeu simultaneamente aos requisitos de prova de autoria, performance de mídia e ao fluxo de pagamento assíncrono, oferecendo uma base sólida para o **WriteSafe**.

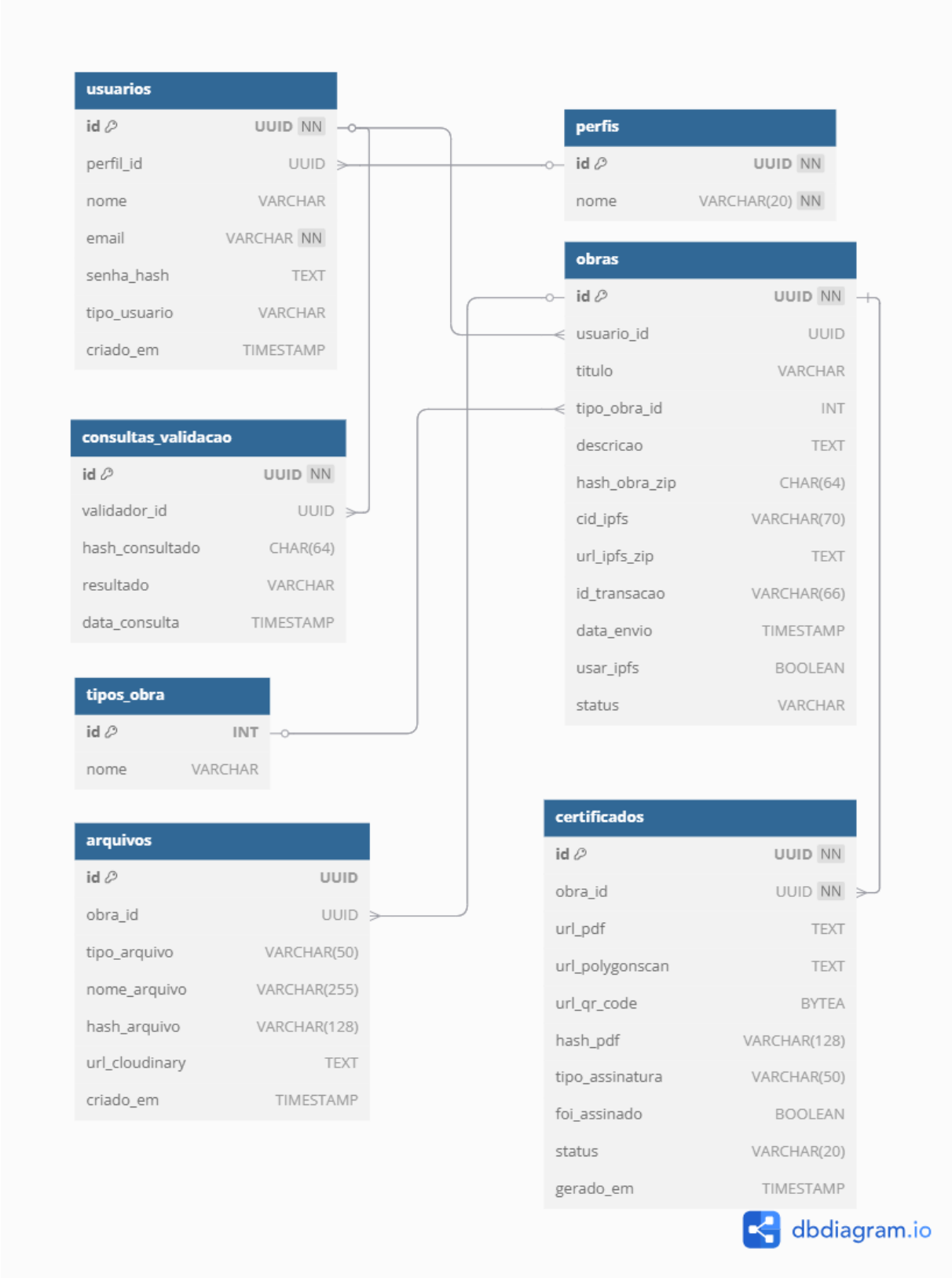

Figura 10 – Modelagem inicial do banco de dados da aplicação

Autoria própria (2025).

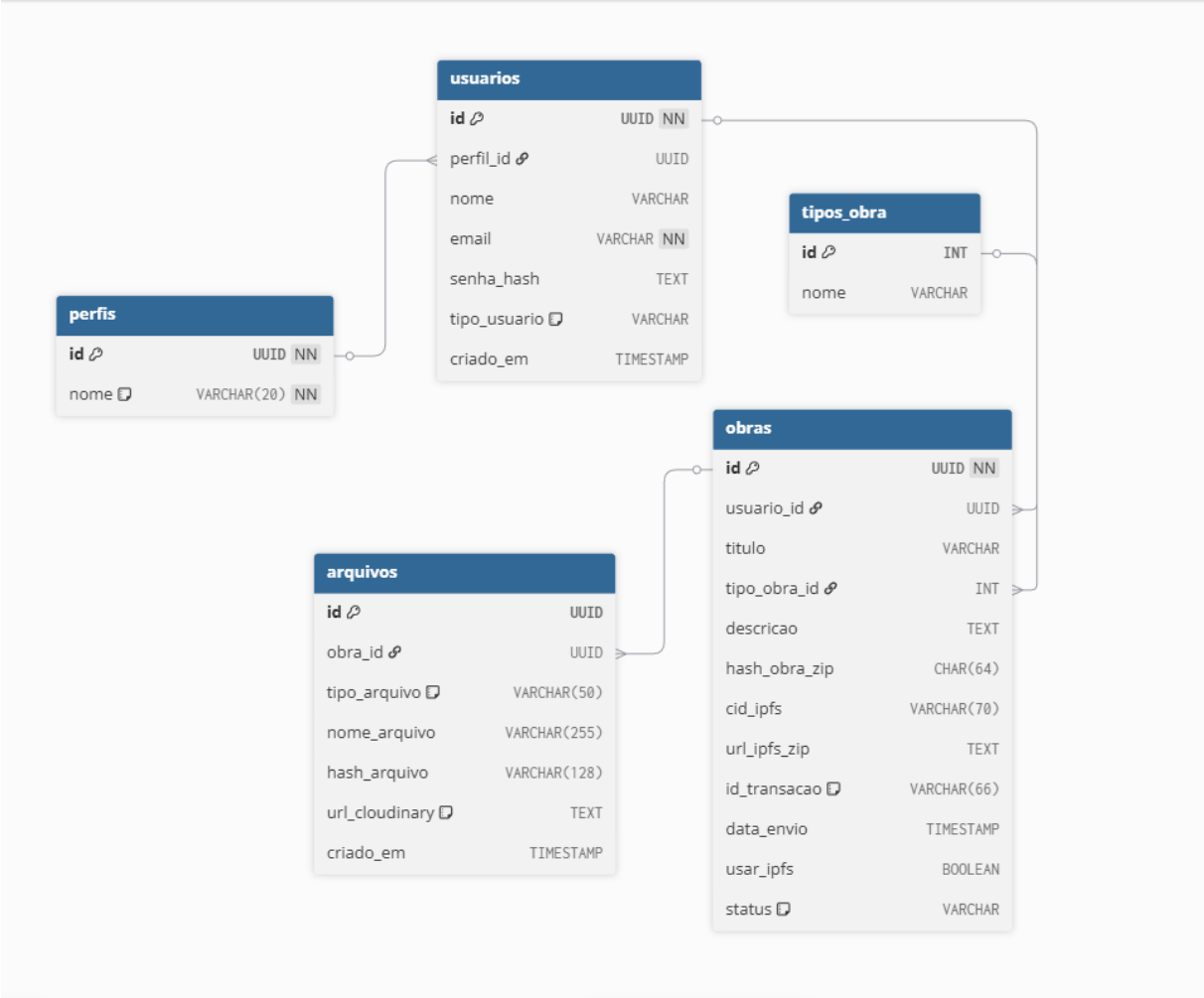

Figura 11 – Modelagem de dados final implementada no MVP

Autoria própria (2025).

7 DESENVOLVIMENTO DO SISTEMA

Este capítulo documenta o processo de desenvolvimento prático do sistema **WriteSafe**, traduzindo os requisitos de análise e as especificações de projeto, apresentados nos capítulos anteriores, em uma aplicação funcional. A abordagem de desenvolvimento foi iterativa e incremental, inspirada nos princípios do framework Scrum, permitindo a entrega de valor contínua e a adaptação a desafios técnicos descobertos durante a implementação.

A estrutura deste capítulo segue a cronologia do projeto, detalhando as principais tecnologias empregadas, a execução de cada ciclo de desenvolvimento, as funcionalidades implementadas, desde a autenticação de usuários até orquestração do registro em blockchain, e, por fim, uma análise transparente dos desafios técnicos encontrados e das soluções adotadas.

7.1 Sprint 0: Fundação do Projeto e Estruturação do Ambiente de Desenvolvimento

O objetivo primordial desta primeira sprint não foi a implementação de funcionalidades visíveis ao usuário, mas sim a construção dos alicerces técnicos e da infraestrutura de desenvolvimento do projeto WriteSafe. O foco foi estabelecer um ambiente de trabalho robusto, escalável e organizado, que permitisse um desenvolvimento ágil e eficiente nas sprints subsequentes.

A Sprint 0 teve um caráter puramente técnico e fundacional. Como tal, **nenhuma História de Usuário foi diretamente implementada nesta fase**. As atividades realizadas, no entanto, serviram como pré-requisitos essenciais que viabilizaram o desenvolvimento de todas as HUs a partir da sprint seguinte, especialmente ao definir a arquitetura que suportaria a **HU01 (Criação de Conta)** e a **HU10 (Painel Administrativo)**.

A estratégia de desenvolvimento foi definida com foco em agilidade e controle. Optou-se por uma arquitetura **monorepo**, centralizando os códigos do frontend e do backend em um único repositório **Git** para facilitar o gerenciamento de dependências e garantir a consistência do projeto.

Para o ambiente de desenvolvimento local, a decisão foi rodar os serviços da aplicação (frontend e backend) **manualmente** em seus próprios terminais. Essa abordagem foi escolhida para proporcionar maior controle sobre cada serviço individualmente, simplificando o processo de depuração (*debugging*) e permitindo reinicializações rápidas.

O banco de dados foi isolado do sistema operacional principal através do uso de **Docker** para a instanciação de um contêiner com o **MySQL**, garantindo um ambiente de banco de dados consistente e portátil.

As tecnologias e ferramentas utilizadas nesta sprint estão listadas a seguir:

- **Versionamento:** Git.
- **Frontend:** Angular (com Angular CLI).

- **Backend:** NestJS (com Nest CLI).
- **Banco de Dados:** MySQL (versão 5.7), rodando em Docker.
- **ORM (Object-Relational Mapping):** Prisma.

7.2 Sprint 1: UI/UX Inicial e Identidade Visual

Embora o planejamento inicial (apresentado na Seção 7.2) previsse a implementação das funcionalidades de cadastro e login, durante o desenvolvimento a equipe identificou a necessidade de priorizar a definição da identidade visual e da interface base da aplicação. Assim, o foco da Sprint 1 foi redirecionado para o desenvolvimento da camada de UI/UX inicial e dos componentes visuais que sustentariam as próximas entregas, ou seja, fazer a estrutura de navegação principal e garantir uma aparência coesa e profissional para a aplicação antes de implementar as funcionalidades interativas.

