MEC-SETEC
INSTITUTO FEDERAL MINAS GERAIS - *Campus* Formiga
Curso de Ciência da Computação

ARQUITETURA DESCENTRALIZADA PARA A DISTRIBUIÇÃO E APROPRIAÇÃO DE JOGOS ELETRÔNICOS

Ruan Rafael Leite Silva

Orientador: Everthon Valadão dos Santos

Formiga - MG

2025

RUAN RAFAEL LEITE SILVA

ARQUITETURA DESCENTRALIZADA PARA A DISTRIBUIÇÃO E APROPRIAÇÃO DE JOGOS ELETRÔNICOS

Monografia do trabalho de conclusão de curso apresentado ao Instituto Federal Minas Gerais - Campus Formiga, como requisito parcial para a obtenção do título de Bacharel em Ciência da Computação.

Orientador: Everthon Valadão dos Santos

Formiga - MG

2025

Silva, Ruan Rafael Leite
S004.4a Arquitetura descentralizaa para a distribuição e apropriação de jogos eletrônicos. /Ruan Rafael Leite Silva – Formiga : IFMG, 2025.
74 p. :il. color.

Orientador: Prof. Me. Everthon Valadão dos Santos
Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) – Instituto Federal de Educação, Ciência e Tecnologia de Minas Gerais – *Campus* Formiga.

1. Blockchain. 2. IPFS. 3. Marketplace. 4. Jogos digitais. 5. Tokens não Fungíveis. I. Santos, Everthon Valadão dos. II. Título.

CDD 004.4

Ficha catalográfica elaborada pela Bibliotecária Lívia Renata Santos- CRB/6-2561

MINISTÉRIO DA EDUCAÇÃO
SECRETARIA DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE MINAS GERAIS
Campus Formiga
Diretoria de Ensino
Docência Área Acadêmica de Computação
Rua São Luiz Gonzaga, s/n - Bairro São Luiz - CEP 35570-000 - Formiga - MG
- www.ifmg.edu.br

RUAN RAFAEL LEITE SILVA

Arquitetura descentralizada para a distribuição e apropriação de jogos eletrônicos

Trabalho de Conclusão de Curso apresentado ao Instituto Federal de Minas Gerais - Campus Formiga, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

APROVADO em: 10 de dezembro de 2025.

BANCA EXAMINADORA

Prof.º Me. Everthon Valadão dos Santos (orientador, IFMG)

Prof.º Dr. Bruno Ferreira (IFMG)

Prof.º Me. Walace de Almeida Rodrigues (IFMG)

Documento assinado eletronicamente por **Bruno Ferreira**, **Professor**, em 10/12/2025, às 18:15, conforme Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Everthon Valadao dos Santos**, **Professor**, em 10/12/2025, às 18:15, conforme Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Walace de Almeida Rodrigues**, **Professor**, em 10/12/2025, às 18:18, conforme Decreto nº 10.543, de 13 de novembro de 2020.

A autenticidade do documento pode ser conferida no site https://sei.ifmg.edu.br/consultadocs informando o código verificador **2540080** e o código CRC **4CC65925**.

Criado por everthon.valadao, versão 2 por everthon.valadao em 28/11/2025 10:37:08.

À minha família, que sempre acreditou em mim e me deu todo o suporte necessário para chegar até aqui.

Agradecimentos

Gostaria de agradecer ao professor Everthon, por ser meu orientador e um ótimo tutor durante todo o curso. Agradeço também em especial minha família por ter me dado essa oportunidade e terem me apoiado sempre durante a jornada, sem eles nada teria sido possível.

"A menos que você tente fazer algo além do que já domina, você nunca crescerá."

(Ralph W. Emerson)

Resumo

A distribuição digital de jogos eletrônicos é dominada por plataformas centralizadas como Steam e Epic Games, que fornecem aos usuários apenas licenças de uso revogáveis em vez de propriedade real sobre os ativos adquiridos. Esse modelo mantém o controle dos jogos nas mãos dessas plataformas, limitando a autonomia dos usuários e sua capacidade de transferir, comercializar ou utilizar os jogos fora do sistema original. Este trabalho propõe o desenvolvimento de uma arquitetura descentralizada para distribuição e apropriação de jogos eletrônicos, utilizando tecnologias de *Blockchain* e sistemas de arquivos distribuídos. Foi desenvolvido um protótipo funcional (*Minimum Viable Product*) que integra a *Blockchain* Solana para o gerenciamento de licenças através de *Tokens* Não Fungíveis (NFTs), o protocolo IPFS (*InterPlanetary File System*) para distribuição *peer-to-peer* dos arquivos, um servidor indexador centralizado opcional e configurável, e uma aplicação *desktop* desenvolvida em Tauri que unifica autenticação via carteira digital, navegação no *marketplace*, biblioteca de jogos e execução. Os testes realizados validaram a viabilidade técnica da solução, demonstrando a resiliência do sistema e propriedade real dos ativos pelos usuários. O trabalho valida a hipótese central de que é tecnicamente viável construir um marketplace de jogos que transfira o controle dos ativos digitais das plataformas intermediárias para os usuários finais. O sistema desenvolvido demonstra que a apropriação real de licença de jogo pode ser alcançada através da descentralização, sem a necessidade de plataformas intermediárias centralizadas. O sistema implementado engloba o ciclo completo de interação: publicação de jogos por desenvolvedores, autenticação descentralizada de usuários, aquisição de licenças como NFTs, download distribuído via IPFS e execução através da aplicação desktop *launcher*.

Palavras-chave: Blockchain, IPFS, Marketplace, Jogos Digitais, Tokens Não Fungíveis.

Abstract

The digital distribution of video games is dominated by centralized platforms such as Steam and Epic Games, which provide users only with revocable usage licenses rather than actual ownership of the acquired assets. This model keeps control of games in the hands of these platforms, limiting user autonomy and their ability to transfer, trade, or use games outside the original system. This work proposes the development of a decentralized architecture for the distribution and ownership of video games, using Blockchain technology and distributed file systems. A functional prototype (Minimum Viable Product) was developed, integrating the Solana Blockchain for license management through Non-Fungible Tokens (NFTs), the IPFS (InterPlanetary File System) protocol for peer-to-peer file distribution, an optional and configurable centralized indexing server, and a desktop application developed in Tauri that unifies wallet-based authentication, marketplace browsing, game library, and execution. The conducted tests validated the technical feasibility of the solution, demonstrating the system's resilience and actual ownership of assets by users. This work validates the central hypothesis that it is technically feasible to build a game marketplace that transfers control of digital assets from intermediary platforms to end users. The developed system demonstrates that true game license ownership can be achieved through decentralization, without the need for centralized intermediary platforms. The implemented system encompasses the complete interaction cycle: game publishing by developers, decentralized user authentication, license acquisition as NFTs, distributed download via IPFS, and execution through the launcher desktop application.

Keywords: Blockchain, IPFS, Marketplace, Digital Games, Non-Fungible Tokens.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

API	*Application Programming Interface* (Interface de Programação de Aplicações)
AWS	*Amazon Web Services*
CDN	*Content Delivery Network* (Rede de Entrega de Conteúdo)
CID	*Content Identifier* (Identificador de Conteúdo)
CLI	*Command Line Interface* (Interface de Linha de Comando)
CORS	*Cross-Origin Resource Sharing* (Compartilhamento de Recursos de Origem Cruzada)
CSS	*Cascading Style Sheets* (Folhas de Estilo em Cascata)
DHT	*Distributed Hash Table* (Tabela Hash Distribuída)
DLT	*Distributed Ledger Technology* (Tecnologia de Registro Distribuído)
DRM	*Digital Rights Management* (Gerenciamento de Direitos Digitais)
EC2	*Elastic Compute Cloud*
FUSE	*Filesystem in Userspace* (Sistema de Arquivos no Espaço do Usuário)
HTML	*HyperText Markup Language* (Linguagem de Marcação de Hipertexto)
HTTP	*HyperText Transfer Protocol* (Protocolo de Transferência de Hipertexto)
IPFS	*InterPlanetary File System* (Sistema de Arquivos Interplanetário)
IPNS	*InterPlanetary Name System* (Sistema de Nomes Interplanetário)
JSON	*JavaScript Object Notation* (Notação de Objetos JavaScript)
MVP	*Minimum Viable Product* (Produto Mínimo Viável)
NFT	*Non-Fungible Token* (Token Não Fungível)
NPM	*Node Package Manager* (Gerenciador de Pacotes Node)
P2P	*Peer-to-peer* (Ponto a ponto)
PNPM	*Performant NPM*

RPC	*Remote Procedure Call* (Chamada de Procedimento Remoto)
SDK	*Software Development Kit* (Kit de Desenvolvimento de Software)
SOL	Moeda nativa da rede Solana
TTFB	*Time To First Byte* (Tempo até o Primeiro Byte)
URI	*Uniform Resource Identifier* (Identificador Uniforme de Recursos)
URL	*Uniform Resource Locator* (Localizador Uniforme de Recursos)
XML	*eXtensible Markup Language* (Linguagem de Marcação Extensível)

Sumário

1 Introdução

A tecnologia Blockchain tem recebido muita atenção nos últimos anos e vem apresentando um grande crescimento, sendo capaz de transformar diversos setores e indústrias (ALT, 2020). Suas aplicações incluem áreas como *marketplaces* gerais (SUBRAMANIAN, 2017), de dados de saúde (SUBRAMANIAN, 2023), finanças e agricultura (LEDUC; KUBLER; GEORGES, 2021).

Esse aumento de interesse se deve às principais características do modelo descentralizado que a Blockchain oferece, como segurança, confiabilidade e integridade transacional (SUBRAMANIAN, 2017).

Sendo a distribuição digital de jogos realizadas por plataformas centralizadas, os usuários não possuem propriedade sobre os ativos adquiridos. A compra através dessas plataformas não garante o controle total sobre os conteúdos visto que, o controle por parte da plataforma mantêm o usuário dependente de sua existência, regras e permissões. Isso enfraquece a ideia de propriedade digital, já que o comprador não é realmente o dono do jogo.

Diante desse cenário de avanço tecnológico, este trabalho propõe o desenvolvimento de um *marketplace* descentralizado. A arquitetura proposta utilizará uma Blockchain e um servidor centralizado (como *peer* de *bootstrap*) para permitir melhor tempo de resposta, resultando em uma arquitetura híbrida para a distribuição de jogos digitais. A proposta inclui também um *launcher* próprio, responsável por validar a propriedade dos jogos, com base na carteira do usuário, restringindo sua execução apenas ao comprador do ativo.

1.1 Justificativa

A escolha deste projeto está diretamente relacionada à oportunidade identificada de um produto com potencial de mercado e o interesse pessoal do autor, em jogos eletrônicos. O modelo atual de lojas e distribuição digital limita a posse real dos ativos, o que motiva o autor a buscar uma solução que valorize a apropriação do usuário sobre os seus conteúdos comprados.

Além disso, o autor tem interesse científico em Blockchain e já possui alguma experiência na área por meio da atuação profissional com essa tecnologia. Isso favorece a motivação para explorar soluções descentralizadas em um contexto relevante, inovador e em expansão no mercado.

O desenvolvimento do projeto permite aplicar conhecimentos técnicos importantes e relevantes das áreas de sistemas distribuídos e arquitetura de *software*, alinhando o

projeto com os interesses pessoais do autor quanto aos objetivos do curso.

1.2 Objetivos

1.2.1 Objetivo Geral

Desenvolvimento de um sistema com as funcionalidades básicas de um *marketplace* com uma arquitetura híbrida para a distribuição e venda, apropriação e execução de jogos eletrônicos.

A hipótese central deste trabalho consiste na viabilidade técnica de construir um *marketplace* de jogos que transfira o controle dos ativos digitais das plataformas intermediárias para os usuários finais, garantindo apropriação real das licenças através de tecnologias descentralizadas como *Blockchain* e IPFS.

1.2.2 Objetivos Específicos

1. Modelar a representação de um ativo como um *token* não fungível (NFT) e o modelo de posse.
2. Integrar componentes descentralizados (Blockchain, IPFS) e centralizados para equilibrar desempenho e autonomia, compondo uma arquitetura híbrida.
3. Desenvolver o módulo de autenticação via carteira, associando o usuário ao seu ativo.
4. Modelar e desenvolver o fluxo de adição de jogos ao *marketplace*.
5. Desenvolver o módulo de compra e visualização dos jogos adquiridos.
6. Desenvolver o módulo de *download* e propagação dos jogos via IPFS, utilizando o *launcher* como um nó de recepção e propagação dos pacotes.
7. Desenvolver o *launcher* como ponto de entrada da plataforma, integrando as funcionalidades do *marketplace* com uma interface acessível e intuitiva.

2 Fundamentação Teórica

A distribuição de jogos eletrônicos evoluiu significativamente nas últimas décadas, migrando de mídias físicas para digitais, sendo disponibilizadas por plataformas centralizadas como Steam e Epic Games. Essas plataformas oferecem conveniência aos usuários, mas impõem restrições importantes sobre a posse desses ativos, utilizando-se de licenças revogáveis em vez da propriedade real (MAGALHãES; FANTINI, 2022). Tal limitação impacta os usuários negativamente, que, mesmo após a compra, podem ter seu acesso revogado caso a plataforma decida remover ou suspender a distribuição.

Com o avanço das tecnologias descentralizadas e, principalmente, do uso da Blockchain, surgiram novas possibilidades para a gestão e propriedade dos ativos, de forma auditável e imutável (POP et al., 2020). Nesse modelo, os jogos podem ser representados como *tokens* não fungíveis (NFTs), permitindo aos usuários o controle direto sobre seus ativos, inclusive fora da plataforma na qual foram adquiridos (MAGALHãES; FANTINI, 2022). Essa abordagem também viabiliza maior transparência, criação de mercados secundários e interoperabilidade entre sistemas (POP et al., 2020).

Contudo, soluções totalmente descentralizadas ainda enfrentam seus próprios desafios, como desempenho, escalabilidade e experiência do usuário. Por essa razão, modelos híbridos são propostos como alternativas viáveis, conciliando os benefícios da descentralização — como a autonomia do usuário — com a estabilidade, usabilidade e desempenho superior dos sistemas centralizados (POP et al., 2020).

A adoção de uma arquitetura híbrida neste projeto busca equilibrar ambos os mundos. A camada descentralizada será responsável pela autenticação e licenciamento dos ativos digitais. A distribuição será parcialmente descentralizada, sendo propagada pelos nós *peer-to-peer* e por um servidor central, garantindo que ao menos um nó esteja sempre disponível. Já a leitura e indexação das informações serão realizadas de forma centralizada, permitindo uma disponibilização mais rápida e personalizada.

2.1 Arquitetura de sistemas

No contexto deste trabalho, a compreensão sobre os diferentes modelos de arquitetura é essencial para fundamentar as decisões do projeto relacionados a distribuição de jogos digitais e gestão de propriedade decentralizada. A seguir, são apresentados os principais modelos de arquitetura relevantes para o desenvolvimento da solução proposta.

2.1.1 Sistemas centralizados

Sistemas centralizados são aqueles que seguem o modelo cliente-servidor, onde um ou mais clientes podem ser conectados diretamente a um servidor central e, o processamento ocorrerá de forma centralizado nessa entidade central. Nó cliente, por sua vez (usuários), existe uma aplicação responsável apenas por apresentar uma interface gráfica para transformar as ações dos usuários em uma chamada a determinada funcionalidade do servidor central (RUSTAMOV; BEKHZOD, 2024). Os principais benefícios de tal modelo, é o custo benefício para sistemas pequenos, simplicidade de implementação e gerenciamento (não há a necessidade de coordenação entre múltiplos nós) e baixa latência, em troca de um ponto único de falhas (caso o nó central falhe, todo o sistema é interrompido), escalabilidade limitada (conforme a quantidade de usuários crescem, não é possível suprir todos com recursos limitados do servidor central), falta de transparência e segurança (a autoridade central tem controle total sobre os dados) (RUSTAMOV; BEKHZOD, 2024).