Embora nenhuma História de Usuário tenha sido completamente finalizada nesta fase, o trabalho realizado foi um pré-requisito fundamental para a camada de apresentação de quase todas as HUs. Especificamente, as atividades desta sprint construíram o ambiente visual onde a **HU01 (Criação de Conta)** e as HUs de login seriam implementadas. Adicionalmente, o desenvolvimento do header criou o principal mecanismo de navegação que seria utilizado para acessar as funcionalidades descritas em **HU02**, **HU06**, **HU07** e **HU10**.

A estratégia adotada foi a de construir a "casca"da aplicação primeiro, focando na experiência do usuário não autenticado. Para a estilização, foi escolhido o framework **Tailwind CSS** por sua abordagem *utility-first*, que permite a criação de designs customizados e responsivos de forma rápida e consistente. O desenvolvimento dos componentes seguiu o padrão *standalone* do Angular, promovendo a modularidade.

A maior complexidade desta sprint foi a implementação de um header dinâmico e responsivo, que deveria se adaptar a diferentes tamanhos de tela e, futuramente, ao estado de autenticação do usuário.

As tecnologias e ferramentas utilizadas nesta sprint estão listadas a seguir:

- **Frontend:** Angular.
- **Estilização:** Tailwind CSS, incluindo a biblioteca de componentes Flowbite para elementos de formulário.

As principais atividades realizadas durante esta sprint são apresentadas a seguir:

1. **Configuração do Framework de Estilização:** O **Tailwind CSS** foi instalado e configurado no projeto Angular. O processo exigiu uma configuração manual, incluindo a criação dos arquivos `tailwind.config.js` e `postcss.config.js`, e a resolução de conflitos de versão para garantir a compatibilidade com o Angular Build.

2. **Desenvolvimento das Páginas Públicas:** Foram criados os componentes para as páginas estáticas que formam a apresentação pública do WriteSafe:

    - **Onboarding:** A página inicial da aplicação, projetada para explicar o propósito do sistema e guiar o usuário para as ações de login ou registro.
    - **Sobre:** Página com a descrição e missão do projeto conforme Figura 13.
    - **Contato:** Página com informações para contato de acordo com Figura 14.

3. **Implementação do Componente de Header Responsivo:** Foi desenvolvido um componente de header (`<app-header>`) reutilizável, contendo a logo do projeto, o nome "WriteSafe"e os links de navegação (Figura 12). A principal funcionalidade implementada foi a responsividade: em telas maiores, o menu é exibido horizontalmente; em telas menores, ele se transforma em um menu "hambúrguer"vertical, garantindo uma boa usabilidade em dispositivos móveis.

Ao final da Sprint 1, o projeto WriteSafe deixou de ser apenas uma estrutura de backend e passou a ter uma identidade visual e uma interface navegável. A aplicação agora contava com uma aparência profissional, um layout responsivo e as páginas essenciais para apresentar o produto a um novo usuário. A base visual estava pronta para receber os formulários e as funcionalidades dinâmicas das próximas sprints.

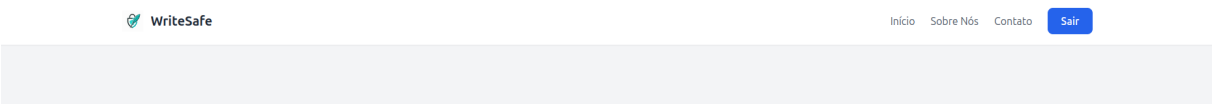

Figura 12 – Cabeçalho do WriteSafe

Figura 13 – Sobre nós

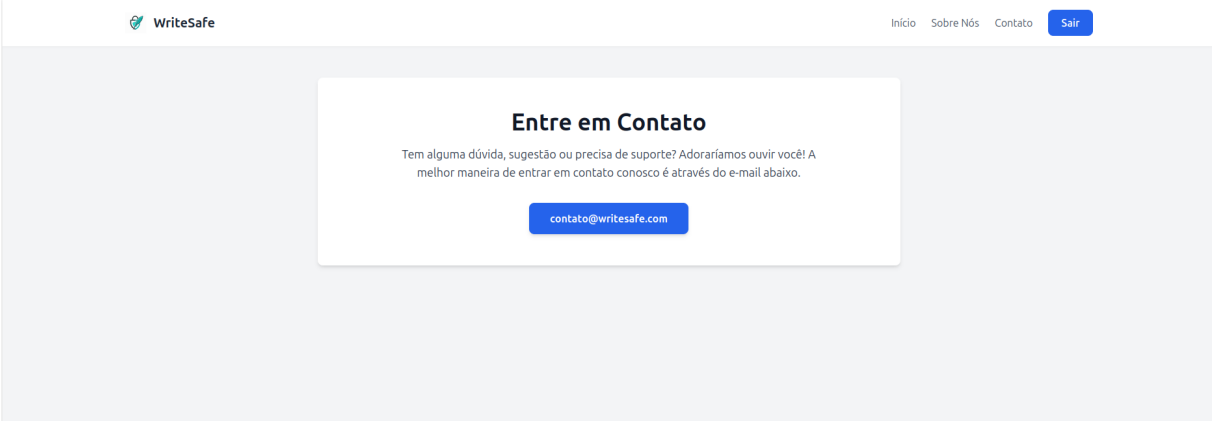

Figura 14 – Contato

7.3 Sprint 2: Backend de Autenticação e Usuários (HU01)

O objetivo desta sprint foi implementar o núcleo do sistema de autenticação e gerenciamento de usuários no backend. Com a interface visual básica já estabelecida, o foco foi criar a lógica de negócio que permitiria a um novo criador se cadastrar na plataforma e, posteriormente, autenticar-se para acessar áreas restritas.

Esta sprint foi diretamente focada na implementação da **HU01 (Criação de Conta)**. Os critérios de aceitação foram atendidos no backend da seguinte forma:

- **Formulário de cadastro:** Foi criado o endpoint `POST /users` para receber os dados do formulário.
- **Validação de e-mail e senha:** A lógica de serviço foi implementada para verificar se um e-mail já existe no banco de dados. Foi utilizado o pacote `class-validator` nos DTOs (Data Transfer Objects) para garantir que os dados recebidos, como e-mail e senha, atendessem aos critérios mínimos de formato e segurança (ex: tamanho mínimo da senha).
- **Segurança da senha:** A senha do usuário nunca é armazenada em texto plano. Foi integrada a biblioteca **bcrypt** para gerar um *hash* seguro da senha antes de persisti-la no banco de dados.

É importante notar que os critérios de envio e confirmação de e-mail foram adiados para futuras iterações, a fim de manter o foco do MVP na funcionalidade principal de registro. Além da HU01, esta sprint também construiu a base para o fluxo de login, que seria finalizado na sprint seguinte.

A estratégia foi seguir a arquitetura modular do **NestJS**, separando as responsabilidades de forma clara. Foram criados dois módulos principais: o `UsuariosModule`, responsável pelo CRUD (Create, Read, Update, Delete) de usuários, e o `AuthModule`, dedicado à lógica de autenticação.

A segurança foi uma prioridade desde o início, com a decisão de implementar o *hashing* de senhas com **bcrypt**. Para o gerenciamento de sessões, optou-se pelo padrão **JSON Web Token (JWT)**, uma abordagem moderna e sem estado (*stateless*) que facilita a comunicação segura entre o frontend e o backend.

As tecnologias e ferramentas utilizadas nesta sprint estão listadas a seguir:

- **Backend:** NestJS.
- **ORM:** Prisma, para todas as interações com o banco de dados MySQL.
- **Segurança de Senhas:** `bcrypt`.
- **Autenticação:** `@nestjs/jwt` e `@nestjs/passport` para a geração e, futuramente, validação de tokens JWT.
- **Validação de Dados:** `class-validator` e `class-transformer` para a validação de DTOs.

As principais atividades realizadas durante esta sprint são apresentadas a seguir:

1. **Geração dos Módulos:** Utilizando o Nest CLI, foram geradas as estruturas completas (*module, controller, service*) para as funcionalidades de `Usuarios` e `Auth`.
2. **Implementação do Cadastro (HU01):** Foi desenvolvido o endpoint `POST /users`. A lógica no `UsuariosService` foi criada para:
    - Receber os dados do usuário através do `CreateUsuarioDto`.
    - Verificar a unicidade do e-mail no banco de dados.
    - Gerar o *hash* da senha com `bcrypt`.
    - Salvar o novo usuário no banco de dados utilizando o **Prisma Client**.
3. **Implementação do Login:** Foi desenvolvido o endpoint `POST /auth/login`. A lógica no `AuthService` foi criada para:
    - Validar as credenciais recebidas (e-mail e senha).
    - Comparar a senha enviada com o *hash* armazenado no banco usando `bcrypt.compare`.
    - Se as credenciais fossem válidas, gerar um **Token JWT** contendo as informações essenciais do usuário (ID, e-mail, nome).

Ao final da Sprint 2, a aplicação deixou de ser apenas uma interface estática. O backend agora possuía uma API funcional para gerenciamento de usuários e autenticação, com endpoints prontos para serem consumidos. A base para um sistema seguro estava estabelecida, com a API preparada para receber os dados dos formulários de registro e login que seriam implementados no frontend na próxima sprint.

7.4 Sprint 3: Fluxo de Autenticação Completo e Segurança

Com os endpoints de backend para autenticação já criados, o objetivo principal desta sprint foi conectar a interface do usuário a essa lógica, tornando as funcionalidades de cadastro e login totalmente operacionais. Além disso, a meta era implementar a camada de segurança que separa as áreas públicas das áreas privadas da aplicação, garantindo que apenas usuários autenticados pudessem acessar funcionalidades restritas.

Esta sprint finalizou a implementação da **HU01 (Criação de Conta)**, ao conectar o formulário de registro do frontend ao endpoint do backend. Mais importante, ela implementou o cerne do benefício da HU01, que é *"acessar a plataforma de forma segura"*.

O trabalho realizado também foi um pré-requisito fundamental para todas as histórias de usuário subsequentes que dependem de um usuário autenticado, como **HU02 (Envio de Obra)**, **HU06 (Certificado Digital)**, **HU07 (Validação Pública)** e **HU10 (Painel Administrativo)**.

No **frontend (Angular)**, foi criado um serviço dedicado (`AuthService`) para centralizar todas as chamadas HTTP relacionadas à autenticação. Os formulários de login e registro foram implementados com **ReactiveFormsModule**, permitindo a captura e validação dos dados do usuário. O fluxo de autenticação do cliente foi definido da seguinte forma: o usuário submete o formulário, o `AuthService` envia os dados para a API, e em caso de sucesso, o Token JWT retornado é armazenado no `localStorage` do navegador, efetivando a sessão do usuário. Para proteger as rotas, foi criado um **AuthGuard**.

No **backend (NestJS)**, a estratégia foi ativar a segurança planejada na sprint anterior. Foi implementada uma **JwtStrategy**, um componente do **Passport.js** responsável por extrair o token JWT do cabeçalho de autorização, validar sua assinatura com a chave secreta e retornar os dados do usuário. Este mecanismo permitiu a proteção de rotas através do decorador `@UseGuards(AuthGuard('jwt'))`.

As tecnologias e ferramentas utilizadas nesta sprint estão listadas a seguir:

- **Frontend:** Angular, ReactiveFormsModule, HttpClient.
- **Backend:** NestJS, `@nestjs/jwt`, `@nestjs/passport`, `passport-jwt`.

As principais atividades realizadas durante esta sprint são apresentadas a seguir:

1. **Implementação da Lógica de Frontend:** Foi criado o `AuthService` no Angular para encapsular as chamadas aos endpoints `/users` e `/auth/login`. Os componentes de **Login**, ilustrado na Figura 17, e **Registro**, apresentado na Figura 16, foram atualizados para utilizar este serviço, enviando os dados dos formulários e tratando as respostas de sucesso (armazenando o token e redirecionando) e de erro (exibindo um pop-up).

2. **Gerenciamento de Token no Cliente:** Foi implementada a lógica para armazenar o `access_token` JWT no `localStorage` do navegador após um login bem-sucedido e para removê-lo durante o logout.

3. **Criação do Guarda de Rota (Frontend):** Foi desenvolvido o `authGuard` no Angular. Sua lógica verifica a existência do token no `localStorage`. Caso o token não exista, o acesso à rota protegida é bloqueado e o usuário é redirecionado para a página de login. Foi necessário ajustar este guarda para ser compatível com a Renderização no Servidor (SSR), evitando chamadas ao `localStorage` no ambiente do servidor.

4. **Implementação da Estratégia de Autenticação (Backend):** Foi criada a classe `JwtStrategy` no `AuthModule`. Esta estratégia foi configurada para extrair o token do cabeçalho, validá-lo com a `JWT_SECRET` definida no arquivo `.env` e buscar o usuário no banco de dados para anexá-lo ao objeto da requisição.

5. **Proteção dos Endpoints:** As rotas que exigem autenticação, como as futuras rotas de obras e estatísticas, foram protegidas no backend utilizando o decorador `@UseGuards(AuthGuard('jwt'))`.

Ao final da Sprint 3, o WriteSafe evoluiu de um site com páginas estáticas para uma aplicação web segura e funcional. O ciclo de autenticação completo — cadastro, login, proteção de rotas e logout — estava totalmente implementado, como visto na Figura 15 (Homepage) e nas telas de autenticação. Havia uma clara separação entre as áreas públicas e privadas, e a base de segurança para todas as futuras funcionalidades do usuário logado estava estabelecida e validada.