2.1.2 Sistemas *Peer to peer*

Os sistemas *Peer-to-peer* (P2P), diferentemente dos sistemas cliente-servidor centralizados, são projetados para distribuir serviços e recursos de forma descentralizada entre os nós de uma rede. Nessa arquitetura, cada nó pode atuar simultaneamente como cliente e servidor, compartilhando capacidades e responsabilidades na rede. O funcionamento do sistema não depende de servidores centralizados, permitindo que a rede continue operacional mesmo com a entrada e saída de nós participantes (COULOURIS et al., 2011).

2.1.3 Sistemas distribuídos

Sistemas distribuídos são caracterizados por aqueles em que seus componentes estão localizados em diversos dispositivos e computadores conectados pela rede, que operam em conjunto para algo útil. O objetivo principal desse tipo de sistema é o compartilhamento de recursos, podendo estes, serem hardwares, outros dispositivos, softwares e dados (COULOURIS et al., 2011).

Segundo Rustamov e Bekhzod (2024), os principais benefícios desse tipo de arquitetura são o equilíbrio de qualidades, oferecendo melhor desempenho, tolerância a falhas, uma melhor escalabilidade e flexibilidade, permitindo diferentes tipos de sistemas em um mesmo ambiente quando comparados aos sistemas centralizados.

2.2 Blockchain

A tecnologia blockchain ou também conhecida como *Distributed Ledger Technology* – DLT é caracterizada por ser um registro de transações compartilhado e distribuído,

não necessitando de uma autoridade central ou de terceiros para o seu funcionamento (MAGALHãES; FANTINI, 2022). Ou seja, cada nó pode se comunicar com outros nós, sem a necessidade de uma autoridade central para ditar como isso ocorre.

Em sua essência, blockchain é um livro-razão (*Ledger*) compartilhado e distribuído que registra todas as transações (SUBRAMANIAN, 2017). É estruturada como uma cadeia formada por blocos ligados. Os blocos, por sua vez, armazenam todas as transações que ocorreram no sistema em um curto período (como, por exemplo, cerca de 12 segundos para a Ethereum). Cada novo bloco inclui também a referência para o bloco anterior ao incluir sua *hash* em seu interior, criando um histórico imutável de toda a atividade da rede (POP et al., 2020).

A blockchain opera em uma rede *peer-to-peer* e a validação da segurança da rede são asseguradas pelos nós (*peers*) e algoritmos de consenso. Quando uma nova transação ou um novo bloco é proposto, os dados são propagados por toda a rede, e cada nó pode verificar e validar os dados. Para evitar problemas de indisponibilidade e segurança como, por exemplo, nós maliciosos ou defeituosos, são necessários os protocolos de consenso para que os nós possam chegar a um acordo válido sobre o estado do sistema (POP et al., 2020).

2.2.1 Carteiras (*wallets*)

A blockchain utiliza criptografia de chave pública para garantir a segurança das transações na rede através de um par de chaves, sendo a chave pública para servir como um endereço de recebimento de criptomoedas ou dados, enquanto a chave privada garante o acesso dos ativos digitais, com ela, é possível verificar a propriedade, autorizar e assinar transações (SUSNJARA; SMALLEY, 2021).

Uma carteira (*wallet*) é um software ou hardware que gerencia e armazena as chaves criptográficas necessárias para realizar operações na blockchain, como enviar, receber e gerenciar ativos digitais, permitindo ao usuário interagir com a rede.

2.2.2 Contas na rede Solana

Segundo Solana Foundation (2025a) todos os dados na rede Solana são armazenados em contas. Todas as contas precisam depositar e manter um saldo mínimo *rent* (aluguel) para que seus dados sejam armazenados na blockchain, a Figura 1 representa a estrutura de armazenamento dos dados de uma conta na rede. O montante necessário a ser mantido é proporcional ao tamanho dos dados da conta, e o cálculo de aluguel pode ser realizado utilizando a CLI da solana. A Tabela 1 mostra a comparação entre diferentes tamanhos de contas armazenadas na rede através de execuções do comando: `solana rent <bytes> url m`.

Tabela 1 – comparação de valores por tamanho armazenado na rede

Armazenamento	**Mínimo em SOL**	**Valor aproximado em Real**
1 byte	0,00089784 SOL	R$ 0,66
1 KB	0,00801792 SOL	R$ 5,87
1 MB	7,29897984 SOL	R$ 5347,29

Fonte: Próprio autor

O endereço de uma conta é único, e a maioria das contas utiliza chaves públicas Ed25519 como seu endereço, comumente representadas por strings base58.

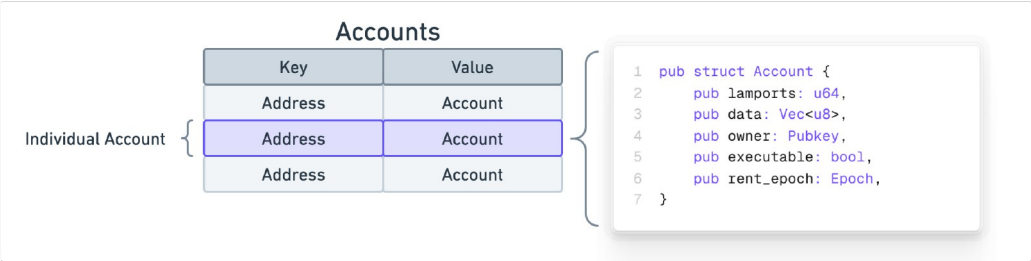

Figura 1 – Estrutura de contas na rede Solana

Fonte: Solana Foundation (2025a)

Existem dois tipos de contas: contas de programas, que armazenam código executável de *smart contracts*, e contas de dados, que não contêm código executável. As contas de dados incluem contas de estado do programa (que registram o estado atual após determinada ação) e contas de carteira (criadas pelo *System Program*) (Solana Foundation, 2025a).

2.2.3 Programas

Contratos inteligentes (*smart contracts*) são programas armazenados e executados na Blockchain que implementam e validam regras de negócio automaticamente. Quando chamados, os contratos executam automaticamente as ações programadas de forma determinística, garantindo transparência e auditabilidade nas transações.

Na Solana os contratos inteligentes são chamados de programas, um programa é uma conta sem estado que contém código executável. Os usuários podem interagir com programas por meio de transações e instruções (Solana Foundation, 2025b).

2.2.3.1 *System program* ou Programa de sistema

O programa de sistema na Solana é o único programa capaz de criar outras contas na rede, transferir SOL (moeda nativa da rede) de contas de sistema (como contas de carteira, por exemplo) para outras contas, reservar espaço para os dados de uma conta e

alterar o proprietário de um programa para outra conta de programa (Solana Foundation, 2025b).

2.2.3.2 *Candy Machine*

A *Candy Machine* é um *smart contract* (programa) desenvolvido pela Metaplex que automatiza a cunhagem e distribuição de NFTs na rede Solana. Diferentemente da cunhagem manual, que exige a assinatura do criador para cada transação, a *Candy Machine* permite que desenvolvedores configurem previamente os metadados de seus NFTs e coleções sem executar a cunhagem imediatamente.

Esse mecanismo resolve um problema de escalabilidade, pois sem ele, cada venda exigiria que o criador executasse manualmente a transação de cunhagem e transferisse o NFT para o comprador. Com a *Candy Machine*, os NFTs são criados sob demanda no momento da compra, delegando os custos de transação aos compradores e permitindo distribuição automatizada sem intervenção do criador.

A *Candy Machine* também oferece guardas (*guards*), que são módulos configuráveis para adicionar regras à cunhagem como, por exemplo, pagamentos obrigatórios carteira (Core Candy Machine, 2025).

2.2.4 Tokens não fungíveis — NFT

Um NFT ou Token não-fungível (*Non-Fungible token*) é um token criptográfico armazenado em uma blockchain que tem como principal objetivo comprovar a propriedade de ativos digitais.

Por ser não fungível significa que não pode ser trocado por algo idêntico, o que o torna adequado para identificar algo ou alguém de maneira única, como licenças no caso do presente projeto. Por serem de origem descentralizada, possuem vários benefícios como verificabilidade (os metadados podem ser verificados publicamente), execução transparente (cunhagem — *minting*, compra, venda e transferências são publicamente acessíveis) (WANG et al., 2021).

2.2.4.1 Coleções (*collection*)

Grupos ou séries de NFTs que compartilhem as mesmas características podem ser chamados de coleções (*NFT Collection*). No caso do presente projeto, cada jogo será representado por uma coleção e cada *token* dessa coleção será uma licença. Dessa forma, é possível que as licenças compartilhem os mesmos metadados dos jogos, como nome, imagem e preço.

2.2.4.2 Metadados

Os metadados de um NFT incluem informações sobre o ativo, como nome, descrição, imagem e atributos personalizados. Embora seja tecnicamente possível armazená-los diretamente na blockchain (*on-chain*), isso implica custos elevados devido ao modelo de *rent* conforme explicado na seção sobre contas na rede solana.

Neste projeto, o IPFS foi utilizado como forma de armazenamento *off-chain* para os metadados, estes incluem nome, descrição, URLs da imagem de capa e do arquivo executável de cada jogo. Uma vez armazenados, apenas o URI apontando para o *hash* IPFS dos metadados é salvo no NFT.

2.3 IPFS - *InterPlanetary Filesystem*

O Sistema de arquivos interplanetário (IPFS) é um protocolo de rede projetado para ser um sistema de arquivos distribuído. Diferentemente da arquitetura tradicional web, que utiliza URLs para a localização dos recursos, o IPFS utiliza endereçamento baseado no próprio conteúdo.

Um nó participante da rede pode armazenar arquivos no seu armazenamento local temporariamente ou por tempo indeterminado através do *pinning*. Além disso, os nós podem comunicar entre si para a transferência e distribuição de dados pela rede.

O armazenamento é realizado por blocos identificados de forma única através do *hash* criptográfico do seu conteúdo (CID). Os blocos podem conter links para outros blocos através de seus CIDs, formando uma estrutura de grafo. Isso garante resistência à adulteração e deduplicação automática (DOAN et al., 2022).

Os arquivos armazenados no IPFS podem ser acessados por meio de uma API por meio de um nó conectado à rede, desde que se conheça o CID do arquivo a ser encontrado. Quando um nó não contém o objeto solicitado, ele é responsável por buscar outros nós que armazenem o conteúdo desejado.

Segundo Benet (2014), para realizar buscas dos arquivos solicitados, o nó IPFS pode consultar uma tabela *hash* distribuída (*Distributed Hash Table - DHT*) a fim de localizar os *peers* ou objetos solicitados. Caso os arquivos sejam iguais ou menores que 1KB, serão armazenados diretamente na DHT, caso contrário, é armazenada apenas a referência para o *peer* que pode fornecer o bloco.

Caso um objeto exista na rede, mas o nó requisitado não o armazena, após encontrar um *peer* na DHT com os arquivos solicitados, o nó baixa e armazena localmente, pelo menos temporariamente. Os nós que desejam garantir que os objetos não sejam excluídos após algum tempo, devem fazer o *pinning* (fixação) do arquivo, assegurando que serão mantidos no armazenamento local deste nó por tempo indeterminado (BENET, 2014).

Neste projeto, cada *launcher*[1] se comportará como um nó do protocolo. Ao baixar um jogo, o arquivo será fixado para não ser deletado localmente e, simultaneamente, será distribuído para os outros nós da rede, sem necessitar que uma autoridade central forneça os arquivos.

2.4 Trabalhos Relacionados

O modelo de licenças para a distribuição digital de jogos é comumente utilizada pelas plataformas centralizadas. Este modelo permite que as plataformas revoguem o acesso dos usuários aos conteúdos adquiridos, conforme estabelecido em seus termos de uso.

Casos documentados mostram que tais revogações ocorrem na prática. Em abril de 2024, a Ubisoft removeu o jogo *The Crew* das bibliotecas dos compradores após desligar seus servidores, impossibilitando o acesso mesmo para usuários que haviam adquirido o produto (FERDINAND, 2024). De forma similar, em junho de 2025, a Epic Games Store anunciou que removeria *Dark and Darker* das bibliotecas dos usuários em novembro de 2025, impossibilitando o acesso ao jogo, mesmo para aqueles que o haviam adquirido (CHALK, 2025). Esses casos deixam evidente a fragilidade do modelo de licenças revogáveis adotados por grande parte das plataformas de distribuição digital.

2.4.1 Steam

O Steam é uma plataforma gratuita de jogos para computadores e consoles portáteis (*steam deck*) que opera no modelo centralizado. Na plataforma os usuários podem comprar licenças de uso e executá-las diretamente de sua biblioteca.

A plataforma estabelece que os usuários adquirem apenas licenças de uso, não a propriedade real dos jogos (Valve Corporation, 2025a). Ou seja, essas licenças estão passíveis de serem revogadas mediante violação dos termos de uso ou descontinuação do produto.

Segundo Wilde (2021), a maioria dos desenvolvedores de jogos demonstra insatisfação com o modelo de taxa de 30%. A competição entre plataformas forçou mudanças na indústria. O Steam reduziu suas taxas, mas apenas para jogos que já ultrapassaram $10 milhões de dólares em receita. Em contraste, a Epic Games opera com uma taxa fixa de 12% para todos os desenvolvedores.

Diferente do modelo centralizado do Steam, a arquitetura proposta neste trabalho utiliza tokens não fungíveis (NFTs) para representar a posse real dos jogos adquiridos. A

[1] Um *launcher* de jogos é uma aplicação que serve como interface para gerenciar e executar jogos eletrônicos, contém funcionalidades como biblioteca de jogos, instalação e inicialização.

representação dos ativos como NFTs garante ao comprador o controle direto e verificável sobre seus jogos através da blockchain, eliminando a possibilidade de revogação pela plataforma.

O Steam utiliza técnicas de proteção contra a violação dos direitos autorais chamado DRM (*Digital Rights Management*), que consiste em mecanismos aplicados nos executáveis dos jogos para a verificação da propriedade antes de sua execução. Contudo, segundo a própria documentação (Valve Corporation, 2025b), a solução pode ser facilmente removível, cabendo aos desenvolvedores implementarem proteções adicionais caso desejem maior segurança.

2.4.2 Blizzard Entertainment

A Blizzard Entertainment é uma desenvolvedora e distribuidora de jogos eletrônicos conhecida por jogos populares como *World of Warcraft*, *StarCraft* e *Diablo*. A empresa utilizou tecnologia P2P baseada no protocolo BitTorrent para a distribuição de *patches* e atualizações através do *Blizzard Downloader* (REKHI, 2025). Nessa implementação, cada cliente do *launcher* Battle.net atuava como *peer*, contribuindo para a distribuição de atualizações de forma descentralizada, reduzindo a carga nos servidores centralizados da empresa.

O presente trabalho se diferencia ao utilizar distribuição P2P baseada no IPFS, e principalmente por representar as licenças como NFTs na Blockchain. Essa abordagem garante apropriação real e não revogável dos ativos digitais pelos usuários, independentemente da plataforma ou de suas políticas, diferentemente do modelo de licenças temporárias adotado pela Blizzard conforme os termos de uso da plataforma (Blizzard Entertainment, 2023).

2.4.3 Fractal

O Fractal (Fractal, 2025a) é um *marketplace* (mercado) de jogos Web3 onde os usuários podem descobrir, comprar e vender NFTs e outros ativos digitais de jogos baseados em Blockchain.

Embora o Fractal utilize blockchain para certificação de propriedade dos NFTs, os jogos são distribuídos através de infraestrutura centralizada da plataforma (Fractal, 2025b), utilizando CDN[2] para a distribuição dos binários.

A diferença entre os projetos está no escopo dos NFTs: no Fractal, os NFTs representam itens opcionais dentro dos jogos (modificações, vantagens, conteúdo adicional),

[2] *Content Delivery Network* (CDN) é uma rede de entrega de conteúdo, utilizando-se de um conjunto de computadores dispersos geograficamente para acelerar a entrega de conteúdos da Web para usuários de todo o mundo.

sendo que o acesso aos jogos não depende da posse desses ativos. Já no presente trabalho, os NFTs são as próprias licenças não revogáveis dos jogos, sendo obrigatórios para sua execução e validados na blockchain antes do lançamento pelo *launcher*.