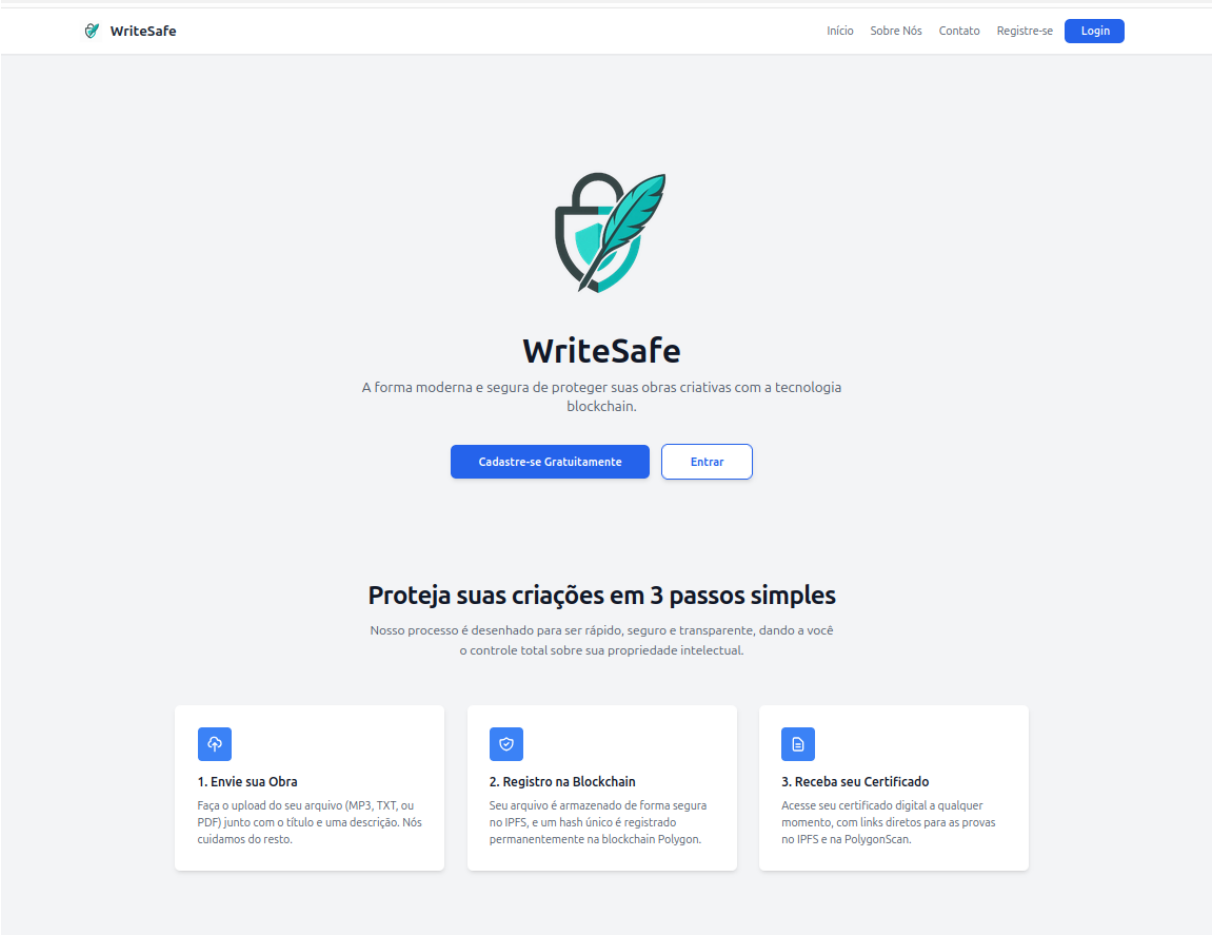

Figura 15 – Homepage do WriteSafe

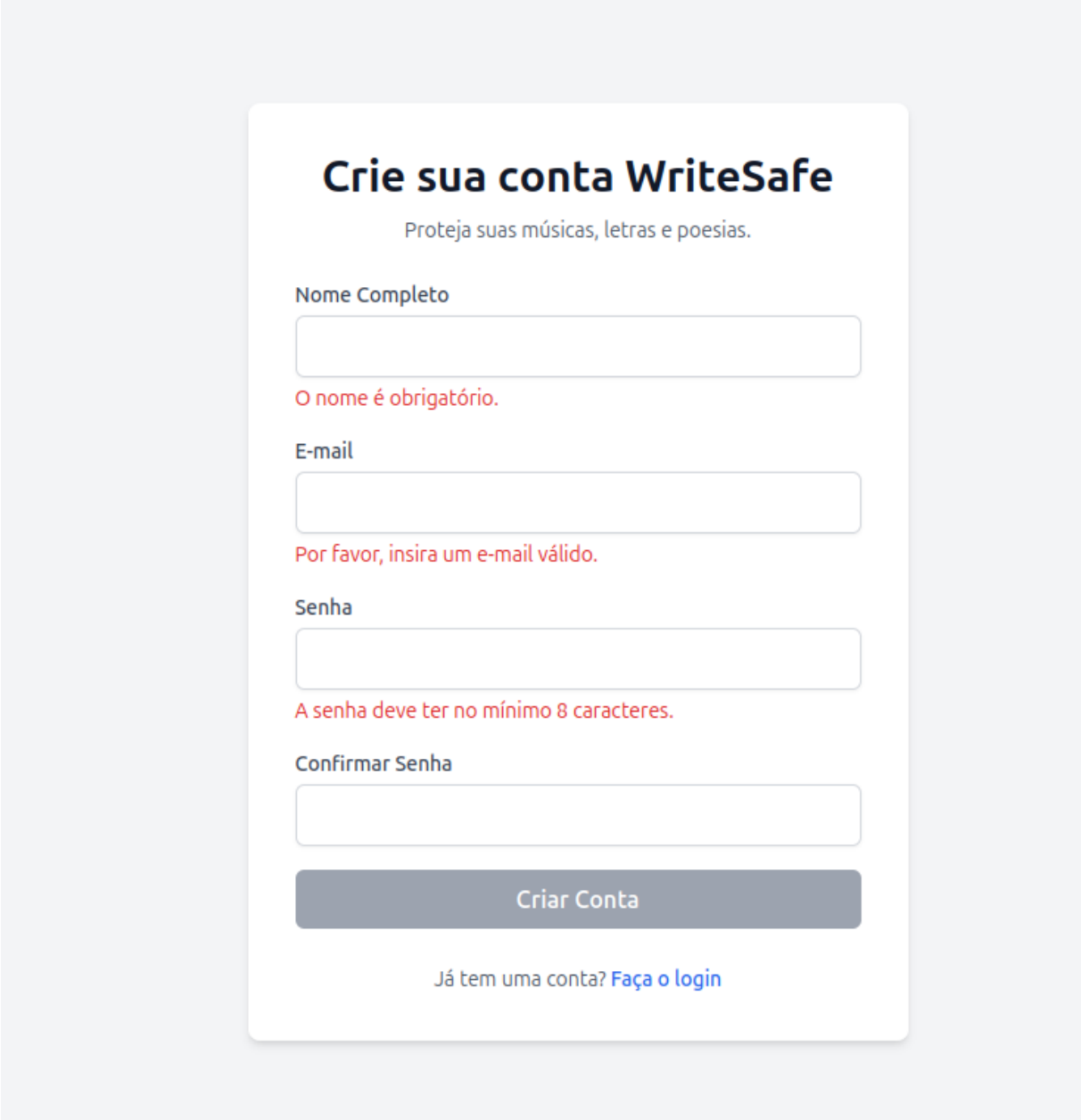

Figura 16 – Página de criação de conta do WriteSafe

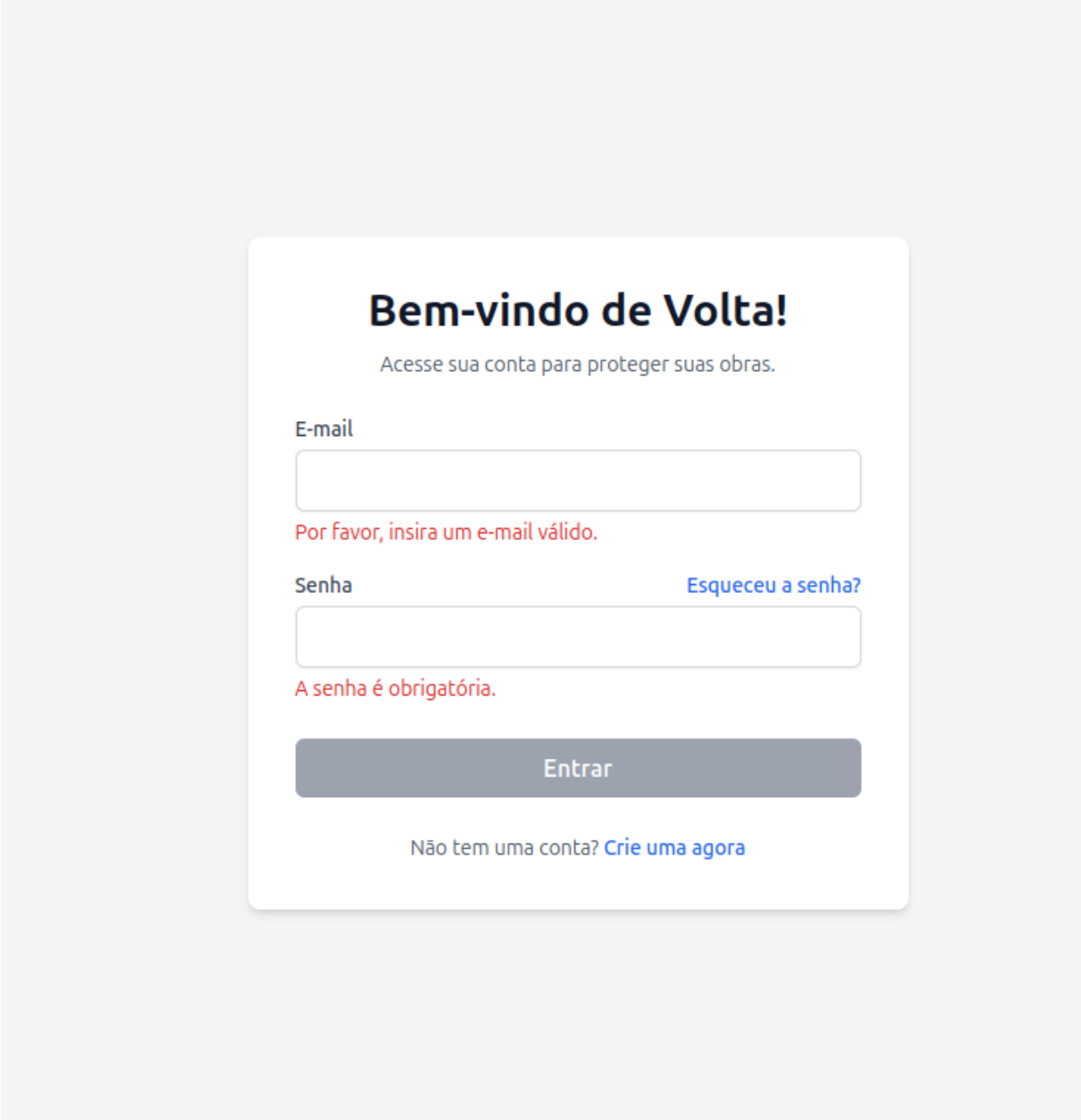

Figura 17 – Página de login do WriteSafe

7.5 Sprint 4: Frontend do Registro de Obra (HU02)

Com os sistemas de autenticação e navegação estabelecidos, o objetivo desta sprint foi desenvolver a interface de usuário para a funcionalidade central do WriteSafe: o registro de uma nova obra. O foco foi criar uma experiência de usuário (UX) intuitiva e robusta para o envio de arquivos, metadados **e dados de pagamento**, preparando o terreno para o fluxo completo de registro.

Esta sprint foi dedicada à implementação completa do frontend para a **HU02 (Envio de Obra)**, incluindo a coleta segura das informações de pagamento necessárias para a futura orquestração do fluxo. Os critérios de aceitação foram atendidos da seguinte forma:

- **Escolha e envio de arquivos:** Foi implementada uma área de upload com suporte a clique e *drag-and-drop*.
- **Validação de formato e tamanho:** A lógica no frontend foi desenvolvida para validar arquivos (MP3, TXT, PDF) e aplicar o limite de 10 MB, com feedback via pop-up.
- **Solicitação de metadados:** Foram criados os campos para "Título", "Descrição"e o **dropdown dinâmico** para "Tipo de Obra".
- **Coleta Segura de Dados de Pagamento:** Foram integrados os campos seguros (*secure fields*) do **Mercado Pago** para a coleta dos dados do cartão de crédito (número, validade, CVC), além dos campos para nome do titular e CPF, necessários para transações no Brasil. A lógica de **tokenização** do cartão foi implementada no frontend, garantindo que os dados sensíveis nunca trafeguem pelo servidor da aplicação.
- **Geração de hash:** A interface que inicia o processo foi implementada.
- **Mensagem de sucesso/erro:** O componente de pop-up personalizado foi integrado para feedback.

O trabalho desta sprint, portanto, não apenas atendeu à HU02, mas também estabeleceu a fundação técnica para o fluxo de pagamento simulado que seria finalizado na Sprint 7.

A estratégia principal foi construir um formulário unificado e interativo utilizando o **ReactiveFormsModule** do Angular. Para a seção de pagamento, foi adotada a abordagem de **Checkout Transparente (Bricks)** do Mercado Pago, utilizando seu SDK JavaScript v2. Esta escolha permitiu a criação de campos de cartão de crédito seguros (renderizados em *iframes*) diretamente na página, proporcionando uma experiência de usuário fluida e garantindo a conformidade com as normas de segurança (PCI Compliance), pois os dados sensíveis são enviados diretamente do navegador do cliente para o Mercado Pago.

A lógica do **dropdown dinâmico** para o "Tipo de Obra"foi mantida, reagindo ao *mimetype* do arquivo e solicitando a entrada do usuário apenas para PDFs.

As tecnologias e ferramentas utilizadas nesta sprint estão listadas a seguir:

- **Frontend:** Angular, ReactiveFormsModule.
- **Estilização:** Tailwind CSS.
- **Pagamento:** Mercado Pago JS SDK v2.

As principais atividades realizadas durante esta sprint são apresentadas a seguir:

1. **Criação da Página "Registrar Nova Obra":** O componente foi gerado e a rota `/registrar-obra` foi adicionada e protegida.

2. **Desenvolvimento do Formulário Reativo:** O `FormGroup` foi atualizado para incluir não apenas os campos da obra (`titulo`, `descricao`, `tipoObraId`), mas também os campos necessários para a tokenização do cartão (`cardholderName`, `docNumber`).

3. **Implementação da Área de Upload:** O HTML e o TypeScript foram mantidos para suportar clique e *drag-and-drop*.

4. **Lógica do Dropdown Dinâmico:** O método `handleFile` foi mantido para controlar a visibilidade e as opções do dropdown com base no tipo de arquivo.

5. **Validação de Arquivo e Feedback de UX:** A validação de tamanho e tipo, junto ao pop-up personalizado, foi mantida.

6. **Integração dos Campos de Pagamento:**

    - O script do SDK do Mercado Pago foi adicionado ao `index.html`.
    - As `divs` com os IDs específicos (`form-checkout__-cardNumber-container`, etc.) foram adicionadas ao template HTML.
    - No `ngOnInit` do componente, o SDK foi inicializado com a *Public Key* de teste.
    - A lógica para criar e "montar"os campos seguros individuais (`cardNumber`, `expirationDate`, `securityCode`) foi implementada.
    - O método `onSubmit` foi modificado para, antes de chamar o backend, executar a função `mp.fields.createCardToken()`, passando os dados do titular e recebendo o token seguro do cartão.

Ao final da Sprint 4, a página "Registrar Nova Obra", conforme ilustrado na Figura 18, estava funcionalmente completa do ponto de vista do frontend. Ela não apenas capturava os dados da obra e validava o arquivo, mas também coletava os dados de pagamento de forma segura através da tokenização do Mercado Pago. A interface estava pronta para enviar todos os dados necessários (arquivo, metadados e token do cartão) para o backend orquestrar o fluxo completo de registro e pagamento.

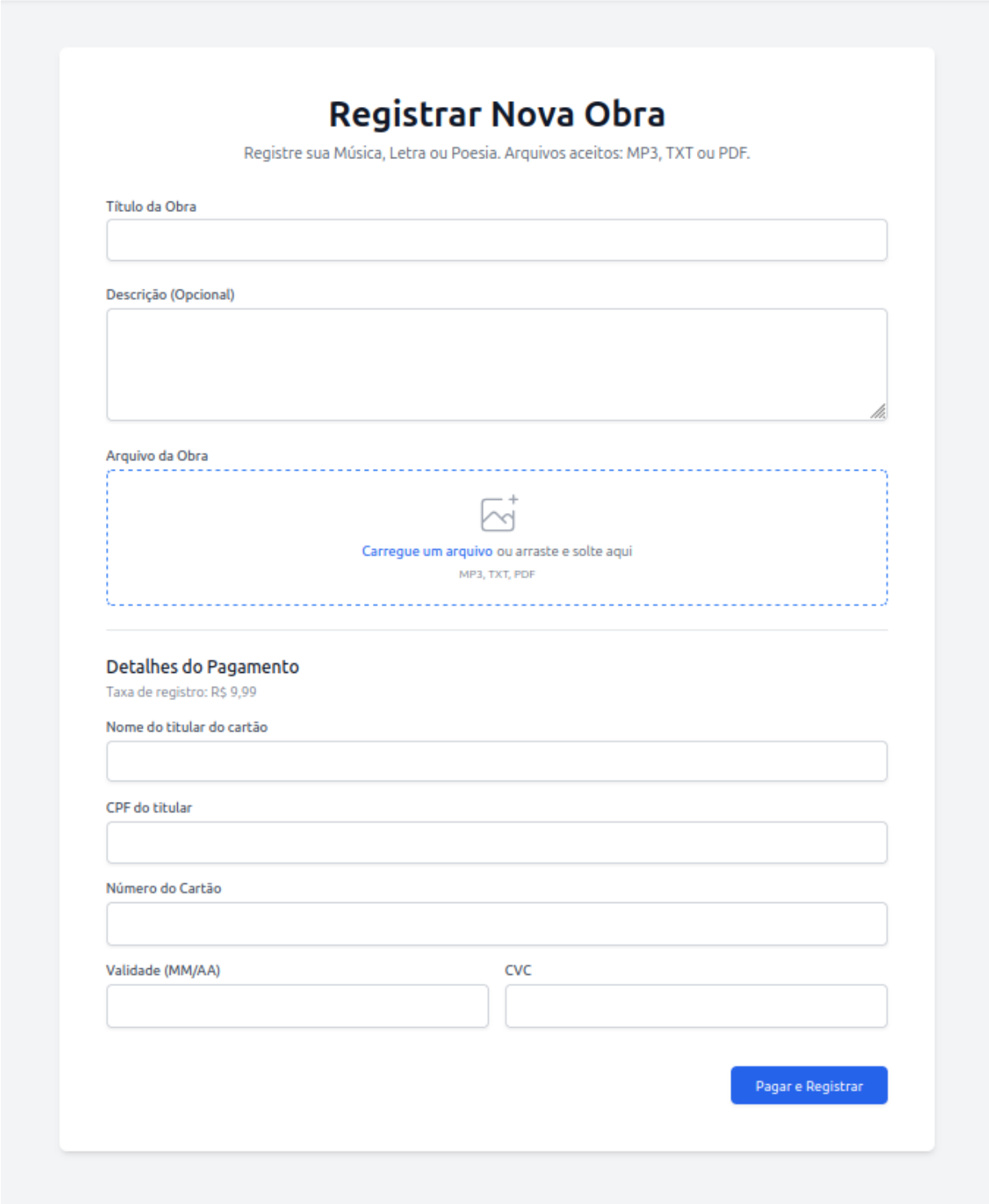

Figura 18 – Interface de registro de obra, incluindo campos de pagamento

7.6 Sprint 5: Integração de Armazenamento Externo (HU03, HU05)

Com a interface do frontend pronta para capturar os dados da obra, o objetivo desta sprint foi implementar a lógica de backend para o armazenamento seguro e a geração das provas criptográficas do arquivo. O foco foi integrar a aplicação com os serviços externos es-

senciais: **Cloudinary** para armazenamento intermediário e **Pinata** para o registro permanente no **IPFS**.

Esta sprint abordou diretamente duas HUs cruciais, embora com uma adaptação estratégica em relação ao plano original para aumentar a robustez do sistema.

- **HU05 (Armazenamento no IPFS):** A história original propunha que o usuário pudesse *escolher* se armazenaria ou não o arquivo no IPFS. Durante o desenvolvimento, foi tomada a decisão arquitetônica de **sempre utilizar o IPFS** para o registro final, a fim de garantir a máxima integridade e descentralização da prova de autoria. Para otimizar a experiência do usuário e facilitar o fluxo de pagamento, foi implementado um **modelo de armazenamento híbrido**: o arquivo é primeiramente enviado ao **Cloudinary** como um buffer temporário e seguro, e somente após a confirmação do pagamento, ele é baixado pelo servidor e enviado ao IPFS via Pinata. Esta abordagem cumpriu o objetivo principal da HU, garantindo uma cópia acessível e descentralizada, ao mesmo tempo que solucionou o desafio de reter o arquivo durante a espera pela confirmação do pagamento. O critério de compactar o arquivo em ZIP foi adiado para futuras iterações, em favor do upload direto do arquivo original para simplificar o MVP.

- **HU03 (Geração de Hash):** Esta HU foi totalmente implementada no backend. A lógica foi adicionada ao `ObrasService` para calcular a "impressão digital"do arquivo. Foi utilizado o módulo `crypto` nativo do Node.js para gerar um hash **SHA-256** a partir do buffer do arquivo. Adicionalmente, a integração com o Pinata também atende a esta HU, pois o **Content Identifier (CID)** retornado pelo IPFS é, por si só, um hash criptográfico único que representa o conteúdo do arquivo.

A estratégia foi criar serviços dedicados e reutilizáveis no backend NestJS para cada integração externa, seguindo o princípio da separação de responsabilidades. Foi criado um `CloudinaryModule` para encapsular toda a lógica de comunicação com o Cloudinary. A lógica de upload para o Pinata foi implementada dentro do `ObrasService`, utilizando a biblioteca `axios` para fazer requisições HTTP do tipo `multipart/form-data`.

A segurança foi mantida como prioridade. Todas as chaves de API e segredos dos serviços externos foram armazenados no arquivo `.env` do backend e acessados através do `ConfigService` do NestJS, evitando a exposição de credenciais no código-fonte.

As tecnologias e ferramentas utilizadas nesta sprint estão listadas a seguir:

- **Backend:** NestJS.
- **Armazenamento Intermediário:** Cloudinary (via SDK `cloudinary`).
- **Armazenamento Descentralizado:** IPFS (via API do Pinata).
- **Comunicação HTTP:** `axios` e `form-data` para o upload para o Pinata.

- **Criptografia:** Módulo `crypto` do Node.js para a geração do hash SHA-256.

As principais atividades realizadas durante esta sprint são apresentadas a seguir:

1. **Criação de Contas Externas:** Foram criadas as contas de desenvolvedor nos planos gratuitos dos serviços **Cloudinary** e **Pinata**.
2. **Configuração de Credenciais:** As chaves de API e os segredos de ambos os serviços foram obtidos e adicionados como variáveis de ambiente ao arquivo `backend/.env`.
3. **Implementação do Módulo Cloudinary:** Foi desenvolvido um módulo completo (`CloudinaryModule`, `CloudinaryProvider`, `CloudinaryService`) para abstrair a lógica de upload. O serviço expõe um método `uploadArquivo` que recebe um arquivo e o envia para a nuvem da Cloudinary.
4. **Implementação da Integração com Pinata:** No `ObrasService`, foi criado o método privado `uploadParaPinata`. Este método monta uma requisição `multipart/form-data` contendo o arquivo e envia para a API do Pinata, utilizando o token JWT de autenticação no cabeçalho.
5. **Implementação da Geração de Hash:** A lógica para calcular o hash SHA-256 do arquivo foi adicionada ao fluxo de registro, utilizando o módulo `crypto`.

Ao término da Sprint 5, o backend do WriteSafe estava equipado com as capacidades fundamentais para o seu fluxo de negócio. A aplicação agora podia receber um arquivo, armazená-lo de forma segura em dois sistemas de nuvem distintos (um centralizado e um descentralizado) e gerar as provas criptográficas necessárias. A base técnica estava pronta para a próxima etapa crucial: a integração com a blockchain Polygon.

7.7 Sprint 6: Configuração e Deploy do Smart Contract (HU04)

O objetivo desta sprint foi construir e publicar o “cartório digital” do WriteSafe, a peça central da prova de autoria na blockchain. O foco foi mergulhar no ecossistema de desenvolvimento da **Polygon**, configurar o ambiente, escrever o Smart Contract e realizar o seu *deploy* (publicação) em uma rede de testes pública.

Esta sprint foi dedicada a construir a infraestrutura necessária para atender à **HU04 (Registro na Blockchain)**. Ao final desta fase, todos os pré-requisitos do lado da blockchain estavam completos:

- **Registro na rede Polygon:** O contrato foi publicado com sucesso na rede de testes **Polygon Amoy**, estando pronto para receber registros.

- **Carteira da plataforma:** Foi criada uma carteira institucional para o servidor, eliminando a necessidade de os criadores interagirem diretamente com a blockchain, conforme especificado na análise.

A efetiva chamada deste contrato a partir do backend, que finaliza a HU04, foi planejada para a sprint seguinte, que integraria todos os serviços.

A estratégia foi criar um subprojeto dedicado à blockchain dentro da pasta `backend`, para isolar as dependências e configurações. A ferramenta escolhida para gerenciar o ciclo de vida do Smart Contract foi o **Hardhat**, um ambiente de desenvolvimento padrão da indústria, conhecido por sua flexibilidade e integração com o ecossistema JavaScript/TypeScript.

Um desafio significativo foi a incompatibilidade entre a versão mais recente do Hardhat e a versão do Node.js utilizada pelo NestJS. A solução foi criar um arquivo `package.json` separado dentro da pasta `backend/blockchain`, permitindo a instalação de uma versão compatível do Hardhat (v2) sem afetar o projeto principal do backend.

Para a conexão com a Polygon, foi utilizado o serviço **Alchemy** como provedor de nó (RPC), garantindo uma conexão estável com a rede de testes Amoy.

As tecnologias e ferramentas utilizadas nesta sprint estão listadas a seguir:

- **Linguagem do Contrato:** Solidity.
- **Ambiente de Desenvolvimento Blockchain:** Hardhat.
- **Blockchain:** Polygon (rede de testes Amoy).
- **Provedor de Nó:** Alchemy.
- **Gerenciamento de Carteira:** MetaMask.

As principais atividades realizadas durante esta sprint são apresentadas a seguir:

1. **Criação de Contas Externas:**
    - Foi criada uma conta de desenvolvedor na **Alchemy** para obter a URL do nó RPC da rede Polygon Amoy.
    - Utilizando a extensão de navegador **MetaMask**, foi criada uma nova carteira digital para servir como a carteira institucional do servidor do WriteSafe conforme Figura 19.
2. **Configuração do Ambiente Hardhat:** Foi inicializado um novo projeto Hardhat na pasta `backend/blockchain`, resolvendo os conflitos de versão do Node.js. O arquivo de configuração `hardhat.config.ts` foi modificado para se conectar à rede Amoy, utilizando as credenciais da Alchemy e a chave privada da carteira do servidor, ambas armazenadas de forma segura no arquivo `.env`.

3. **Escrita do Smart Contract:** Foi desenvolvido o contrato `WriteSafeRegistry.sol`. O contrato foi projetado com uma estrutura de dados (`struct`) para armazenar o autor, o *timestamp* e o hash do IPFS, e uma função pública `registrarObra` para criar novos registros.

4. **Obtenção de Fundos de Teste:** Um desafio notável foi obter MATIC (a moeda da Polygon) de teste para a carteira do servidor, devido às novas medidas anti-spam dos *faucets* (torneiras) públicos. Após diversas tentativas em diferentes plataformas, os fundos foram obtidos com sucesso através da plataforma oficial da moeda, viabilizando o pagamento das taxas de transação (*gas*) para o deploy.

5. **Compilação e Deploy:** O Smart Contract foi compilado com sucesso usando o comando `npx hardhat compile`. Em seguida, utilizando o sistema de deploy **Hardhat Ignition**, o contrato foi publicado na rede de testes Polygon Amoy. O endereço do contrato publicado foi então salvo como uma variável de ambiente (`WRITESAFE_CONTRACT_ADDRESS`) para uso futuro pelo backend.

Ao término da Sprint 6, o “cartório digital” do WriteSafe estava oficialmente no ar na blockchain. O projeto agora possuía um Smart Contract funcional e publicado, um endereço de contrato definido, e todas as credenciais necessárias para que o backend pudesse interagir com ele. Esta sprint superou a principal barreira técnica do projeto, deixando tudo preparado para a orquestração final do fluxo de registro na próxima fase.

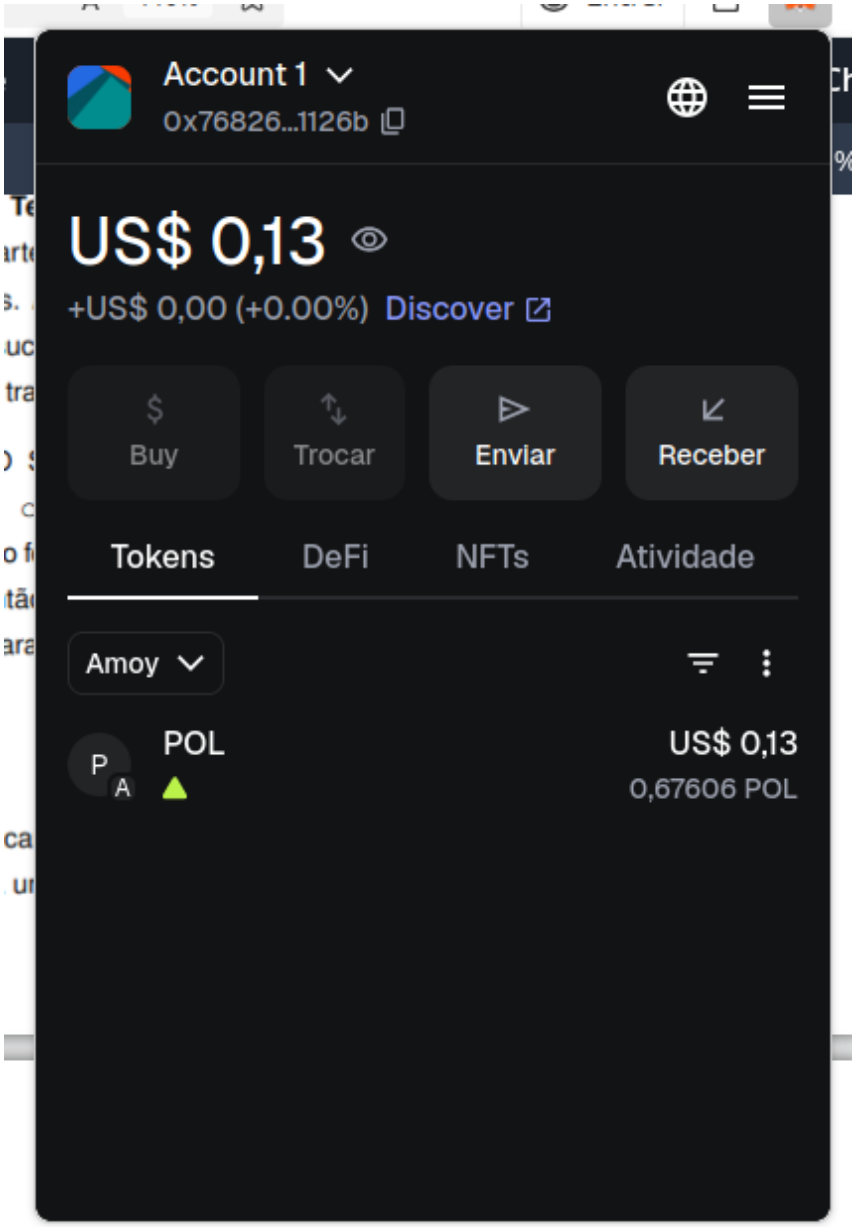

Figura 19 – Carteira da plataforma WriteSafe na MetaMask

7.8 Sprint 7: Orquestração do Fluxo de Pagamento e Registro (HU02 a HU07)

O objetivo desta sprint foi orquestrar o fluxo de ponta a ponta para o registro de uma obra, integrando a interface do usuário com os serviços de armazenamento, a blockchain e um sistema de pagamento. A meta era transformar a ação de "Registrar Obra"em uma transação completa, segura e automatizada, resultando na geração de todas as provas digitais para o criador.