2.4.4 Elixir Games

A Elixir Games (Elixir Games, 2025) é uma plataforma e estúdio de publicação de jogos Web3, possuindo um cliente (Elixir Launcher) semelhante a plataformas como Steam ou Epic Games Store, que permite aos usuários acessar, baixar e jogar jogos baseados em Blockchain e tradicionais suportados pela Elixir, incluindo a integração de carteiras de criptomoedas e gestão de ativos NFT.

Embora a plataforma utilize NFTs para a representação de ativos colecionáveis dentro dos jogos, as licenças dos próprios jogos ainda podem ser revogadas. Conforme os termos de uso da plataforma, os usuários estão sujeitos a terem seu acesso encerrado, perdendo o acesso aos jogos adquiridos (Elixir Games, 2024).

O presente projeto se diferencia principalmente por representar a propriedade dos jogos como NFTs, garantindo apropriação irrevogável, e pela distribuição descentralizada dos arquivos via rede IPFS. Dessa forma, o usuário mantém posse de seus ativos digitais mesmo em caso de descontinuação da plataforma, uma vez que a propriedade está registrada na Blockchain e os arquivos propagados pela rede distribuída.

3 Materiais e Método

3.1 Materiais

- HTML[1]: *HyperText Markup Language* não é uma linguagem de programação, mas sim uma linguagem de marcação. Isto significa que é possível utilizá-lo para estruturar documentos a partir de marcações que determinem componentes como cabeçalho, listas, tabelas, entre outros (ROBBINS, 2018).

- CSS[2]: *Cascading Style Sheets*: O CSS é uma linguagem utilizada para descrever a apresentação visual dos documentos em HTML ou XML, descrevendo como cada elemento deve ser renderizado na tela (Mozilla Developer, 2025).

- Tailwind CSS[3]: Tailwind CSS é um framework CSS que utiliza o padrão *utility-first* para a estilização do HTML que adota princípios de composição no lugar de herança, facilitando a reutilização de classes e agilizando o desenvolvimento (TailwindCSS, 2025).

- Javascript[4]: Javascript é uma linguagem de programação que hoje está presente na grande maioria dos sites e navegadores modernos. Pode ser executado em computadores de mesa, consoles, tablets e smartphones (FLANAGAN, 2012). Dentre suas diversas funcionalidades, neste projeto foi utilizado para adicionar comportamentos no HTML através da biblioteca React, comunicar-se com a blockchain e a criação do servidor centralizado.

- TypeScript[5]: O TypeScript é uma linguagem de programação de código aberto criada pela Microsoft. e é conhecido também por ser um *Super set* (Superconjunto) de javascript (GOLDBERG, 2022). A linguagem engloba toda a sintaxe já existente do javascript, além de uma sintaxe própria para definir os tipos de dados. Seus principais componentes incluem um verificador de tipos que ajuda os desenvolvedores a identificarem quando algo está errado em seu código, um compilador que gera o código equivalente em javascript e também relata quaisquer problemas de compilação encontrados, e um serviço de linguagem que fornece utilitários para integrações a outras ferramentas que os desenvolvedores possam utilizar como, por exemplo, o editor de código (GOLDBERG, 2022).

[1] https://developer.mozilla.org/pt-BR/docs/Web/HTML
[2] https://developer.mozilla.org/pt-BR/docs/Web/CSS
[3] https://tailwindcss.com/
[4] https://developer.mozilla.org/pt-BR/docs/Web/JavaScript
[5] https://www.typescriptlang.org/docs/

- PNPM[6]: PNPM é um gerenciador de pacotes javascript como alternativa aos seus similares, NPM e Yarn. A principal diferença entre seus concorrentes, é sua arquitetura, no qual procura otimização de espaço. Segundo a análise comparativa de (LIMA, 2025) o PNPM apresentou o maior equilíbrio entre os demais, apresentando um bom desempenho, menor consumo de memória em disco, boa taxa de resolução de vulnerabilidades (segurança) e compatibilidade com o ecossistema javascript.

- React[7]: React é uma biblioteca para construir interfaces web ou nativas utilizando javascript, ou TypeScript. Segundo os criadores, com React é possível criar pedaços individuais (componentes) que podem ser agrupados e reutilizados em diferentes partes da aplicação para formarem páginas, telas e aplicações inteiras (React,).

- Rust[8]: Rust é uma linguagem de programação de código aberto projetada para que desenvolvedores possam escrever softwares mais rápidos e confiáveis. Oferece o controle de implementações de baixo nível, e com o seu sistema de propriedade, permite que a linguagem forneça garantias de segurança de memória sem a necessidade de um coletor de lixo. Além disso, o código é inspecionado em tempo de compilação através do *borrow checker*, garantindo que suas regras de propriedade e empréstimo sejam cumpridas (KLABNIK; NICHOLS, 2018).

- Cargo[9]: Cargo é o gerenciador de pacotes da linguagem Rust, com ele é possível fazer o *download* de pacotes e suas dependências, compilá-los e distribuí-los através do *package registry* crates.io (repositório de pacotes para o Rust).

- MongoDB[10]: MongoDB é um banco de dados NoSQL orientado a documentos, que armazena dados em formato BSON (*Binary JSON*). Sua estrutura flexível permite o armazenamento de documentos com esquemas variáveis, facilitando a persistência de metadados com diferentes atributos. Neste projeto, foi utilizado para armazenar os dados do servidor indexador centralizado que correspondem aos metadados dos NFTs.

- Tauri[11]: Tauri é um *framework* para a criação de binários para todas as principais plataformas de área de trabalho ou móveis. Sendo ele agnóstico a tecnologia *frontend*, com ele é possível que os desenvolvedores utilizem qualquer tecnologia que compile para HTML, javascript e CSS. Segundo os desenvolvedores, seus principais diferenciais são uma fundação segura para construir aplicações, um menor *bundle size* (tamanho total do pacote de código e recursos de uma aplicação) (Tauri, 2025).

[6] https://pnpm.io/pt/motivation
[7] https://react.dev/
[8] https://rust-lang.org/pt-BR/tools/install/
[9] https://doc.rust-lang.org/cargo/guide/
[10] https://www.mongodb.com/
[11] https://tauri.app/

- Blockchain Solana[12]: Segundo a comparação realizada por (MASTORAKIS, 2025), a rede solana se baseia no modelo *read-write aware*[13], no qual permite a execução paralela de transações de contratos inteligentes por meio da premissa de que o sistema conhece antecipadamente os estados específicos da blockchain que uma transação irá acessar e modificar. A comparação demonstra que a rede Solana pode chegar aproximadamente a 65.000 transações por segundo (contra 30 pela Ethereum), além de possuir taxas de transações mais baixas, o que torna Solana uma melhor escolha para o projeto.

- IPFS[14]: O IPFS (*InterPlanetary File System*) é um sistema de armazenamento de arquivos distribuídos peer-to-peer (P2P), como alternativa aos cliente-servidores tradicionais HTTP, sua proposta descentralizada permite o armazenamento e recuperação de dados sem a necessidade de um servidor centralizado. A distribuição por diferentes nós da rede, torna a tecnologia resiliente a censura e perda de dados, além de garantia de imutabilidade, pois quaisquer alterações em seus arquivos, resultaria em uma hash diferente da original. O que o torna adequado para a Web3 (POKHREL et al., 2023).

3.1.1 Hardware utilizado

Para o desenvolvimento e realização dos testes de validação do sistema, foram utilizados dois computadores com configurações diferentes:

Tabela 2 – Especificações dos computadores utilizados

Componente	**Computador 1**	**Computador 2**
Sistema Operacional	macOS	Windows 11
Processador (CPU)	Apple M3 Pro	AMD Ryzen 5 4500
Memória RAM	36GB	24GB
Arquitetura	ARM64 (Apple Silicon)	x86_64

Fonte: Próprio autor

O Computador 1 (macOS) foi utilizado tanto para o desenvolvimento do sistema quanto para os testes, enquanto o Computador 2 (Windows) foi utilizado apenas para os testes de validação. A utilização de computadores com diferentes sistemas operacionais e arquiteturas permitiu validar a compatibilidade multiplataforma do *launcher* e verificar o funcionamento da distribuição *peer-to-peer* em ambientes distintos.

[12] https://solana.com/pt

[13] Antes da execução de uma transação, os clientes devem especificar quais contas serão lidas ou escritas. Isso significa que antes da execução a rede já sabe quais contas bloquear para evitar problemas de concorrência

[14] https://ipfs.tech/

3.2 Método

A metodologia adotada consistiu no desenvolvimento de um protótipo funcional (*Minimum viable product* - MVP) para validar a viabilidade técnica e as funcionalidades essenciais de uma arquitetura descentralizada para a distribuição e apropriação de jogos eletrônicos.

3.2.1 Processo de Desenvolvimento

O desenvolvimento do projeto foi realizado utilizando uma abordagem iterativa, permitindo ajustes contínuos com base nos resultados obtidos em cada etapa. A organização das tarefas, o registro de dúvidas, anotações importantes e trechos relevantes de artigos consultados foram armazenados na ferramenta Notion.

O acompanhamento foi realizado através de reuniões semanais com o orientador. Inicialmente, as reuniões foram dedicadas à definição do escopo, refinamento dos objetivos e discussão das abordagens técnicas a serem adotadas. Durante a fase de implementação, o foco foi a apresentação dos resultados obtidos e discussão de soluções para os problemas técnicos encontrados. Por fim, as reuniões foram concentradas na definição dos critérios de validação, planejamento e execução dos testes. Esse processo permitiu ajustes contínuos no direcionamento do projeto, garantindo alinhamento entre o desenvolvimento e os objetivos estabelecidos.

3.2.2 Definição dos requisitos funcionais

Foram determinados os requisitos mínimos para o funcionamento da aplicação, abordando:

- Autenticação dos usuários.
- Representação dos jogos como *tokens* não fungíveis (NFTs).
- *Download* e *upload* descentralizado dos arquivos utilizando o IPFS (*InterPlanetary File System*).
- Gestão dos metadados dos jogos através de indexação centralizada opcional para otimização de buscas e filtros.
- Validação e execução dos jogos por meio do *launcher*.

3.2.3 Modelagem da arquitetura

Para a construção da arquitetura, foram criados diagramas arquiteturais claros, especificando a integração entre os componentes descentralizados (*blockchain* e IPFS),

as responsabilidades atribuídas ao servidor centralizado e os fluxos de interação com o *launcher*.

3.2.4 Desenvolvimento do protótipo

O protótipo foi desenvolvido de forma modular, com etapas bem definidas, iniciando-se pela criação dos contratos inteligentes (*smart contracts*) na *blockchain* para gerenciar as propriedades dos jogos. Em seguida, foram desenvolvidos os serviços complementares como cadastro, autenticação e compra. Posteriormente, foi realizada a integração com IPFS para receber e enviar os pacotes dos jogos de forma distribuída e, para finalizar, a criação do aplicativo *desktop*, responsável por integrar o sistema de ponta a ponta, permitindo aos usuários adquirir, baixar e executar os jogos.

3.2.5 Testes

Para validar a viabilidade técnica e os objetivos propostos, foram realizados testes funcionais abrangendo: o ciclo completo de interação desde a publicação até a execução de um jogo; a resiliência do sistema na ausência do servidor indexador; o funcionamento da distribuição *peer-to-peer* via IPFS entre múltiplos *launchers*; a análise comparativa de *overhead* entre os protocolos IPFS e HTTP; a interoperabilidade e apropriação real dos ativos através de carteiras externas; e a viabilidade econômica das transações na *Blockchain* Solana.

4 Desenvolvimento

Este capítulo apresenta o processo de desenvolvimento do sistema, detalhando desde o planejamento até a implementação da arquitetura proposta. Inicialmente, são apresentados os *mockups* de baixa fidelidade para guiar a construção das interfaces, seguidos pela descrição da arquitetura a ser implementada e suas integrações. Em seguida, são abordados os aspectos técnicos essenciais para o funcionamento do sistema: a configuração da *Candy Machine* para cunhagem automatizada de NFTs, a inicialização da *ledger* local para testes e desenvolvimento, e a execução do nó IPFS através da aplicação *desktop Tauri Sidecar*.

O desenvolvimento prossegue com a descrição da elaboração de uma prova de conceito para a validação da viabilidade técnica, seguida pela implementação do *marketplace* para publicação e aquisição de jogos, o desenvolvimento da biblioteca para gerenciamento dos ativos adquiridos, e o módulo de execução com suporte a múltiplas plataformas. Cada seção demonstra como os conceitos e tecnologias apresentados no capítulo anterior foram aplicados na construção de um protótipo funcional do sistema.

4.1 Planejamento e Mockup

Para o planejamento do sistema, foram coletadas capturas de tela de exemplos de plataformas similares que já existem, tendo sido validadas no mercado de games atualmente. O design da aplicação foi pensado em adotar os mesmos padrões de seus similares para facilitar a integração de novos usuários e conveniência ao manter o *layout* similar àqueles que já são utilizados.

Após a coleta de exemplos das aplicações já existentes, iniciou-se o processo de elaboração dos *mockups* utilizando-se a ferramenta Figma[1]. Durante toda a navegação do sistema, haverá *links* de cabeçalho para a navegação do usuário entre as principais áreas do sistema, sendo elas a tela inicial e a biblioteca. Haverá também botões de acesso ao perfil, depósito e publicação de um novo jogo.

As Figuras 2 e 3 demonstram a ideia do layout inicial do *marketplace*. A tela será composta por um banner principal para destaques estratégicos e um texto de apoio, com o propósito de incentivar a utilização da plataforma. Logo abaixo, foi adicionado um *grid* com o jogo destaque e a última seção da tela inicial é a seção de jogos, onde os usuários verão uma lista, com opções de filtros e buscas para os jogos listados.

A Figura 4 apresenta a biblioteca, que inclui uma listagem simples mostrando

[1] https://www.figma.com/

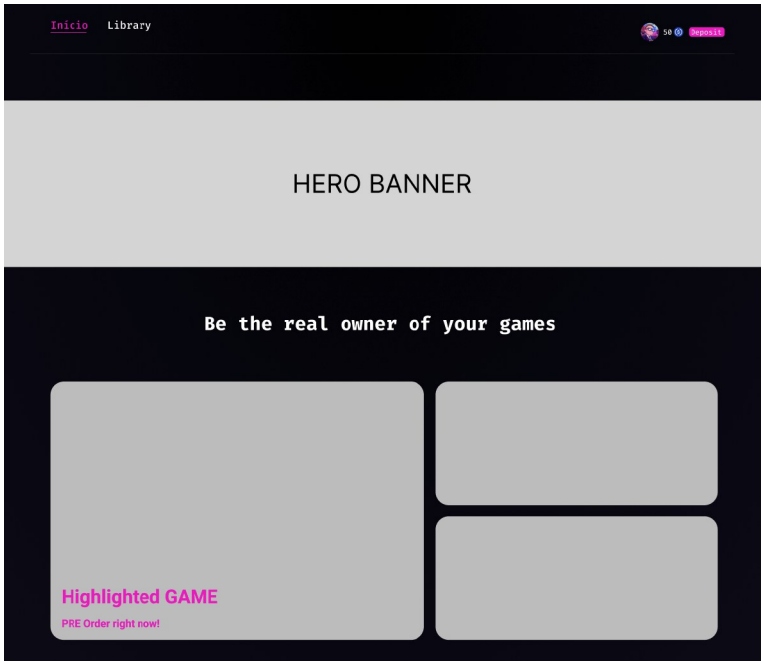

Figura 2 – Mockup da tela principal do marketplace

Fonte: Próprio autor

Figura 3 – Mockup da listagem de jogos do marketplace

Fonte: Próprio autor

todos os jogos do usuário com as opções de filtro entre instalados e todos os disponíveis, além de uma barra com os jogos favoritos e um campo de busca.

4.2 Arquitetura

O sistema completo será distribuído em um *launcher* que será por onde o usuário terá acesso à *blockchain* e aos jogos comprados, podendo baixá-los e executá-los. Também existirá um servidor central, responsável por fazer a indexação e *caching* das informações para uma melhor usabilidade e experiência mais rápida.