Esta sprint foi a mais abrangente, pois finalizou e integrou a implementação de múltiplas HUs, representando o clímax do desenvolvimento do MVP:

- **HU02 (Envio de Obra):** Foi finalizada com a adição da etapa de pagamento, que agora precede o registro definitivo.
- **HU03 (Geração de Hash), HU04 (Registro na Blockchain) e HU05 (Armazenamento no IPFS):** A execução dessas três HUs foi automatizada e condicionada ao sucesso de um pagamento, sendo orquestrada pelo backend. O sucesso do upload para o IPFS pôde ser verificado diretamente no painel do Pinata, como mostra a Figura 20.
- **HU06 (Certificado Digital):** A lógica para a geração do certificado em PDF foi implementada no backend. A funcionalidade foi integrada à página "Minhas Obras", permitindo que o criador baixe o documento.
- **HU07 (Validação Pública):** A entrega desta HU foi consolidada ao garantir que os hashes de transação, como o exemplificado na Figura 21, fossem salvos corretamente no banco de dados e exibidos como links funcionais.

Além disso, foi implementada uma funcionalidade de negócio não prevista inicialmente nas HUs: a integração com um gateway de pagamento real (**Mercado Pago**) para simular a monetização do serviço.

A estratégia inicial de simular um pagamento com PIX foi abandonada devido a limitações no ambiente de testes do gateway. A abordagem foi pivotada para a implementação de um fluxo de pagamento com **cartão de crédito de teste**, que se mostrou mais robusto e completo para o MVP.

A arquitetura adotada foi a de um **fluxo assíncrono baseado em webhooks**, considerada a mais segura e escalável. O processo foi dividido em duas fases:

1. **Iniciação (Síncrona):** O frontend envia os dados da obra e do cartão. O backend tokeniza o cartão de forma segura com o SDK do Mercado Pago, envia o arquivo para o Cloudinary como armazenamento temporário, cria uma "pré-obra"no banco com status `AGUARDANDO_PAGAMENTO`, e inicia a cobrança.
2. **Finalização (Assíncrona):** Após o Mercado Pago processar o pagamento, ele envia uma notificação (*webhook*) para um endpoint dedicado no backend. Somente após receber e validar essa notificação, o backend executa o processo final e custoso: baixar o arquivo do Cloudinary, enviá-lo para o IPFS, registrar na Polygon e atualizar o status da obra para `COMPLETO`.

Para viabilizar o teste do webhook em ambiente local, foi utilizada a ferramenta **ngrok**, que criou um túnel seguro entre o servidor do Mercado Pago na internet e a aplicação NestJS rodando na máquina de desenvolvimento. A Figura 22 ilustra o painel do Mercado Pago com o log de entrega bem-sucedida das notificações de teste para o endpoint local.

As tecnologias e ferramentas utilizadas nesta sprint estão listadas a seguir:

- **Gateway de Pagamento:** Mercado Pago (SDK para backend e SDK JS v2 para frontend).
- **Túnel de Desenvolvimento:** ngrok.
- **Geração de PDF:** `pdfkit` e `qrcode` no backend NestJS.
- **Comunicação com Blockchain:** `ethers.js` para interagir com o Smart Contract.

As principais atividades realizadas durante esta sprint são apresentadas a seguir:

1. **Configuração do Gateway de Pagamento:** Foi criada uma conta de desenvolvedor no **Mercado Pago** e configurada uma aplicação para obter as credenciais de teste (*Public Key* e *Access Token*). Foram superados desafios relacionados à validação da conta e à obtenção das chaves corretas para o ambiente de sandbox.
2. **Implementação do Backend de Pagamento:** Foi criado o `PagamentosModule` no NestJS. A lógica para criar pagamentos com cartão de crédito e para processar as notificações de webhook foi implementada no `PagamentosService`.
3. **Refatoração do Fluxo de Registro:** O `ObrasController` e o `ObrasService` foram profundamente refatorados para acomodar o fluxo assíncrono de pagamento. Foi criado o método `iniciarRegistro` (para a fase 1) e o método `executarRegistroBlockchain` (para a fase 2, acionado pelo webhook).
4. **Implementação da Geração de Certificado:** No `ObrasService`, foi implementado o método `gerarCertificadoPDF`, que utiliza a biblioteca `pdfkit` para criar um certificado em PDF contendo a logo, os dados da obra, os hashes de prova e um QR Code que leva à transação no Polygonscan.
5. **Implementação do Frontend de Pagamento:** A página "Registrar Nova Obra"foi atualizada para incluir os campos seguros do cartão de crédito, injetados pelo SDK do Mercado Pago. A lógica do componente foi reescrita para, no `onSubmit`, primeiro tokenizar o cartão e depois enviar o token seguro para o backend.

Ao término da Sprint 7, o fluxo principal e mais complexo do WriteSafe estava completo e funcional. A aplicação agora orquestrava com sucesso a interação entre o usuário, o gateway de pagamento e os múltiplos serviços de backend (Cloudinary, IPFS, Polygon, banco de dados). O sistema era capaz de processar um registro de obra de ponta a ponta de forma segura e assíncrona, culminando na geração de um certificado digital válido.

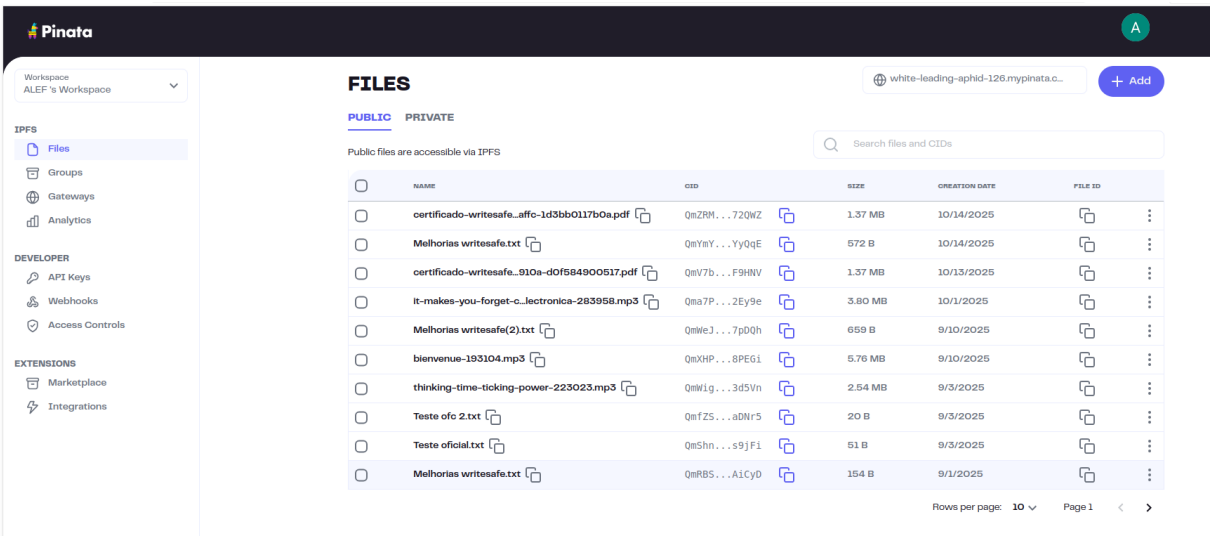

Figura 20 – Interface da plataforma Pinata, a hospedeira dos arquivos IPFS

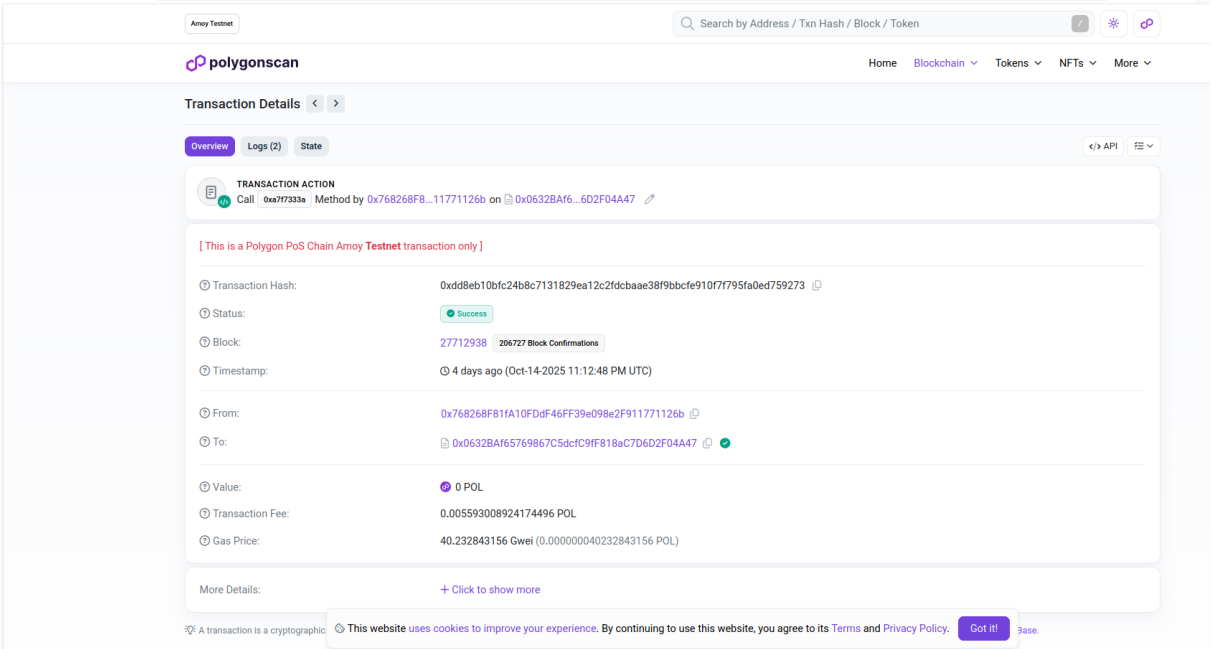

Figura 21 – Exemplo de registro da transação imutável na rede da Polygon, na Polygonscan

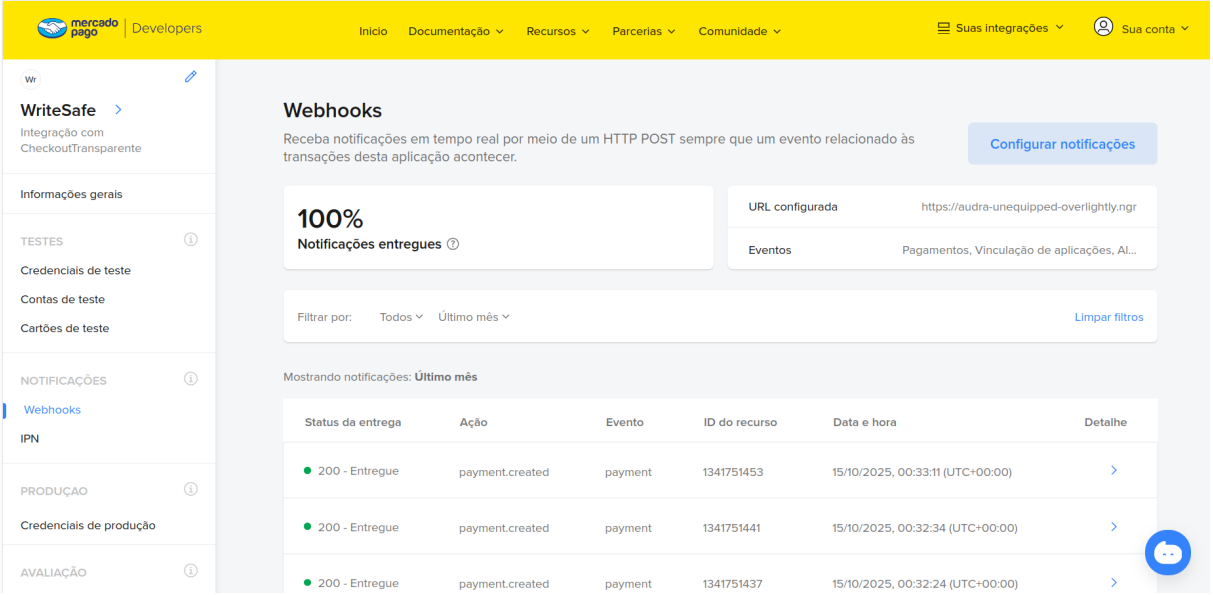

Figura 22 – Configuração e log de Webhooks no Mercado Pago, evidenciando o recebimento bem-sucedido (Status 200) das notificações de teste no backend

7.9 Sprint 8: Dashboards, Finalização do MVP(HU06, HU07, HU10)

O objetivo desta sprint final foi concluir o MVP, entregando as interfaces de visualização de dados tanto para o criador quanto para o administrador.

Esta sprint foi focada em entregar as interfaces que permitem aos usuários visualizar os resultados do fluxo de registro, atendendo a HUs de diferentes perfis:

- **HU06 (Certificado Digital) e HU07 (Validação Pública):** Estas HUs foram finalizadas com a criação da página "Minhas Obras". A interface implementada exibe uma lista de todos os registros do usuário, fornecendo links diretos para as provas no IPFS e no Polygonscan, além de um botão funcional para baixar o certificado em PDF gerado pelo backend.

- **HU10 (Painel Administrativo):** Esta HU foi completamente implementada. Foi desenvolvida uma área administrativa segura, acessível apenas a usuários com o perfil "Admin". O painel exibe estatísticas vitais da plataforma, como o número total de usuários e de obras registradas, consultando um endpoint de backend dedicado e protegido.

- **HU08 e HU09 (Validador Público):** As funcionalidades para o perfil de Validador Público foram conscientemente deixadas fora do escopo do MVP e planejadas para iterações futuras, a fim de priorizar o fluxo principal do criador e a gestão pelo administrador.

A estratégia de desenvolvimento seguiu o padrão estabelecido nas sprints anteriores. Para o **frontend**, o foco foi criar componentes data-driven, que buscam informações da API e as exibem de forma clara. Foi implementado um novo guarda de rota, o `adminGuard`, para

proteger a rota do painel administrativo com base na permissão (*role*) contida no Token JWT do usuário.

No **backend**, a estratégia foi criar endpoints seguros e eficientes para fornecer os dados agregados. O `AuthService` foi modificado para incluir a *role* do usuário no *payload* do JWT no momento do login. Foi criado um novo módulo, o `AdminModule`, com rotas protegidas por um `RolesGuard`, garantindo que apenas administradores pudessem acessar as estatísticas gerais da plataforma.

As tecnologias e ferramentas utilizadas nesta sprint estão listadas a seguir:

- **Frontend:** Angular.
- **Backend:** NestJS, Prisma.
- **Containerização:** Docker.

As principais atividades realizadas durante esta sprint são apresentadas a seguir:

1. **Desenvolvimento da Página "Minhas Obras":** Foi criado o endpoint `GET /obras` no backend para buscar as obras do usuário autenticado. No frontend, a página foi implementada para consumir esses dados e exibi-los em uma tabela, incluindo os links para as provas e o botão para baixar o certificado, conforme Figura 24.
2. **Desenvolvimento do Dashboard do Usuário:** A página do dashboard foi enriquecida para buscar e exibir dados reais. Foi criado um endpoint `GET /obras/stats` no backend para fornecer estatísticas específicas do usuário, que foram exibidas em cards na interface.
3. **Implementação do Painel Administrativo:** O sistema de permissões foi finalizado conforme Figura 23. No backend, o JWT passou a incluir a *role* do usuário, e foi criado o `AdminModule` com o endpoint `GET /admin/stats`, protegido pelo `RolesGuard`. No frontend, foi criada a página `/admin/dashboard`, protegida pelo `adminGuard`, que busca e exibe as estatísticas gerais da plataforma.

Ao término da Sprint 8, o MVP do WriteSafe estava funcionalmente completo. A aplicação não só executava o fluxo de registro de ponta a ponta, mas também fornecia interfaces para que os usuários e administradores pudessem visualizar e gerenciar os dados.

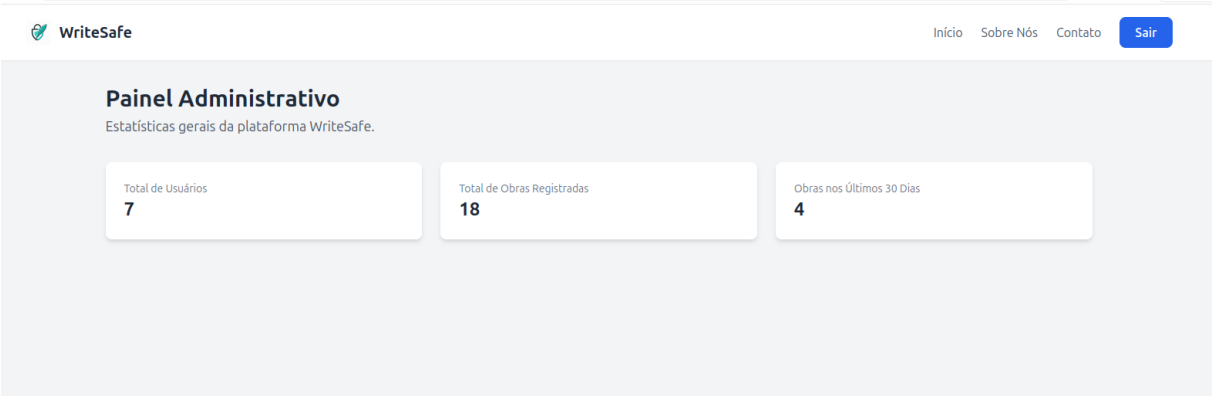

Figura 23 – Interface administrativa do WriteSafe

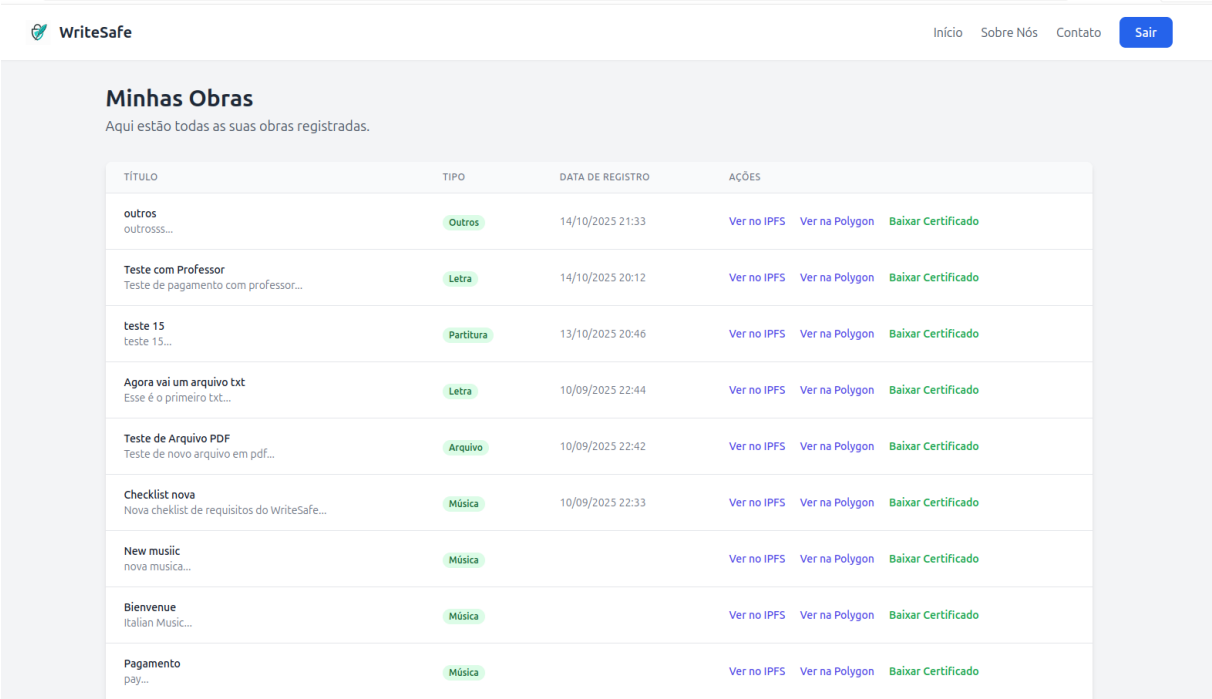

Figura 24 – Listagem de obras registradas no WriteSafe

8 CONCLUSÃO

Este trabalho concluiu o ciclo de desenvolvimento do WriteSafe, resultando na implementação de uma aplicação web funcional para a prova digital de autoria de músicas, letras e poesias. O sistema utiliza a blockchain Polygon para o registro de anterioridade, o armazenamento descentralizado por meio do IPFS (via Pinata) e o Cloudinary como serviço intermediário de hospedagem, validando a arquitetura proposta para gerar uma evidência de autoria imutável, datada e publicamente verificável.

O fluxo de registro desenvolvido abstrai a complexidade da Web3 para o usuário final. Foi criado um processo que se inicia com o envio da obra e a simulação de pagamento com cartão de crédito, utilizando o ambiente de testes do Mercado Pago. A confirmação do pagamento, gerenciada por webhooks, aciona um processo assíncrono no backend que orquestra o envio do arquivo ao IPFS, o registro de seu Content Identifier (CID) em um Smart Contract na rede Polygon Amoy e a persistência dos dados no banco da aplicação. Como resultado, o sistema gera e disponibiliza ao criador um certificado digital em PDF, contendo os metadados da obra, os hashes de comprovação e um QR Code que direciona à transação pública no Polygonscan.

Com essas entregas, o desenvolvimento alcançou os objetivos definidos para o Mínimo Produto Viável (MVP), abrangendo as funcionalidades essenciais destinadas aos perfis de Criador e Administrador. O resultado é uma plataforma funcional que não apenas cumpre os requisitos de registro autoral, mas também oferece interfaces intuitivas de gerenciamento — como o painel “Minhas Obras” e o painel Administrativo para acompanhamento de estatísticas. Mais do que uma solução técnica, o WriteSafe representa uma contribuição prática à valorização da autoria, ao oferecer uma alternativa acessível, transparente e tecnologicamente avançada para a proteção de direitos no contexto digital.

A base técnica consolidada neste trabalho possibilita a expansão natural da plataforma em futuras versões, incluindo a implementação das funcionalidades do perfil de Validador Público, a criação de um marketplace para licenciamento de obras e a integração com novos métodos de pagamento. Essas perspectivas reforçam a viabilidade de modernizar os mecanismos de registro autoral, demonstrando que tecnologias descentralizadas podem ser aplicadas de forma efetiva, segura e socialmente relevante à proteção da propriedade intelectual.