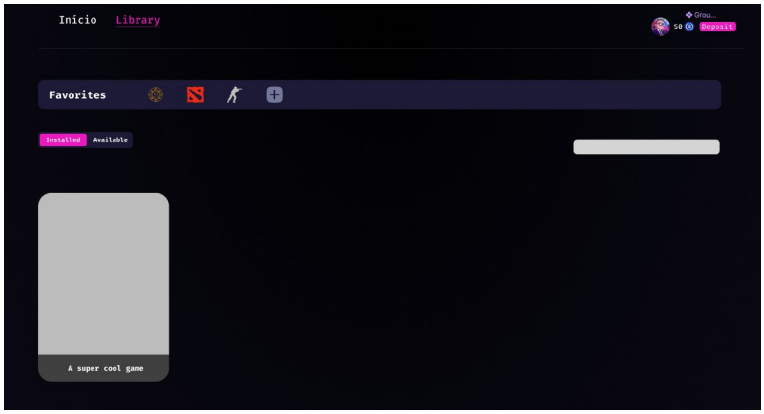

Figura 4 – Mockup da tela da biblioteca de jogos

Fonte: Próprio autor

O armazenamento dos arquivos será realizado através da rede IPFS de forma que os arquivos dos jogos sejam feitos de forma distribuída, independente de um servidor central; contudo, o servidor da aplicação ainda será um nó da rede, para que sempre haja ao menos um *peer* propagando os arquivos dos títulos.

A propriedade dos títulos será registrada na *blockchain* Solana para os pagamentos e criação dos *NFTs* utilizando-se a *Candy Machine* da *Metaplex Foundation*.

O cliente será construído como uma aplicação desktop sobre o Tauri, o qual cria uma *Web View* para a interface ser construída de forma agnóstico a tecnologia. Para tal fim, foi utilizado ReactJS com Tailwind para a estilização e elaboração das interfaces.

A integração entre esses componentes resulta em uma arquitetura que combina elementos descentralizados para as operações críticas de propriedade e distribuição, com elementos centralizados (opcionais) para buscas e filtros do catálogo de jogos. A Figura 5 apresenta a visão geral dessa arquitetura, ilustrando os fluxos de comunicação entre os componentes do sistema.

Conforme ilustrado, a arquitetura é composta por quatro camadas principais. À esquerda, a camada da *blockchain* Solana é responsável pelo gerenciamento de licenças através do programa *Candy Machine* da Metaplex, que permite a criação de coleções de NFTs para cada jogo publicado e a cunhagem automatizada de licenças.

A camada da aplicação, representada em amarelo, engloba os três módulos principais: o *marketplace*, responsável pelo catálogo e busca de jogos; o módulo de execução, que gerencia a inicialização dos jogos baixados; e a biblioteca, que sincroniza com a *blockchain* Solana os jogos adquiridos através da verificação dos NFTs presentes na carteira do usuário.

O nó IPFS local, executado através do Kubo em *sidecar*, atua como ponte entre a aplicação e a rede distribuída. Esse componente é responsável por armazenar os arquivos dos jogos baixados (*pinning*) e propagá-los para outros nós da rede, permitindo que cada

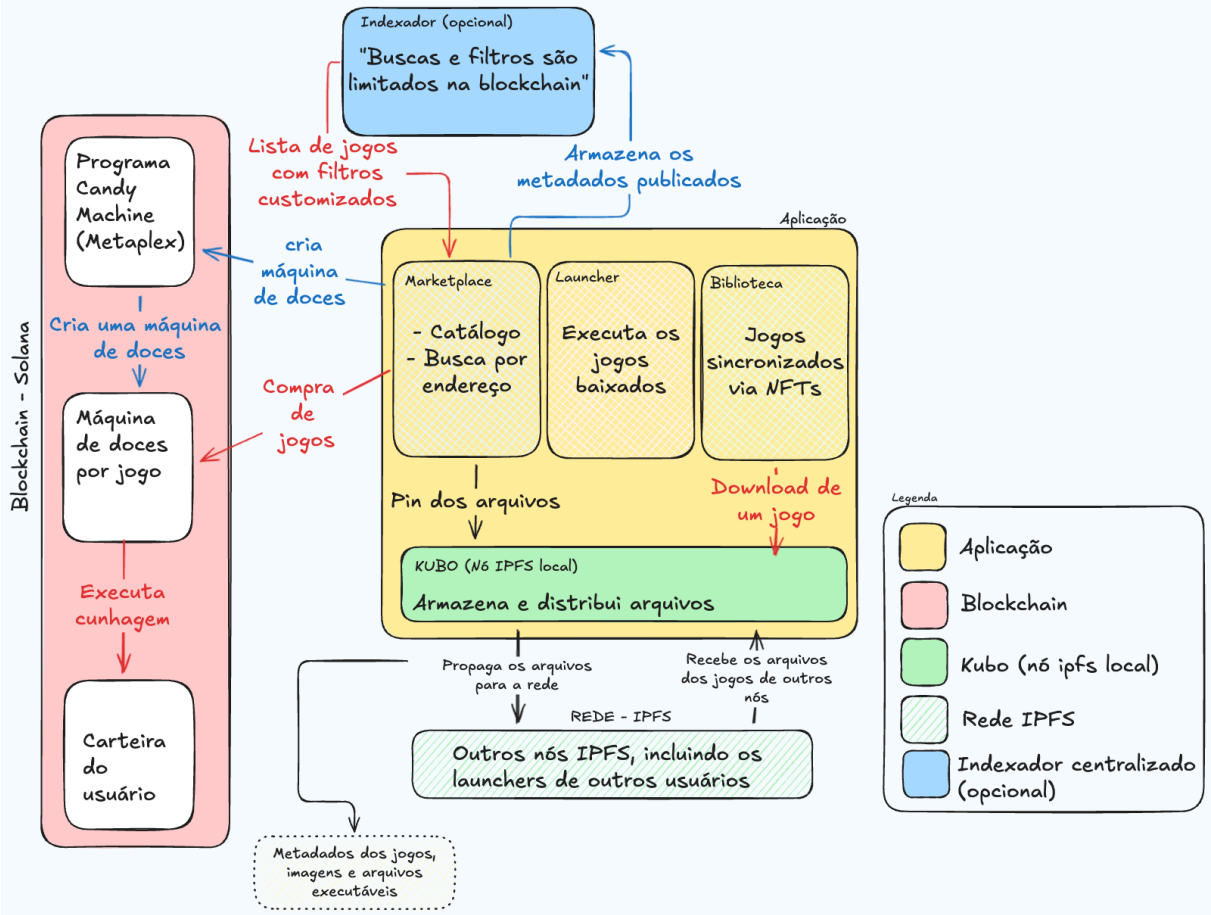

Figura 5 – Arquitetura geral do sistema proposto

Fonte: Próprio autor

launcher contribua para a distribuição descentralizada do conteúdo.

Por fim, o indexador centralizado, representado em azul, é um componente opcional que fornece buscas e filtros customizados para o catálogo de jogos. Sua natureza opcional garante que o sistema permaneça funcional mesmo em sua ausência, preservando a característica descentralizada da arquitetura.

4.3 Candy Machine

No contexto deste projeto, a *Candy Machine* foi utilizada para automatizar a distribuição de licenças. Cada jogo publicado no *marketplace* corresponde a uma *Candy Machine* configurada com os metadados do jogo e o preço definido pelo desenvolvedor.

Para implementar o pagamento obrigatório ao desenvolvedor, foi configurado o guarda *solPayment* no momento da criação da *Candy Machine*, conforme ilustrado na Figura 8. Dessa forma, toda cunhagem de licença é realizada apenas mediante pagamento à carteira do desenvolvedor.

4.4 *Testing ledger*

Registrar transações na rede Solana exige custos de gás e taxas de prioridade, e para fins de desenvolvimento, foi iniciado um validador local utilizando a linha de comando fornecida pela Solana Foundation. Dessa forma, é possível fornecer *Airdrops*[2] para ser possível executar os pagamentos das taxas e também a compra de ativos em modo teste sem custos reais.

A *ledger* padrão possui apenas os contratos nativos da rede previamente disponíveis. Para utilizar o protocolo Metaplex (necessário para NFTs e *Candy Machines*), é preciso fazer o *dump* (baixar o binário) dos contratos diretamente da rede e disponibilizá-los localmente. Para isso, foram utilizados os comandos executados conforme o Apêndice B. E para iniciar a *ledger* com os contratos baixados localmente, é possível utilizar o comando conforme o Apêndice C.

Após a execução do último comando, um nó da rede estará disponível localmente, podendo ser acessado via JSON RPC através da url: `http://127.0.0.1:8899`

4.5 Nó IPFS e Tauri Sidecar

Para que o *launcher* possa se comunicar com a rede IPFS e propague os dados para a rede, é necessário que a aplicação também forneça um cliente que faça a integração com o protocolo. Foi utilizado o Kubo[3] (IPFS) em sidecar, fornecendo as mesmas funcionalidades de um nó juntamente com o executável final. O binário é compilado especificamente para uma arquitetura de sistema operacional, e a seleção do binário correto para a plataforma em utilização (macOS, Windows, Linux) é feita automaticamente pelo Tauri, desde que os binários fornecidos contenham como sufixo o identificador da arquitetura (*e.g.* aarch64-apple-darwin).

Durante a execução do *launcher* é necessário que o processo do Kubo esteja em execução, e para isso, durante a inicialização da aplicação, é executado o processo para estar sempre ativo. Além disso, como a aplicação é baseada em um navegador local, é necessário configurar as regras de CORS (Cross-Origin Resource Sharing) no nó IPFS, para o navegador permitir o compartilhamento dos recursos no endereço local.

4.6 Prova de conceito

Para a elaboração da solução foi necessária uma primeira prova de conceito para validação da viabilidade técnica. Para isso, foi implementado um formulário simples para

[2] Airdrops: Distribuição de tokens gratuitamente
[3] Kubo é uma interface de linha de comando para executar nós IPFS

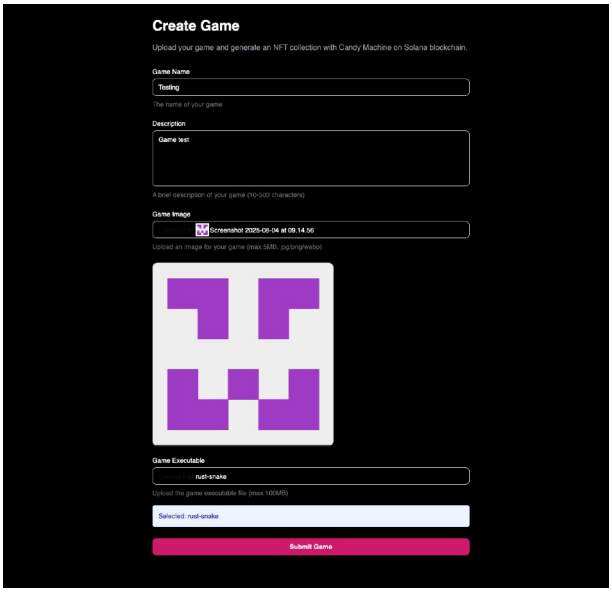

Figura 6 – Protótipo da tela de cadastro de um jogo

Fonte: Próprio autor

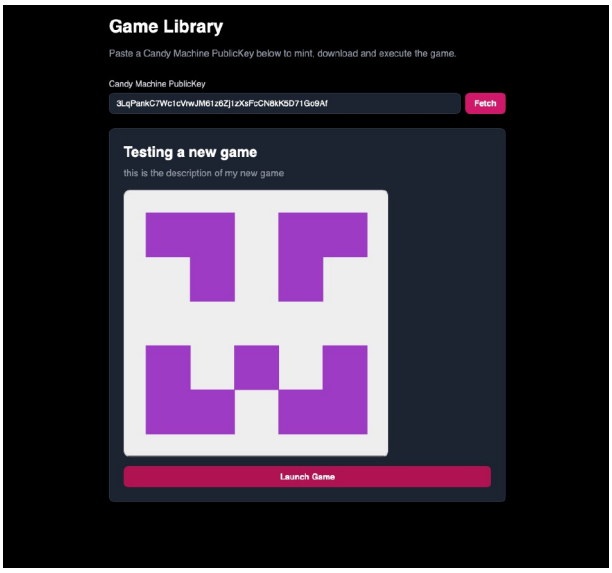

Figura 7 – Protótipo da versão inicial da biblioteca

Fonte: Próprio autor

o cadastro e *upload* de um jogo, gerando-se um NFT e armazenando seu arquivo binário na rede IPFS (Figura 6).

A segunda parte da elaboração da prova de conceito foi implementar a funcionalidade de download e execução do jogo localmente, através do IPFS. Para isso foi adicionado um campo de busca por endereço da Candy machine na *blockchain* conforme a Figura 7.

Ao encontrar uma Candy machine, é mostrado os metadados da coleção, incluindo a imagem, descrição e URL para download do conteúdo. Ao clicar no botão o jogo é iniciado, finalizando a prova de conceito.

A implementação da prova de conceito foi essencial para entender e testar as integrações entre diferentes componentes, para formar uma arquitetura distribuída que integre o armazenamento do arquivo binário e metadados dos NFTs, a cunhagem das licenças, execução e configuração do nó IPFS em um ambiente local com sidecar.

4.7 Servidor Indexador

Para melhorar a experiência de navegação e busca no *marketplace*, foi desenvolvido um servidor indexador centralizado responsável por armazenar e disponibilizar os metadados dos jogos publicados na plataforma. O servidor foi implementado como uma API REST utilizando a linguagem Rust, e o armazenamento dos dados foi realizado através do MongoDB, um banco de dados NoSQL orientado a documentos.

A API disponibiliza quatro rotas principais para: registro de novos jogos com seus metadados e endereço da *Candy Machine*, listagem de todos os jogos cadastrados, busca com filtros por nome, categorias e faixa de preço, e obtenção dos jogos em destaque para a tela inicial.

É importante ressaltar que o servidor indexador é um componente opcional da arquitetura. Todas as informações armazenadas no banco de dados são derivadas da *blockchain*, servindo apenas como uma camada de otimização para consultas. Caso o indexador esteja indisponível, o sistema permanece funcional através de consultas diretas à *blockchain* Solana.

4.8 Marketplace

Com a prova de conceito pronta, foi realizado novos ajustes para que a aplicação se tornasse uma loja de jogos. Foi adicionado uma nova tela de listagem para mostrar os jogos cadastrados, preços, executar a cunhagem de um novo NFT (licença), download e execução do jogo.

Para o cenário de uma loja de jogos, a cada cunhagem, é necessário previamente que a carteira que está comprando, faça um pagamento de um valor fixo para uma carteira de destino (desenvolvedor). Para isso, a Core Candy Machine (2025) fornece guardas (*guards*), que é um pedaço modular de código para restringir a cunhagem ou adicionar novas funcionalidades a Candy Machine.

Para adicionar novos guardas, basta adicionar a configuração do guarda (no caso de pagamentos, *solPayment*) no momento da criação da Candy Machine (exemplo conforme a Figura 8), e toda cunhagem de licença será realizada apenas mediante pagamento.

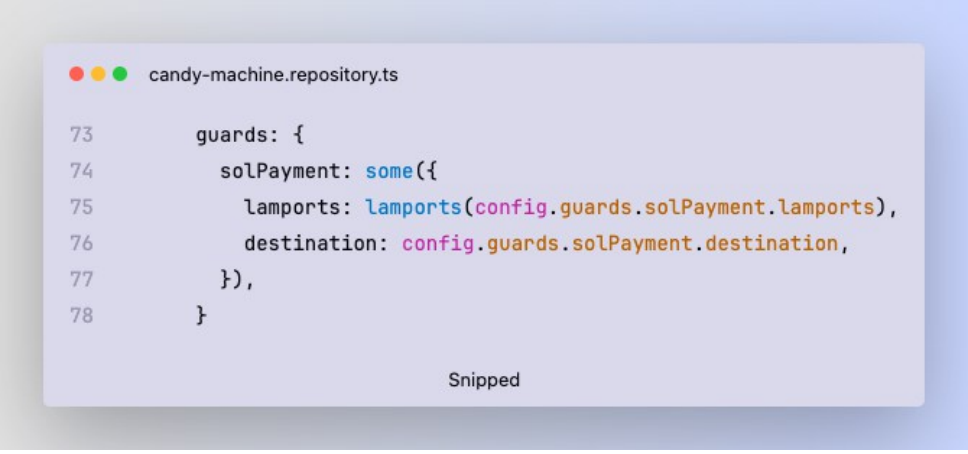

Figura 8 – Configuração do guarda de pagamento

Fonte: Próprio autor

4.9 Biblioteca

Após a criação da página da loja, foi necessário a adição de uma nova tela para a biblioteca. O usuário pode ver todas as suas licenças adquiridas e gerenciar instalações, ou executar um jogo direto pelo *launcher*.

Os arquivos dos jogos estão distribuídos e para realizar o download local dos arquivos é necessária uma chamada **cat** do IPFS, isto pode ser realizado através do nó IPFS local conforme a Figura 9.

ipfs.ts

```ts
async downloadFileStreaming(
  cid: string,
  onChunk: (chunk: Uint8Array, loaded: number, total: number) => Promise<void>
) {
  const url = `http://localhost:5001/api/v0/cat?arg=${cid}`;
  const response = await fetch(url, {
    method: "POST",
  });

  ...

  const reader = response.body.getReader();
  let downloaded = 0;

  while (true) {
    const { done, value } = await reader.read();

    if (done) break;
    downloaded += value.length;

    await onChunk(value, downloaded, total);
  }
}
```

Snipped

Figura 9 – Função para download de arquivo via IPFS

Fonte: Próprio autor

Após o download, ainda é necessário executar o *pinning* dos arquivos para permanecerem no nó local por tempo indeterminado, tornando-o *seeder* para outros usuários da plataforma, conforme a Figura 10.

ipfs.ts

```ts
async pinFile(cid: string): Promise<void> {
  const url = `http://localhost:5001/api/v0/pin/add?arg=${cid}`;

  const response = await fetch(url, {
    method: "POST",
  });

  if (!response.ok) {
    throw new Error(
      `Failed to pin file: ${response.status} ${response.statusText}`
    );
  }

  const result = await response.json();
  console.log("File pinned successfully:", result);
}
```

Snipped

Figura 10 – Função para o *pinning* de um arquivo

Fonte: Próprio autor

4.10 Execução

Os executáveis dos jogos são desenvolvidos especificamente para determinada arquitetura. Para funcionar no ambiente correto que está a ser executado, foi adicionado uma nova seção na tela de cadastro de jogos, para os desenvolvedores fornecerem binários específicos para tipos diferentes de arquitetura (Figura 11)

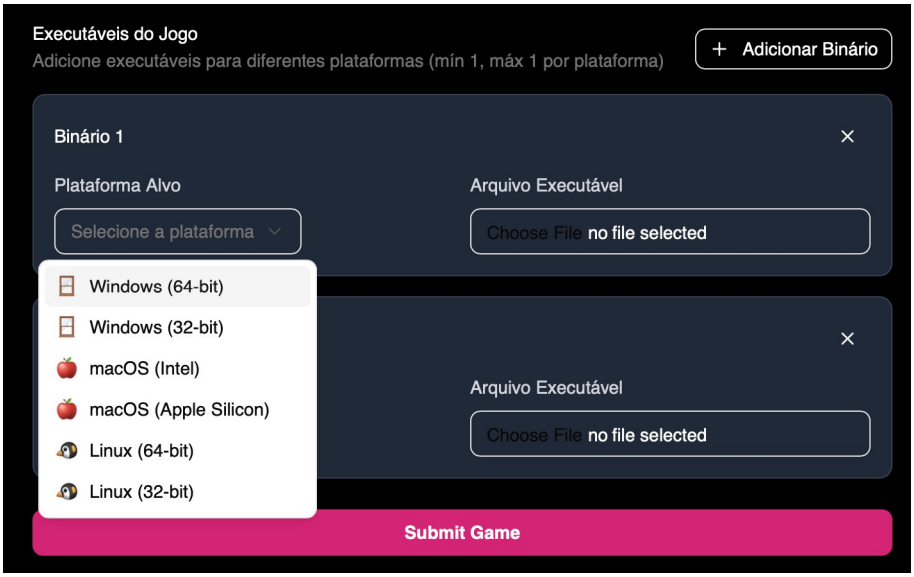

Figura 11 – Seleção de múltiplas plataformas

Fonte: Próprio autor

Desta forma, na biblioteca, é possível a aplicação identificar a plataforma atual, e baixar o binário correspondente e gerar os comandos específicos de execução para cada tipo de sistema (exemplo de comando para o macOS Apple Silicon conforme a Figura 12).

Figura 12 – Função para a execução de um jogo no macOS

Fonte: Próprio autor

5 Resultados

Como resultado deste trabalho, foi desenvolvido um protótipo funcional (MVP) de um *marketplace* descentralizado para distribuição e apropriação de jogos eletrônicos.[1] O sistema implementado integra tecnologias de Blockchain (Solana) e IPFS em uma arquitetura **híbrida**, atendendo aos objetivos propostos de estabelecer um modelo alternativo ao atual de licenças revogáveis. O sistema implementa o ciclo completo de interação do usuário: desde a autenticação via carteira digital, passando pela publicação e aquisição de jogos, até o download descentralizado do jogo adquirido e a sua execução local.

5.1 Arquitetura

A arquitetura implementada adota um modelo híbrido que combina componentes descentralizados e centralizados para equilibrar autonomia do usuário, desempenho e escalabilidade. Os principais componentes do sistema são:

Blockchain (Solana): Responsável pelo gerenciamento das licenças dos jogos via NFTs organizados em coleções. Cada jogo publicado corresponde a uma coleção, e cada compra resulta na cunhagem automatizada de um NFT via uma *Candy Machine*, representando a licença daquele jogo para o proprietário.

IPFS (InterPlanetary File System): Utilizado para a distribuição descentralizada dos arquivos dos jogos. Os executáveis e metadados são armazenados e propagados através da rede. Cada *launcher* instalado atua como um nó, contribuindo para a disponibilidade e distribuição dos conteúdos.

Servidor centralizado (indexador): Fornece serviços de indexação e otimização para melhorar a experiência do usuário. Mantém um índice dos jogos disponíveis e seus metadados para consultas rápidas, além de atuar como um nó IPFS disponível, garantindo ao menos um ponto de origem para download dos arquivos. O sistema permite que os usuários alterem o endereço do indexador através das configurações do *launcher*, garantindo autonomia em caso de falhas ou por preferência pessoal.

Launcher: Aplicação *desktop* que integra todos os componentes da arquitetura, servindo como interface unificada para o usuário. Permite a autenticação via carteira digital, navegação e compra de jogos, download através do IPFS, validação de propriedade e execução dos jogos adquiridos.

A integração entre esses componentes permite que o sistema opere de forma descentralizada para as operações críticas de propriedade e distribuição, enquanto mantém

[1] Vídeos demonstrativos do sistema em funcionamento estão disponíveis no Apêndice D.

elementos centralizados opcionais para otimização de desempenho, resultando em uma solução que combina as vantagens de ambas as abordagens.

5.2 Autenticação e gestão de carteiras

O sistema implementa autenticação totalmente descentralizada, eliminando a necessidade de cadastro tradicional com dados pessoais ou servidores de autenticação centralizados. Na tela inicial, o usuário pode escolher entre gerar uma nova carteira ou importar uma existente através da chave privada, conforme ilustrado na Figura 13.

Figura 13 – Tela de autenticação via carteira

Fonte: Próprio autor

A carteira gerada ou importada permanece sob controle exclusivo do usuário, que pode exportar sua chave privada a qualquer momento através do menu principal (Figura 14). Isso garante a interoperabilidade do sistema, permitindo que o usuário utilize a mesma carteira em outras aplicações da rede Solana, realize transferências diretas ou interaja com outros contratos inteligentes, independentemente da plataforma desenvolvida neste trabalho. Essa abordagem atende ao objetivo de apropriação real dos ativos digitais, diferentemente do modelo proposto pelas plataformas centralizadas.

5.3 Representação de jogos como NFTs

O sistema implementa um modelo de representação no qual cada jogo publicado na plataforma corresponde a uma coleção de NFTs (*Collection*) na Blockchain Solana, e

Figura 14 – Exportação da chave privada

Fonte: Próprio autor

cada licença individual adquirida é representada por um NFT dentro dessa coleção.

Nas plataformas tradicionais como Steam e Epic Games, a compra de um jogo concede apenas uma licença de uso revogável, mantendo a plataforma como proprietária do ativo e intermediária obrigatória para o acesso. No modelo implementado neste trabalho, a propriedade do NFT confere ao usuário controle direto sobre o ativo digital. O ativo permanece na sua carteira e pode ser transferido, comercializado em mercados secundários ou utilizado fora da plataforma, independentemente do *marketplace* original ou de suas políticas.

A cunhagem dos NFTs ocorre automatizadamente via *Candy Machines*, eliminando a necessidade de intervenção manual dos desenvolvedores para cada venda realizada. Esse mecanismo garante escalabilidade ao processo de distribuição, permitindo que as compras ocorram simultaneamente, de forma independente e descentralizada. O modelo implementado atende ao objetivo central do trabalho de estabelecer apropriação real dos jogos adquiridos, transferindo o controle dos ativos dos intermediários centralizados para os usuários finais. A imutabilidade e auditabilidade da Blockchain garantem que a propriedade registrada não pode ser revogada unilateralmente pela plataforma.

5.4 Marketplace

O *marketplace* permite que desenvolvedores publiquem seus jogos mediante um processo que automatiza a criação da coleção e da *Candy Machine* correspondentes, permitindo a configuração de preço, nome, descrição e imagem, além da publicação de

executáveis para diferentes plataformas. O formulário de cadastro organiza os campos em uma coluna à esquerda e exibe uma visualização em tempo real de como o jogo aparecerá à direita, conforme ilustrado na Figura 15.

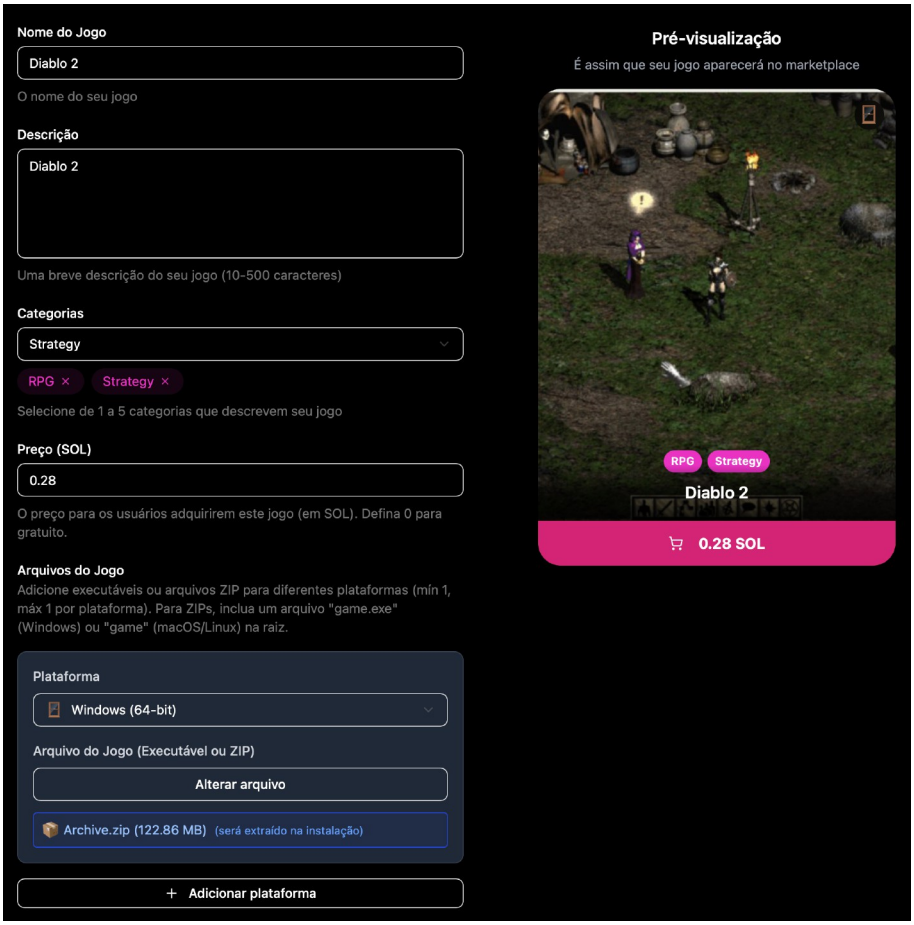

Figura 15 – Interface de publicação de jogos com visualização em tempo real

Fonte: Próprio autor

A navegação e pesquisa dos jogos disponíveis utilizam primariamente o indexador centralizado (*peer* de *boostrap* com catálogo filtrável via API), que mantém um catálogo otimizado para consultas rápidas e customizadas por filtros de preço, categorias ou nome, conforme ilustrado na Figura 16. Essa abordagem proporciona experiência de usuário similar às plataformas de jogos tradicionais.

Entretanto, o sistema possui resiliência através da consulta direta à Blockchain. Caso o indexador centralizado apresente falhas, os usuários ainda podem buscar jogos específicos utilizando o endereço da *Candy Machine* diretamente na interface do *launcher*, e realizar compras sem depender de intermediários. Além disso, os usuários podem até mesmo criar o próprio indexador ou escolher um existente, bastando alterá-lo nas configurações conforme ilustrado na Figura 17. Essa característica preserva a descentralização fundamental do sistema, mantendo o indexador como componente opcional para otimização de desempenho (catálogo indexado e filtrável via API) e filtros personalizados.

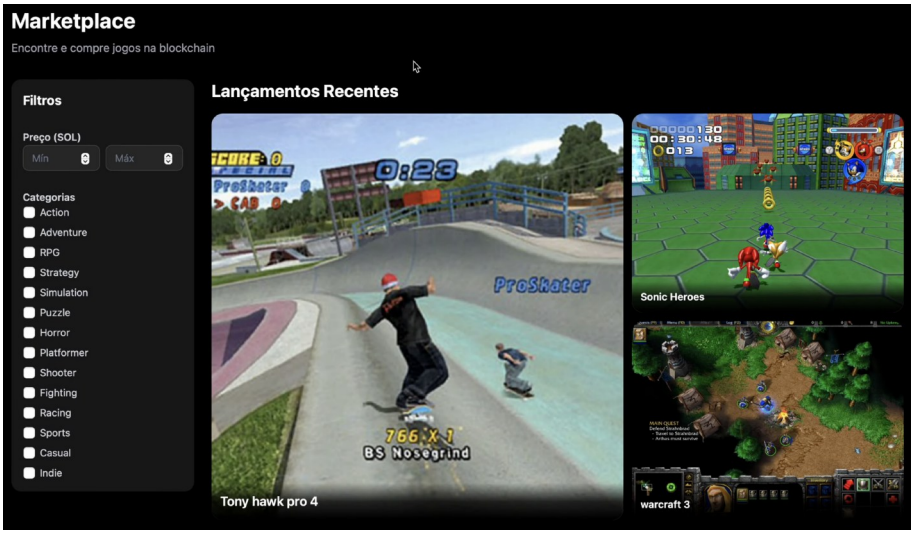

Figura 16 – Tela do marketplace

Fonte: Próprio autor

O modelo implementado atende ao objetivo de estabelecer um *marketplace* que combine usabilidade por meio de componentes centralizados opcionais com a autonomia dos usuários e resiliência proporcionadas pela descentralização do sistema, que continua funcional mesmo sem uma infraestrutura centralizada.

5.5 Biblioteca

A biblioteca implementada consulta diretamente a Blockchain para identificar os NFTs presentes na carteira do usuário autenticado, exibindo automaticamente os jogos de propriedade do usuário. O sistema organiza a visualização por filtros que permitem ao usuário navegar entre todos os jogos, apenas os já instalados ou apenas os não instalados.

Conforme ilustrado na Figura 18, a biblioteca também fornece opções para download e remoção dos jogos diretamente através da interface. O usuário pode baixar o jogo tanto imediatamente após a compra no *marketplace* quanto posteriormente através da própria biblioteca. Os arquivos são baixados via IPFS e, após a conclusão do download, o *pinning* é realizado automaticamente, fazendo com que o *launcher* passe a contribuir com a distribuição daquele jogo na rede P2P do IPFS. A remoção de jogos instalados libera espaço em disco local, excluindo os arquivos do jogo e o respectivo *pinning*, mas não afeta a propriedade do NFT, permitindo que o usuário realize um novo *download* a qualquer momento.