Os trabalhos futuros previstos, com suas respectivas funcionalidades e aprimoramentos técnicos, encontram-se detalhados no Apêndice A, que apresenta o planejamento de evolução da plataforma para os três perfis de usuário e consolida as próximas etapas de desenvolvimento do WriteSafe.

REFERÊNCIAS

AMBLER, S. W. **Object-Relational Mapping**. 2003. Available online: http://www.agiledata.org/essays/mappingObjects.html.

ANDERSON, D. J. **Kanban: Successful Evolutionary Change for Your Technology Business**. 2010. Available online: https://www.amazon.com/Kanban-Successful-Evolutionary-Technology-Business/dp/0984521402.

Authora Digital. **Authora Digital**. 2024. Acesso em: 5 abr. 2025. Disponível em: https://www.authoradigital.com.br/.

BENET, J. Ipfs - content addressed, versioned, p2p file system. *In*: **Proceedings of the 2014 Conference on Innovations in Distributed Systems**. [*S.l.*: *s.n.*], 2014. Available online: https://www.researchgate.net/publication/263930348_IPFS_-_Content_Addressed_Versioned_P2P_File_System.

BRASIL. **Medida Provisória nº 2.200-2, de 24 de agosto de 2001**. 2001. Disponível em: http://www.planalto.gov.br/ccivil_03/mpv/2200-2.htm.

BUTERIN, V. **Ethereum White Paper: A Next Generation Smart Contract and Decentralized Application Platform**. 2014. Available online: https://ethereum.org/en/whitepaper/.

CAO, Y. **China's Supreme Court Rules that Blockchain Evidence is Legally Binding**. 2018. Disponível em: https://www.coindesk.com/policy/2018/09/07/chinas-supreme-court-rules-that-blockchain-evidence-is-legally-binding/.

CHACON, S.; STRAUB, B. **Pro Git**. 2014. Available online: https://git-scm.com/book/en/v2.

CLOUDINARY. **Cloudinary: Media Management and Delivery Platform**. 2023. Acesso em: 17 jun. 2025. Disponível em: https://cloudinary.com.

CORPORATION, O. **MySQL Documentation**. 2023. Available online: https://dev.mysql.com/doc/.

DOCKER, I. **Docker Overview**. 2023. Disponível em: https://docs.docker.com/get-started/overview/.

DSDM Consortium. **The DSDM Agile Project Framework Handbook**. Ashford, England: DSDM Consortium, 2014. ISBN 978-0-9548869-7-5.

Fundação Biblioteca Nacional. **Tabela de Preços de Serviços – EDA (Escritório de Direitos Autorais)**. 2025. https://www.gov.br/bn/pt-br/atuacao/direitos-autorais-1/tabela-de-precos. Valor da taxa de R$ 40,00 para registro por pessoa física. Disponível em: https://www.gov.br/bn/pt-br/atuacao/direitos-autorais-1/tabela-de-precos.

FYI. **Webhooks.fyi**. 2023. Disponível em: https://webhooks.fyi/.

GITHUB. **GitHub Documentation**. 2023. Available online: https://docs.github.com/.

GITHUB. **GitHub Projects Documentation**. 2023. Available online: https://docs.github.com/en/issues/planning-and-tracking-with-projects/learning-about-projects.

GOOGLE. **Angular Documentation**. 2023. Available online: https://angular.io/docs.

Governo Federal. **Registro de Obras Intelectuais na Biblioteca Nacional**. 2023. Disponível em: https://www.gov.br/bn/pt-br/atuacao/direitos-autorais-1.

InspireIP. **InspireIP - About Us**. 2024. Acesso em: 5 abr. 2025. Disponível em: https://inspireip.io/about-us.

JUNIOR, A. M. de O.; NEJAIM, V. M.; NEJAIM, A. C. B. L. Desafios regulatórios no ecossistema brasileiro de streaming de música: uma análise das lacunas nas leis que tratam dos direitos autorais. **Revista De Gestão E Secretariado**, v. 15, n. 1, p. 1365–1384, 2024.

JusBrasil. **Como registrar músicas na Biblioteca Nacional e garantir seus direitos autorais**. 2023. Acessado em: 4 abr. 2025.

LEGISLATURE, A. S. **HB 2417 – Electronic Transactions; Blockchain Technology**. 2018. https://www.azleg.gov/legtext/53leg/2r/bills/hb2417p.pdf.

LEI nº 9.610, de 19 de fevereiro de 1998. 1998. https://www.planalto.gov.br/ccivil_03/leis/l9610.htm. Acessado em abr. 2025.

MICROSOFT. **TypeScript Documentation**. 2023. Available online: https://www.typescriptlang.org/docs/.

MUSIC, U. **A blockchain platform for music rights and royalty distribution**. 2019. https://www.decential.io/articles/deeper-and-harder-to-show-off-the-ujo-experiment-and-on-chain-musics-identity-crisis. Plataforma descontinuada. Acesso em: 23 abr. 2025. Disponível em: https://www.decential.io/articles/deeper-and-harder-to-show-off-the-ujo-experiment-and-on-chain-musics-identity-crisis.

MYWRITES. **Registro de Composições Musicais com Blockchain**. 2025. https://mywrites.co. Acesso em: 23 abr. 2025. Disponível em: https://mywrites.co.

MYśLIWIEC, K. **Nest.js Documentation**. 2023. Available online: https://docs.nestjs.com/.

NACIONAL, F. B. **Escritório de Direitos Autorais (EDA)**. 2025. Available online: http://www.bn.gov.br/servicos/direitos-autorais.

NAKAMOTO, S. Bitcoin: A peer-to-peer electronic cash system. **Decentralized Network Whitepaper**, 2008. Available online: https://bitcoin.org/bitcoin.pdf.

NVIE. **GitFlow Workflow**. 2010. Available online: https://nvie.com/posts/a-successful-git-branching-model/.

PAGO, M. **Mercado Pago Developers**. 2023. Disponível em: https://www.mercadopago.com.br/developers/pt/docs.

PATENTES, V. M. e. **Registro de anterioridade via Blockchain**. 2023. Acesso em: 23 abr. 2025. Disponível em: https://www.vilage.com.br/registro-de-direitos-autorais/registro-de-anterioridade-via-blockchain/.

SCHWABER, K.; SUTHERLAND, J. **Scrum Guide**. 2020. Available online: https://scrumguides.org/.

TEAM, P. **Prisma Documentation**. 2023. Available online: https://www.prisma.io/docs.

TEAM, T. C. **Tailwind Documentation**. 2023. Available online: https://tailwindcss.com/docs.

THEMESBERG. **Flowbite Documentation**. 2023. Available online: https://flowbite.com/docs/.

TILKOV, S.; VINOSKI, S. **Node.js Overview**. 2010. Available online: https://www.infoq.com/articles/nodejs/.

UNION, E. **Regulation (EU) No 910/2014 – eIDAS Regulation**. 2014. https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:32014R0910.

VERMONT, S. of. **Bill H.868 – Blockchain Data Admissibility**. 2017. https://legislature.vermont.gov/Documents/2018/Docs/BILLS/H-0868/H-0868%20As%20Passed%20by%20Both%20House%20and%20Senate.pdf.

World Intellectual Property Organization. **Berne Convention for the Protection of Literary and Artistic Works**. Geneva: World Intellectual Property Organization (WIPO), 1979. Available online: https://www.wipo.int/treaties/en/ip/berne/.

APÊNDICE A – Funcionalidades Futuras

Este apêndice apresenta as funcionalidades complementares que foram conscientemente despriorizadas do Produto Mínimo Viável (MVP), mas que representam a evolução natural da plataforma **WriteSafe**. As ideias aqui listadas são classificadas como *Should Have* (Deveria Ter) e *Could Have* (Poderia Ter), segundo o modelo de priorização MoSCoW, e servem como um roteiro para iterações futuras.

A.1 Evoluções no Fluxo de Registro e Pagamento

Should Have

- **Precificação Dinâmica de Registro:** Implementar uma API de cotação (ex: CoinGecko) para calcular o custo do registro em tempo real. A lógica de precificação consideraria o custo variável do *gas* da Polygon e o custo de armazenamento do Pinata (baseado no tamanho do arquivo), adicionando a margem de lucro da plataforma.
- **Fluxo de Pagamento com PIX Real:** Substituir a simulação de pagamento com cartão de crédito por uma integração de produção com o Mercado Pago, focada em PIX, que é o método de pagamento preferencial no Brasil.
- **Suporte a Múltiplos Autores:** Refatorar a lógica de registro para permitir que uma obra seja associada a mais de um usuário (coautoria), dividindo a titularidade no registro.
- **Upload de Arquivos `.zip`:** Implementar a funcionalidade (originalmente prevista na HU05) de permitir o upload de um arquivo `.zip` contendo múltiplos arquivos (ex: letra.pdf, musica.mp3, capa.jpg), tratando este pacote como a "obra"principal a ser registrada.

Could Have

- **IPFS Opcional:** Reavaliar a decisão de sempre usar o IPFS (tomada no MVP) e devolver ao usuário a opção de escolher um registro mais barato, que armazene apenas o hash SHA-256 na blockchain, sem o armazenamento descentralizado do arquivo.
- **Assinatura ICP-Brasil:** Integrar um serviço de assinatura digital para assinar o certificado em PDF com um padrão oficial ICP-Brasil, aumentando sua validade jurídica formal.

A.2 Funcionalidades para o Criador

Should Have

- **Página de Detalhes da Obra:** Criar uma página individual para cada obra registrada, exibindo todas as informações, uma prévia do arquivo (via Cloudinary) e um histórico de eventos.
- **Gerenciamento de Perfil:** Implementar uma página onde o criador possa alterar seu nome e redefinir sua senha (fluxo de "esqueci minha senha").
- **Notificações por E-mail:** Enviar e-mails transacionais automáticos para o usuário após o cadastro (confirmação de conta) e após a finalização bem-sucedida de um registro.

Could Have

- **Edição de Metadados:** Permitir que o usuário edite metadados secundários de uma obra (como descrição ou palavras-chave) após o registro, sem afetar as provas imutáveis (hashes).
- **Marketplace de Licenciamento:** Uma grande evolução que permitiria aos criadores licenciar suas obras registradas para outros usuários diretamente pela plataforma.

A.3 Funcionalidades para o Validador Público

Should Have

- **Implementação da HU08 (Consulta por Validador):** Criar uma página de consulta pública, acessível a qualquer visitante, onde seja possível inserir um hash (SHA-256 ou CID) e verificar se ele existe nos registros do WriteSafe.
- **Implementação da HU09 (Visualização por Validador):** Criar a interface que exibe os dados públicos de uma obra validada (Autor, Título, Data, links das provas) para confirmar a autenticidade.

Could Have

- **API Pública de Validação:** Expor um endpoint de API público para que sistemas externos possam consultar e validar registros do WriteSafe automaticamente.

A.4 Funcionalidades para o Administrador

Should Have

- **Gestão de Usuários e Obras:** Expandir o painel administrativo para permitir ao admin visualizar, editar ou remover usuários e registros de obras (em casos de disputas ou violações).
- **Painel para Resolução de Disputas:** Criar uma interface dedicada para gerenciar disputas de autoria.

Could Have

- **Geração de Relatórios:** Adicionar funcionalidades para o admin gerar relatórios periódicos de atividade da plataforma (financeiro, registros, etc.).
- **Gestão de Taxas:** Criar uma interface no painel admin para ajustar dinamicamente o preço fixo ou a margem de lucro dos registros.