O sistema também implementa uma barra de acesso rápido aos jogos favoritos do usuário, permitindo iniciar a execução diretamente com um único clique. Para jogos ainda não instalados, um diálogo de confirmação é exibido com a opção de realizar o download, integrando as funcionalidades de biblioteca e distribuição em um fluxo simplificado.

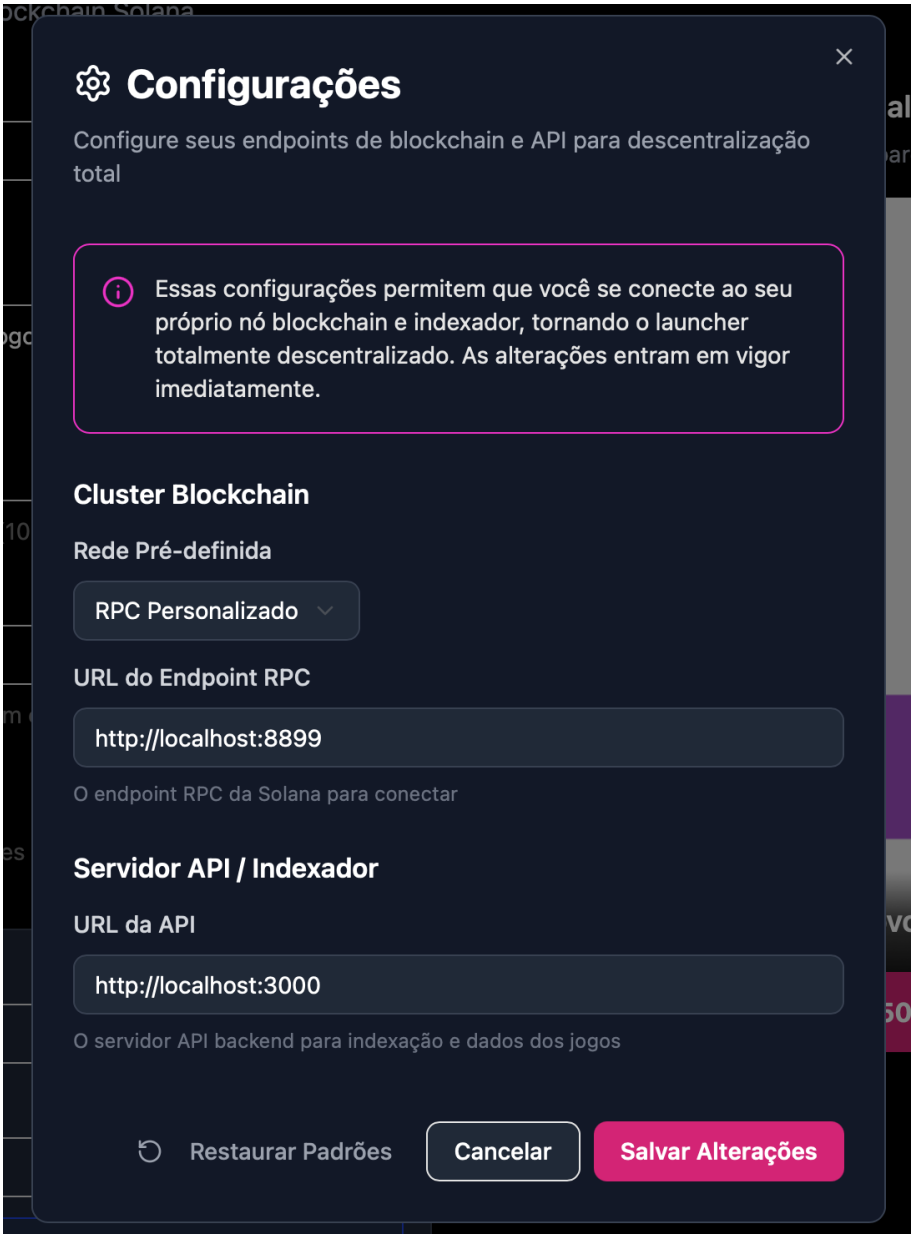

Figura 17 – Configurações da plataforma

Fonte: Próprio autor

Essa implementação garante que a biblioteca represente a real propriedade do usuário registrada na Blockchain. Essa característica elimina a dependência de servidores de autenticação centralizados e reforça o conceito de apropriação real dos ativos digitais, permitindo que o usuário tenha autonomia total dos seus ativos.

5.6 Testes e Validação

Para validar a viabilidade técnica e os objetivos propostos, foram realizados testes funcionais abrangendo os principais componentes do sistema e cenários de uso. Os testes foram realizados em dois computadores diferentes, com foco em três aspectos fundamen-

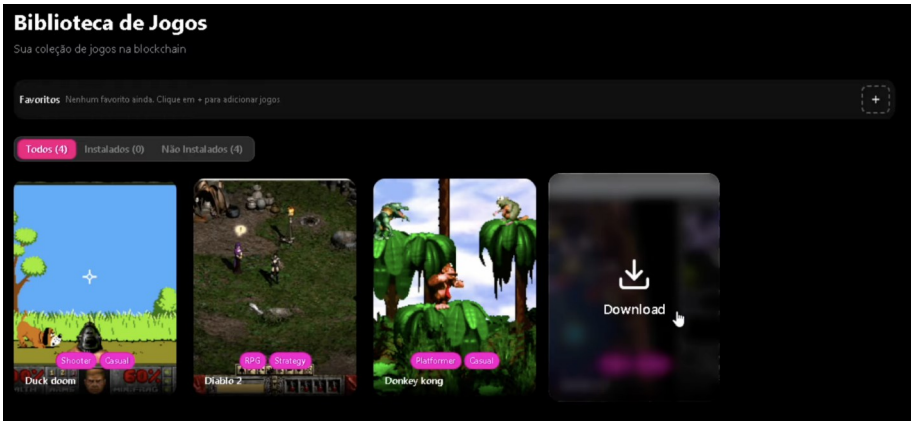

Figura 18 – Biblioteca de jogos

Fonte: Próprio autor

tais: (1) funcionamento do ciclo completo de interação, (2) resiliência da arquitetura descentralizada, e (3) validação de propriedade dos ativos.

Tabela 3 – Resumo dos testes realizados e seus resultados

Teste	**Objetivo**	**Resultado**
Ciclo Completo	Validar o fluxo desde a publicação até a execução do jogo	Fluxo concluído com sucesso em dois computadores distintos
Resiliência sem Indexador	Verificar funcionamento sem o servidor centralizado	*Marketplace* operacional via busca por endereço da *Candy Machine*
Distribuição P2P	Comprovar funcionamento da rede IPFS entre *launchers*	Download bem-sucedido tendo apenas um outro *launcher* como fonte (*seed*) do jogo
Portabilidade da Carteira	Testar acesso aos jogos adquiridos via *login* em diferentes dispositivos	Biblioteca sincronizada automaticamente via Blockchain
Visualização Externa	Verificar existência do NFT fora da plataforma	NFT visível e transferível pela carteira Phantom
Transferência de NFT	Validar apropriação real via transferência externa	Jogo acessível na nova carteira após transferência e passa a ser inacessível na carteira original.

Fonte: Próprio autor.

5.6.1 Validação do Ciclo Completo

Para a validação do ciclo completo do *marketplace*, foi criado um jogo utilizando uma carteira em um dos computadores. No outro computador, foi gerada uma nova carteira, adicionado saldo e, finalmente, foi executada uma compra. Após a compra, o

jogo apareceu na biblioteca do segundo computador, permitindo realizar o download e a execução. Após o fim da execução, foi possível desinstalar o jogo para liberar espaço. O *pinning* foi completamente removido, permitindo que o IPFS realize a limpeza dos arquivos quando necessário.

5.6.2 Resiliência sem Componentes Centralizados

O teste de resiliência foi realizado sem a execução do indexador centralizado, visando validar se o usuário conseguiria concluir todo o fluxo desde a compra até a execução sem a dependência do serviço. Sem o indexador, o *marketplace* não consegue realizar filtros nem buscas otimizadas, exibindo uma mensagem de erro conforme a Figura 19. Contudo, mesmo sem o serviço, ainda é possível trocar a URL para um indexador terceiro ou buscar um jogo diretamente pelo endereço da *Candy Machine* na Blockchain (Figura 20), e o usuário ainda pode utilizar o fluxo completo de compra e execução.

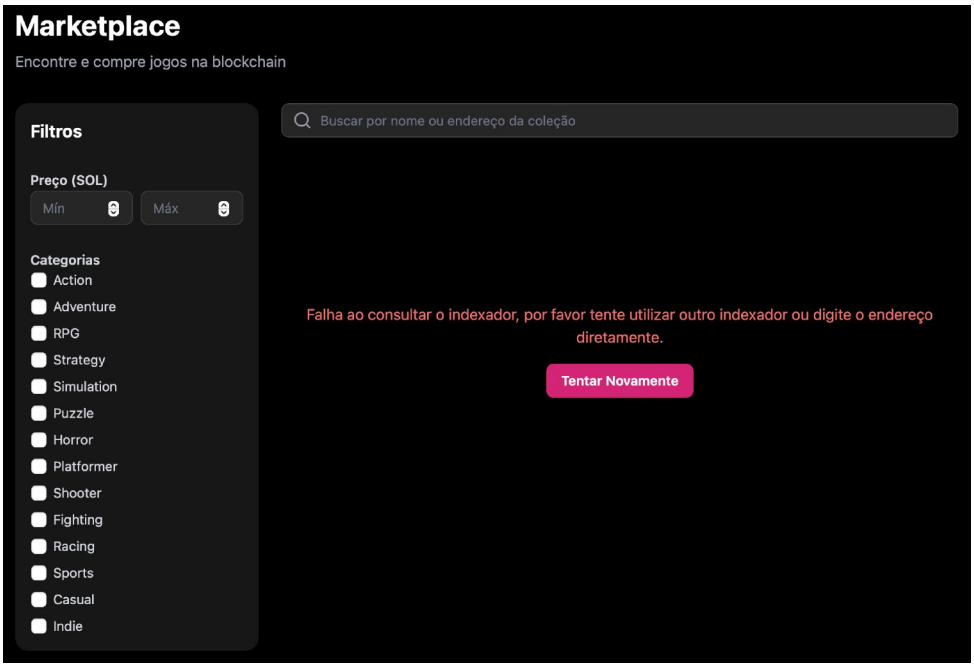

Figura 19 – Marketplace sem o indexador

Fonte: Próprio autor

5.6.3 Distribuição

Para validar o funcionamento da distribuição descentralizada, foi realizado um teste envolvendo dois computadores atuando como nós da rede IPFS, com o servidor de *bootstrap* desligado. O objetivo foi comprovar que o sistema opera efetivamente em modelo P2P, sem a necessidade obrigatória de servidores centralizados para a disponibilidade dos arquivos. O teste foi conduzido nas seguintes etapas:

Figura 20 – Busca por endereço Candy machine sem o indexador

Fonte: Próprio autor

1. Um jogo foi publicado no primeiro computador, tornando-o o único *peer* inicial disponível.
2. O segundo computador adquiriu e baixou o jogo, tornando-se um novo nó distribuidor.
3. O primeiro computador foi removido da rede após a exclusão de todos os arquivos locais do IPFS e a desinstalação dos jogos.
4. Utilizando uma nova carteira, o primeiro computador adquiriu o mesmo jogo e realizou o *download* com sucesso pela rede IPFS, tendo como único *peer* o segundo computador.

O teste demonstrou que o sistema opera realmente de forma descentralizada, com cada usuário compartilhando conteúdo entre si. A capacidade de download bem-sucedido com apenas um nó ativo comprova a viabilidade técnica da distribuição P2P.

5.6.4 Análise do protocolo

Para avaliar a viabilidade técnica da distribuição via IPFS, foi realizada uma análise comparativa do *overhead* do protocolo em relação ao HTTP tradicional. O objetivo desta análise não é comparar a taxa de transferência (que favoreceria injustamente um ou outro protocolo dependendo do número de *peers* ativos), mas sim quantificar o custo adicional do protocolo IPFS em termos de latência inicial (consulta a DHT).

Os testes foram conduzidos medindo o tempo até o primeiro *byte* (TTFB - *Time To First Byte*), que representa o intervalo entre o início da requisição e o recebimento

do primeiro *byte* de dados. Esta métrica isola o *overhead* de descoberta de *peers*, estabelecimento de conexão e negociação de protocolo, independentemente do tamanho do arquivo.

O teste foi realizado com:

- **Cliente:** Computador localizado no Brasil
- **Servidor:** Instância AWS EC2 localizada nos EUA (us-east-1)
- **Arquivo de teste:** 4MB
- **Protocolo HTTP:** Servidor Python via porta 8081
- **Protocolo IPFS:** Kubo CLI servindo o arquivo via API
- **Ferramenta de medição:** `curl` com métrica `time_starttransfer`

Cada protocolo foi testado três vezes para verificar a consistência dos resultados e a Tabela 4 apresenta os tempos medidos até o primeiro *byte* para ambos os protocolos. O comando utilizado para executar os testes foi: `curl -X POST -o /dev/null -w "%{time_starttransfer}" URL`

Tabela 4 – Comparação de latência inicial entre IPFS e HTTP

Execução	HTTP (seg.)	IPFS (seg.)	Overhead
1	0,291	3,725	12,78x
2	0,293	3,052	10,41x
3	0,272	3,165	11,63x
Média	**0,285**	**3,314**	**11,62x**

Fonte: Próprio autor

Os resultados demonstram que o protocolo IPFS introduz aproximadamente 3 segundos de latência adicional antes do início da transferência de dados. Este overhead é composto principalmente por:

1. **Consulta à DHT:** O nó cliente precisa consultar a *Distributed Hash Table* para descobrir quais *peers* possuem o conteúdo solicitado
2. **Estabelecimento de conexão:** Estabelecimento de canal de comunicação com o *peer* remoto

Em contraste, o HTTP apresenta latência de aproximadamente 285 ms, refletindo principalmente a latência de rede entre Brasil e Estados Unidos.

Para o contexto de distribuição de jogos, onde os arquivos geralmente possuem entre centenas de megabytes a dezenas de gigabytes, este acréscimo inicial de 3 segundos é negligenciável em relação ao tempo total de download.

É importante ressaltar que esta análise mede o comportamento com apenas um *peer* ativo. Em um cenário com múltiplos *peers* distribuindo o mesmo conteúdo, o IPFS possibilita download paralelo de diferentes blocos do arquivo a partir de múltiplas fontes simultaneamente, podendo superar significativamente a velocidade de download de servidores HTTP centralizados. Vale destacar que o protocolo IPFS foi em parte baseado em boas práticas estabelecidas no protocolo BitTorrent para disseminação eficiente de grandes arquivos, com escalabilidade. Portanto, o *overhead* inicial do IPFS em relação ao HTTP é compensado quando o número de *peers* aumenta.

5.6.5 Interoperabilidade e Apropriação Real

Para validar o objetivo central do trabalho de estabelecer a propriedade real dos ativos digitais, diferente do modelo de licenças revogáveis das plataformas tradicionais, foram realizados testes focados na portabilidade da carteira, visualização externa e transferência dos NFTs.

5.6.5.1 Portabilidade da Carteira

Foi testado o acesso aos jogos adquiridos em diferentes dispositivos utilizando a mesma carteira:

1. Exportação da chave privada através do *launcher* na primeira instalação
2. Instalação do *launcher* em computador diferente
3. Importação da mesma chave privada
4. Verificação da sincronização automática da biblioteca

O teste comprovou que a biblioteca sincroniza automaticamente com base nos NFTs presentes na Blockchain sem a necessidade de uma sincronização manual.

5.6.5.2 Visualização em Carteira Externa

Para comprovar a existência real do NFT na Blockchain Solana, sem depender da plataforma desenvolvida, foi realizada a verificação através de uma carteira externa (Phantom) na rede *Devnet* (rede de teste da Solana):

1. Aquisição de jogo através do launcher

2. Importação da mesma chave privada na carteira Phantom
3. Verificação da presença do NFT conforme a Figura 21

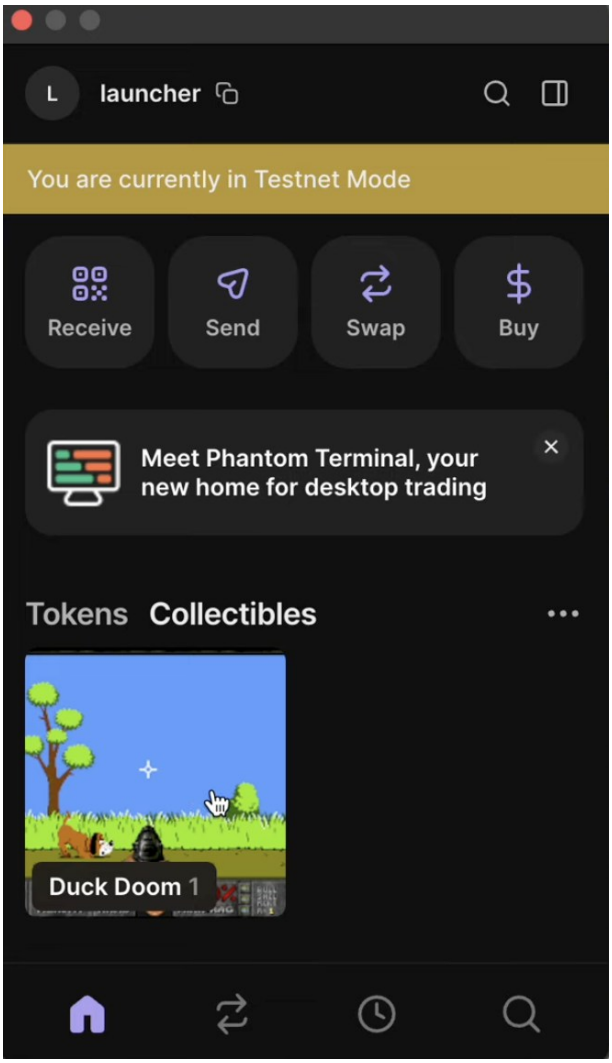

Figura 21 – Carteira Phantom com o NFT criado pela plataforma

Fonte: Phantom (2025)

Para verificar a propriedade real do jogo mesmo fora da plataforma deste TCC, o NFT foi transferido para outra carteira externa através da Phantom (Figura 22).

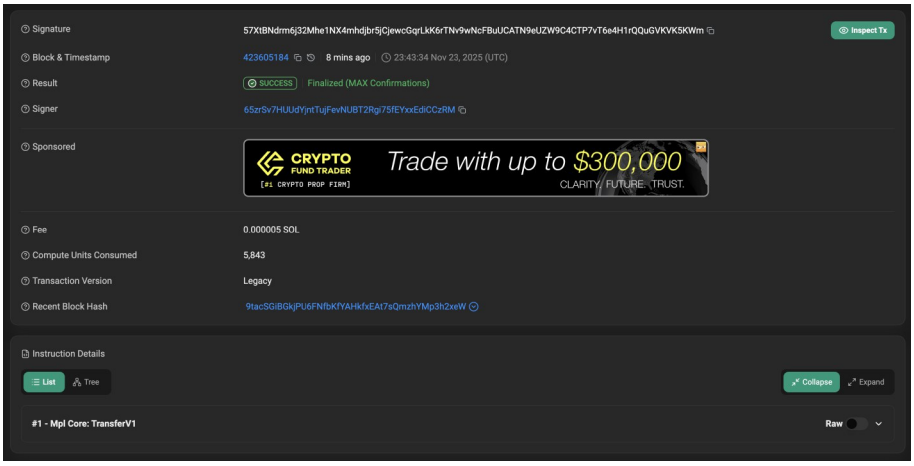

Figura 22 – Transferência do NFT via carteira externa Phantom

Fonte: Solscan (2025)

Logo em seguida, a nova carteira foi importada no sistema desenvolvido, conforme ilustrado pela Figura 23.

Figura 23 – Nova carteira externa importada no sistema

Fonte: Próprio autor

Finalmente, a biblioteca conseguiu sincronizar a propriedade real do jogo pelo novo usuário, além de fornecer as opções de download e execução corretamente (Figura 24).

5.6.6 Viabilidade econômica

Para avaliar a viabilidade econômica da solução, foram mensurados os custos das principais transações executadas na Blockchain Solana. Os valores apresentados referem-se aos custos em SOL durante os testes realizados.

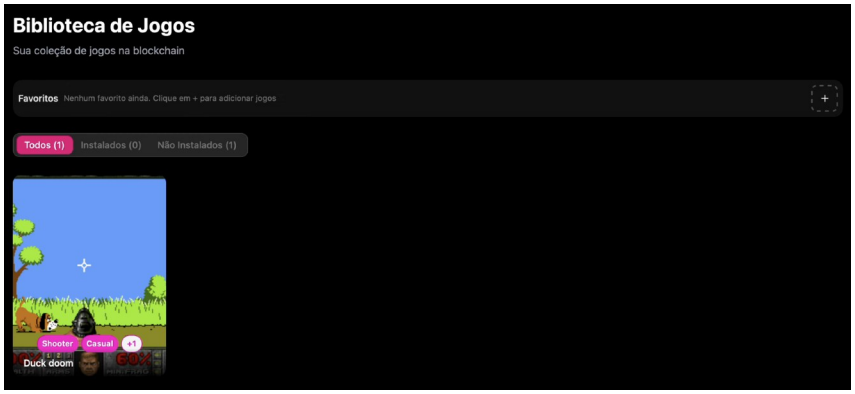

Figura 24 – Visualização do jogo na biblioteca (importado pela carteira Phantom)

Fonte: Próprio autor

5.6.6.1 Publicação de Jogos no Marketplace

A publicação de um novo jogo na plataforma requer a execução de duas transações na Blockchain:

1. **Criação da coleção de NFTs:** representa o jogo como uma coleção de tokens não fungíveis, estabelecendo os metadados base e propriedades compartilhadas.
2. **Criação da Candy Machine:** Configura o mecanismo de distribuição automatizada das licenças (NFTs individuais) do jogo.

A Figura 25 apresenta o custo da transação de criação da coleção, totalizando 0,00183352 SOL (aproximadamente R$ 1,28 na época dos testes).

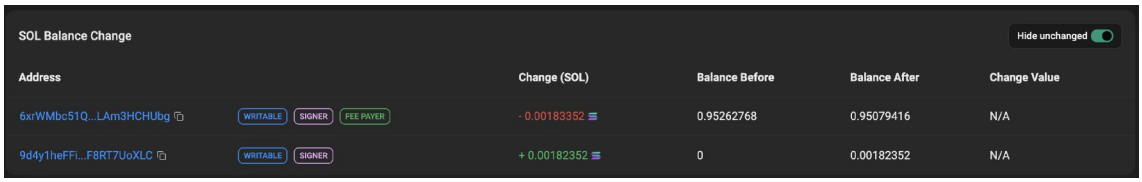

Figura 25 – Custo da transação de criação da coleção NFT no explorador Solscan

Fonte: Solscan (2025)

Já a Figura 26 mostra o custo da criação da *Candy Machine*, com valor de 0,00765208 SOL (aproximadamente R$ 5,30). O custo total para publicação de um jogo no *marketplace* é de 0,0094856 SOL (aproximadamente R$ 6,58).

Portanto, o custo total para cada desenvolvedor de jogo pode ser observado conforme a Tabela 5:

5.6.6.2 Compra de um jogo

Quando um usuário adquire um jogo, é executada uma operação de *mint* que cria um NFT individual vinculado à carteira do comprador. No exemplo a seguir, o

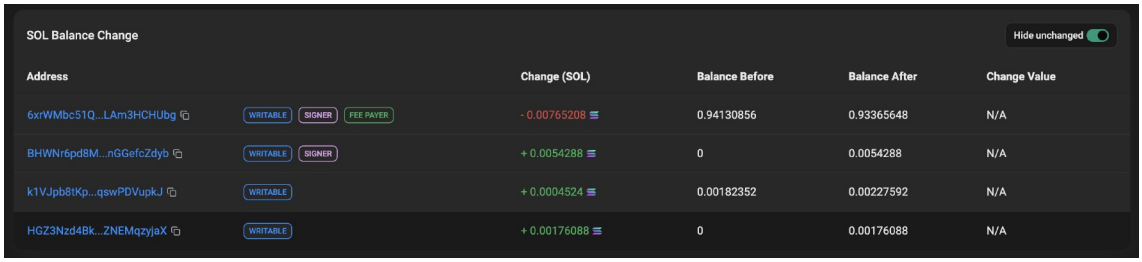

Figura 26 – Custo da transação de criação de *Candy Machine* no explorador Solscan

Fonte: Solscan (2025)

Tabela 5 – Detalhamento dos valores para publicação de um jogo

Tipo da transação	**Valor em SOL**	**Valor aproximado em Real**
Criação da coleção	0,00183352 SOL	R$ 1,28
Criação da *Candy Machine*	0,00765208 SOL	R$ 5,30
Custo total da publicação	**0,0094856 SOL**	**R$ 6,58**

Fonte: Próprio autor

desenvolvedor do jogo definiu o preço que deseja receber por cada cópia do jogo como sendo R$ 303,35. A Figura 27 apresenta o custo dessa transação para o comprador, totalizando 0,39771448 SOL (aproximadamente R$ 306,05). Desse valor, 0,3942 SOL

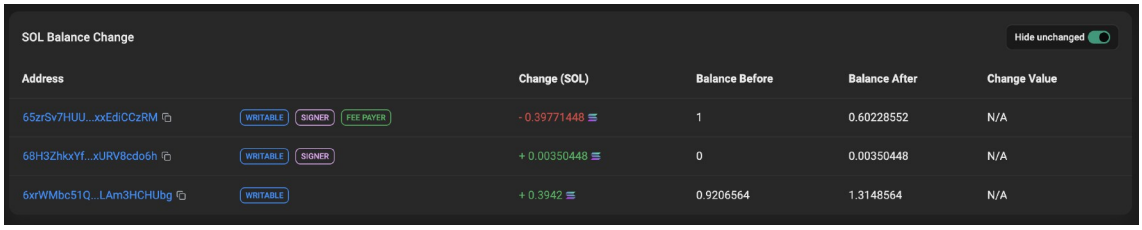

Figura 27 – Custo da transação de *mint* e compra de jogo no explorador Solscan

Fonte: Solscan (2025)

(aproximadamente R$ 303,35) correspondem ao preço do jogo definido pelo desenvolvedor (valor repassado integralmente à respectiva carteira do desenvolvedor do jogo), enquanto 0,00350448 SOL (aproximadamente R$ 2,70) é o aluguel de espaço na rede Solana (*rent*) para armazenamento do NFT e 0,00001 SOL (aproximadamente R$ 0,007) corresponde à taxa de processamento da transação (*transaction fee*). O detalhamento dos custos para a compra de um jogo por um usuário (para o exemplo de um jogo hipoteticamente vendido por R$ 303,35) pode ser observado conforme a Tabela 6:

5.6.6.3 Análise dos Custos

Os valores observados demonstram a viabilidade econômica da solução proposta. É importante ressaltar que os custos em SOL são estáveis e previsíveis, uma vez que o *rent* é calculado de forma determinística com base no tamanho das contas na Blockchain, e as taxas de transação na Solana são fixas por assinatura.

Tabela 6 – Detalhamento dos valores para a compra de um jogo hipotético

Tipo da transação	**Valor em SOL**	**Valor em Real**
Valor cobrado pelo jogo	0,3942 SOL	R$ 303,35
Armazenamento do NFT (licença)	0,00350448 SOL	R$ 2,70
Taxa da transação na Solana	0,00001 SOL	R$ 0,007
Custo total para o comprador	0,39771448 SOL	R$ 306,05

Fonte: Próprio autor

Para desenvolvedores, o custo de publicação de aproximadamente R$ 6,58 representa um investimento acessível, especialmente quando comparado às taxas de entrada de plataformas tradicionais. Para jogadores, o custo adicional de aproximadamente 0,0035 SOL (R$ 2,70) por aquisição é negligenciável em relação aos benefícios de posse real introduzidos pela Blockchain. Foi realizado uma comparação entre o valor de custo para os desenvolvedores entre plataformas similares conforme a Tabela 7, simulando um total de vendas de R$ 30.335,00 de um jogo vendido por R$ 303,35.

Tabela 7 – Comparação o custo das plataformas para os desenvolvedores

Plataforma	**Custo por venda**	**Custo total**
Steam	R$ 91,00 (30%)	R$ 9.100,00
Epic	R$ 36,40 (12%)	R$ 3.640,00
Solução proposta	R$ 0,00	Aproximadamente R$ 6,58

Fonte: Próprio autor

É importante ressaltar que a comparação apresentada considera apenas os custos de transação na *blockchain*, não incluindo taxas de operação da plataforma. O protótipo desenvolvido neste trabalho focou na validação da viabilidade técnica, e não abordou métodos de monetização para a sustentabilidade comercial. Uma implementação em ambiente de produção necessitaria de receita para cobrir custos de infraestrutura (servidor indexador), desenvolvimento e manutenção. Ainda assim, a arquitetura descentralizada proposta reduz significativamente os custos operacionais em comparação às plataformas tradicionais, uma vez que a distribuição dos arquivos é compartilhada entre os próprios usuários via IPFS e o registro de propriedade é mantido pela *blockchain*, permitindo que taxas de plataforma sejam menores que as praticadas por outras propostas centralizadas.

5.6.7 Síntese dos Resultados

Os resultados apresentados neste capítulo demonstram a viabilidade técnica da arquitetura proposta para distribuição e apropriação de jogos eletrônicos. O sistema atende aos objetivos definidos: a representação de jogos como NFTs permite a apropriação real dos ativos; a integração entre Blockchain, IPFS e componentes centralizados opcionais resulta

em uma arquitetura híbrida que equilibra descentralização e usabilidade; e o *launcher* desenvolvido integra todas as funcionalidades em uma única interface.

Os testes realizados comprovaram três características fundamentais do sistema:

- A resiliência da arquitetura, que permanece funcional mesmo sem o indexador centralizado
- O funcionamento da distribuição P2P, na qual cada *launcher* contribui para a propagação dos arquivos na rede
- A apropriação real dos ativos, demonstrada pela visualização e transferência dos NFTs através de carteiras externas, independentes da plataforma.

A análise de viabilidade econômica indicou custos acessíveis tanto para desenvolvedores (aproximadamente R$ 6,58 por publicação de jogo) quanto para usuários (cerca de R$ 2,42 de custo adicional por aquisição de licença de jogo), valores que não representam barreiras significativas à adoção do modelo proposto. Portanto, o sistema desenvolvido valida a hipótese central do trabalho: é tecnicamente viável construir um *marketplace* de jogos que transfira o controle dos ativos digitais das plataformas intermediárias para os usuários finais.

6 Considerações Finais

Neste trabalho foi desenvolvido um *marketplace* descentralizado para jogos digitais composto por contratos inteligentes na Blockchain Solana para gerenciamento de licenças via NFTs, integração com IPFS para distribuição P2P de arquivos, um servidor indexador centralizado opcional e configurável, e uma aplicação *desktop* em Tauri que integra autenticação via carteira, compra, biblioteca e execução de jogos. O sistema implementado engloba o ciclo completo de interação: publicação de jogos por desenvolvedores, autenticação descentralizada de usuários, aquisição de licenças como NFTs, *download* distribuído via IPFS e execução através do *launcher*.

A análise de viabilidade econômica demonstrou custos acessíveis para desenvolvedores (aproximadamente R$ 6,58 por publicação) e usuários (cerca de R$ 2,42 de custo adicional por aquisição), indicando que os valores não representam barreiras significativas para a adoção do modelo proposto. O trabalho valida a hipótese central de que é tecnicamente viável construir um *marketplace* de jogos que transfira o controle dos ativos digitais das plataformas intermediárias para os usuários finais. O sistema desenvolvido demonstra que a apropriação real de licença de jogo pode ser alcançada através da descentralização, sem a necessidade de plataformas intermediárias centralizadas.

6.1 Alcance dos Objetivos

A modelagem da representação de ativos como NFTs foi implementada através de coleções. Cada licença individual é representada por um NFT dentro de uma coleção, que, por sua vez, representa um jogo. A propriedade do NFT confere ao usuário controle total sobre o ativo, permitindo transferências, comercialização ou utilização fora da plataforma. Os testes de interoperabilidade comprovaram que os NFTs são visíveis e transferíveis através de carteiras externas como a Phantom, validando a apropriação real independente da plataforma original.

A integração entre componentes descentralizados e centralizados resultou em uma arquitetura híbrida funcional que equilibra autonomia do usuário com desempenho. A Blockchain gerencia as licenças de forma auditável, o IPFS distribui os arquivos de forma descentralizada com cada *launcher* atuando como nó da rede, e o servidor indexador centralizado fornece otimização opcional para buscas. Os testes de resiliência demonstraram que o sistema permanece funcional mesmo sem o indexador, permitindo buscas diretas por endereço da *Candy Machine* e compras através da Blockchain. A capacidade de alterar o endereço do indexador nas configurações garante autonomia aos usuários, caracterizando os componentes centralizados como substituíveis em vez de pontos únicos de falha.

O módulo de autenticação foi implementado de forma descentralizada, eliminando os cadastros tradicionais e servidores de autenticação centralizados. Os usuários podem gerar novas carteiras ou importar existentes através de sua chave privada, mantendo o controle sobre suas credenciais. A possibilidade de exportar a chave privada garante interoperabilidade com outras aplicações do ecossistema Solana, permitindo a portabilidade total dos seus ativos para fora da plataforma.

O fluxo de publicação de jogos foi desenvolvido para automatizar a criação da coleção de NFTs e da *Candy Machine* correspondente. O processo permite aos desenvolvedores configurar preço, metadados (nome, descrição, imagem) e publicar executáveis para diferentes plataformas através de um formulário com uma pré-visualização em tempo real. A automação elimina intervenção manual para cada venda, permitindo que o processo ocorra de forma independente e sob demanda.

O *marketplace* permite que os usuários façam buscas e comprem jogos, enquanto a biblioteca identifica NFTs na carteira do usuário, exibindo automaticamente os jogos de sua propriedade. A integração entre ambos fornece o fluxo completo desde a descoberta e compra até a visualização dos ativos, com opções de *download* e execução diretamente na interface.

A distribuição via IPFS foi implementada de forma que cada *launcher* (aplicação *desktop*) atue como um nó ativo na rede, contribuindo para a propagação descentralizada dos arquivos. Após o *download* de um jogo, o *pinning* é realizado automaticamente, fazendo com que o *launcher* passe a auxiliar na propagação do conteúdo do jogo para os outros usuários que também queiram baixá-lo após sua compra. Os testes de distribuição P2P comprovaram o funcionamento com apenas dois nós ativos, sem dependência de servidores centralizados. A remoção de jogos instalados exclui o *pinning*, permitindo ao IPFS realizar a limpeza dos arquivos localmente, mas não afeta a propriedade do NFT.

O *launcher* desenvolvido como aplicação *desktop* em Tauri integra todas as funcionalidades do sistema em uma única interface, servindo como ponto de entrada da plataforma. A aplicação permite autenticação via carteira, navegação e compra de jogos no *marketplace*, gerenciamento da biblioteca com filtros por estado de instalação (instalado, não instalado ou todos), *download* através do IPFS, e execução dos jogos adquiridos. A interface proporciona uma experiência de usuário similar às plataformas tradicionais, com o objetivo de facilitar a adoção por usuários com familiaridade nesses sistemas.

6.2 Principais Contribuições

Este trabalho apresenta contribuições relevantes para o campo de distribuição digital e aplicações descentralizadas.

A representação de licenças de jogos digitais como NFTs demonstra que é possível estabelecer propriedade real em vez do modelo tradicional de licenças revogáveis. A abordagem implementada transfere a propriedade para o usuário de forma verificável e independente da plataforma. Esta demonstração indica viabilidade técnica para que outros sistemas de distribuição digital possam adotar padrões similares de apropriação real.

A distribuição P2P para jogos digitais por meio do IPFS, na qual cada instalação da aplicação *desktop* (*launcher*) atua como nó ativo da rede, elimina pontos únicos de falha (servidor centralizado) na distribuição de conteúdo. Contanto que ao menos um *peer* tenha o jogo baixado em dado momento, o que naturalmente poderia ser o próprio desenvolvedor do jogo ao disponibilizá-lo no *marketplace*, o usuário que adquiriu o jogo conseguirá fazer o *download*.

Quanto mais usuários tenham o jogo baixado, pela paralelização no download dos blocos que compõe o jogo no armazenamento do IPFS, maior seria a velocidade de download do jogo conforme sua popularidade. A abordagem desenvolvida pode ser aplicada a outros sistemas que necessitem distribuir grandes volumes de dados de forma descentralizada e resiliente.

6.3 Proteção de Direitos Autorais e Verificação de Propriedade

A arquitetura implementada apresenta uma característica fundamental que deve ser compreendida: os metadados dos NFTs, incluindo os CIDs dos arquivos armazenados no IPFS, são públicos e acessíveis na *blockchain*. Isso significa que qualquer usuário com conhecimento técnico pode:

1. Consultar os metadados de um NFT sem o possuir (público na blockchain)
2. Extrair o CID do IPFS do executável do jogo
3. Baixar o arquivo executável diretamente da rede IPFS através do Kubo ou qualquer outro cliente IPFS

Esta característica não é uma falha de implementação, mas uma consequência do modelo de armazenamento descentralizado baseado em conteúdo. O IPFS, por design, permite que qualquer participante da rede acesse conteúdo desde que conheça seu CID, sem mecanismos nativos de controle de acesso.

6.3.1 Comparação com Modelos Tradicionais

É importante ressaltar que esta característica não torna o sistema proposto mais vulnerável que plataformas tradicionais:

Steam: Segundo Valve Corporation (2025b), a própria Valve reconhece que seu DRM (*Steam Custom Executable Generation*) pode ser facilmente contornado, delegando a implementação de proteções melhores aos desenvolvedores. Jogos no Steam também podem ser copiados após o download caso não implementem DRM adicional.

GOG: A plataforma opera sem DRM, distribuindo executáveis que podem ser copiados livremente. Segundo a GOG (2023), modelos sem DRM são comercialmente viáveis, com usuários frequentemente adquirindo jogos que já possuem em outras plataformas apenas para obter versões sem restrições técnicas.

Solução proposta: Similar às plataformas acima, delegar o DRM aos desenvolvedores, mas com a vantagem de que a verificação de propriedade poderá ser realizada através da *blockchain* (imutável e descentralizada) em vez de servidores da plataforma de *marketplace* que podem ser descontinuados.

6.3.2 Alternativa inicialmente considerada

Uma abordagem possível seria criptografar os executáveis antes do *upload* para o IPFS, com descriptografia realizada pelo *launcher* após validação do NFT. Esta solução impediria downloads não autorizados de usuários que conhecessem o CID do jogo apenas inspecionando a *blockchain*. Contudo, esta abordagem apresenta uma contradição fundamental com a proposta central do trabalho. Se os arquivos são criptografados e apenas o *launcher* possui as chaves de descriptografia, os jogos se tornam dependentes da infraestrutura da plataforma para execução e a descontinuação do *launcher* tornaria os jogos inacessíveis mesmo que o usuário possua o NFT da licença do jogo.

A implementação desta alternativa neste trabalho de conclusão de curso foi rejeitada por contradizer a ideia fundamental do projeto: transferir o controle dos ativos digitais das plataformas intermediárias para os usuários finais.

6.3.3 Proteção da integridade e anti-tampering

A arquitetura implementada já incorpora mecanismos fundamentais de proteção de integridade através do próprio IPFS. O endereçamento baseado em conteúdo garante que qualquer modificação no binário de um jogo resulta em um CID diferente, tornando impossível distribuir versões adulteradas através do CID original. Esta característica do protocolo assegura que, durante a distribuição pela rede P2P, os usuários sempre recebam o binário autêntico publicado pelo desenvolvedor. Contudo, permanecem vulnerabilidades relacionadas a modificações locais realizadas após o *download*, onde usuários mal-intencionados podem alterar os executáveis em seus próprios sistemas para introduzir *cheats*, contornar restrições ou realizar engenharia reversa do código.

Para mitigar modificações pós-*download*, o *launcher* poderia implementar validação

de integridade antes de cada execução, recalculando o *hash* do binário local e comparando com o CID esperado armazenado nos metadados do NFT. Esta verificação detectaria qualquer adulteração local, impedindo a execução de versões modificadas. Além disso, técnicas de proteção em tempo de execução como ofuscação de código, e verificações periódicas de integridade da memória poderiam ser oferecidas através do SDK proposto anteriormente, permitindo aos desenvolvedores fortalecer a proteção contra engenharia reversa e manipulação durante a execução do jogo. Essas proteções, quando combinadas com o DRM baseado em verificação de NFT e as garantias de integridade do IPFS, estabeleceriam um sistema em camadas que protege tanto os direitos de licenciamento quanto a autenticidade do conteúdo distribuído.

6.4 Considerações sobre Escalabilidade

Os testes realizados neste trabalho utilizaram arquivos de tamanho reduzido e um ambiente controlado com dois nós na rede IPFS. Embora os resultados demonstrem a viabilidade técnica da proposta, alguns aspectos devem ser considerados para uma implementação em escala de produção.

Jogos comerciais frequentemente atingem dezenas de gigabytes, e o comportamento do sistema com arquivos dessa magnitude não foi validado. A disponibilidade dos arquivos depende diretamente da quantidade de usuários ativos com o *launcher* em execução. Em cenários com poucos usuários *online*, como no lançamento de novos jogos ou em títulos com baixa popularidade, a velocidade de *download* pode ser prejudicada. Estratégias para mitigar esses cenários, como mecanismos de incentivo e nós dedicados, são discutidas na seção de trabalhos futuros.

6.5 Trabalhos futuros

Embora o sistema desenvolvido demonstre viabilidade técnica da proposta, algumas limitações e oportunidades de melhoria foram identificadas durante o desenvolvimento e testes.

A implementação de mecanismos de proteção de direitos autorais (DRM) representa uma extensão fundamental para a plataforma. Atualmente, o sistema não impede que usuários copiem os arquivos dos jogos após o download, delegando essa responsabilidade aos desenvolvedores. A validação de propriedade em tempo de execução, através de verificações periódicas da presença do NFT na carteira do usuário durante a execução do jogo, fortaleceria a proteção dos direitos autorais. Esta funcionalidade poderia ser oferecida através de um SDK (*Software Development Kit*) que permitisse aos desenvolvedores integrarem verificação de licença diretamente em seus jogos, garantindo que apenas

proprietários legítimos tenham acesso ao conteúdo.

A otimização de armazenamento através da deduplicação de arquivos constitui outra melhoria significativa. Na implementação atual, os arquivos são armazenados tanto no repositório IPFS quanto duplicados no sistema de arquivos do usuário para execução dos jogos. A eliminação dessa duplicação, permitindo que os jogos sejam executados diretamente do repositório IPFS através de sistemas de links simbólicos ou montagem virtual (e.g. FUSE ou *Filesystem in Userspace*), reduziria significativamente o espaço em disco necessário e melhoraria a eficiência do sistema.

A descentralização completa do catálogo de jogos representa a evolução da arquitetura híbrida implementada. Atualmente, o servidor indexador centralizado facilita a descoberta e busca de jogos, mas introduz dependência de infraestrutura centralizada HTTP. A substituição deste componente por uma lista de jogos publicados (catálogo do *marketplace*) armazenada também no IPFS, combinada com um indexador local no *launcher* (aplicação *desktop*) utilizando SQLite para consultas e filtros, eliminaria esta dependência. A sincronização incremental, baixando apenas os blocos faltantes quando novos jogos são adicionados ao catálogo, manteria a eficiência do sistema enquanto alcança descentralização completa para todas as operações críticas da plataforma.

O gerenciamento de atualizações de jogos é necessário para a manutenção dos conteúdos distribuídos. Atualmente, o sistema não oferece um mecanismo para os desenvolvedores publicarem novas versões de seus jogos. A implementação de versionamento através do protocolo IPNS (*InterPlanetary Name System*) permitiria que desenvolvedores atualizassem seus jogos sem a necessidade de republicação completa. O IPNS possibilita a criação de endereços mutáveis que apontam para os *hashes* IPFS mais recentes dos arquivos, eliminando a necessidade de atualizar os metadados na Blockchain a cada nova versão. Resumidamente, o IPNS é um recurso opcional do IPFS que funciona como dicionário de {chave_fixa, CID_mutável} que o *peer* que criou o CID da primeira versão do arquivo (no caso, o jogo) pode atualizar sempre que necessário.

O *launcher* poderia detectar versões desatualizadas resolvendo periodicamente o nome IPNS do jogo e, comparando com a versão local baixada, fazer o download apenas dos (blocos de) arquivos modificados. Essa abordagem manteria os jogos atualizados eficientemente e com distribuição descentralizada das atualizações.

A implementação de um mecanismo de empréstimo de jogos representa outra possibilidade pelo modelo descentralizado proposto. Através de um *smart contract*, seria possível criar um sistema no qual o proprietário de um NFT pudesse emprestá-lo temporariamente a outra carteira, com garantia automática de devolução. O contrato inteligente travaria o NFT por um tempo, transferindo temporariamente os direitos de acesso, e ao final do prazo, o ativo retornaria automaticamente para a carteira original sem necessidade de intervenção manual.

A criação de mecanismos de incentivo e estratégias para garantir disponibilidade dos arquivos representa uma oportunidade de fortalecimento do modelo P2P. Usuários que mantêm seus *launchers* ativos por longos períodos e armazenam diversos jogos baixados contribuem para a disponibilidade e velocidade de distribuição na rede IPFS. Um sistema de recompensas poderia incentivar esse comportamento, através de *tokens*, descontos em compras futuras ou benefícios na plataforma. Além disso, os próprios desenvolvedores poderiam executar múltiplos nós durante o lançamento de seus jogos, garantindo disponibilidade inicial enquanto poucos nós propagam seus jogos. A própria plataforma também poderia manter nós geograficamente distribuídos como parte de sua infraestrutura, financiados por taxas sobre as vendas, conforme discutido na análise de viabilidade econômica. Essa abordagem combinada, alinharia os interesses de todos os participantes com a saúde da rede, promovendo maior resiliência e desempenho na distribuição descentralizada.

Referências

ALT, R. Electronic markets on blockchain markets. *Electronic Markets*, Springer Science and Business Media LLC, v. 30, n. 2, p. 181–188, jun. 2020. ISSN 1422-8890. Disponível em: <http://dx.doi.org/10.1007/s12525-020-00428-1>. Citado na página 16.

BENET, J. Ipfs - content addressed, versioned, p2p file system. *arXiv*, arXiv, 2014. Disponível em: <https://arxiv.org/abs/1407.3561>. Citado na página 23.

Blizzard Entertainment. *Contrato de Licença de Usuário Final da Blizzard*. 2023. Disponível em: <https://www.blizzard.com/pt-br/legal/48fcaa57-2657-43b3-9c4b-b6d986a991b9/contrato-de-licen%C3%A7a-de-usu%C3%A1rio-final-da-blizzard>. Citado na página 25.

CHALK, A. *Dark and Darker will be purged from Epic Games Store libraries later this year, but it's still available on Steam for now*. 2025. Disponível em: <https://www.pcgamer.com/games/dark-and-darker-will-be-purged-from-epic-games-store-libraries-later-this-year-but-its-still-available-o>. Citado na página 24.

Core Candy Machine. *Candy Guards*. 2025. Disponível em: <https://developers.metaplex.com/core-candy-machine/guards>. Citado 2 vezes nas páginas 22 e 38.

COULOURIS, G. F. et al. *Distributed Systems*. 5. ed. Upper Saddle River, NJ: Pearson, 2011. ISBN 0-13-214301-1. Citado na página 19.

DOAN, T. V. et al. Towards decentralised cloud storage with ipfs: Opportunities, challenges, and future directions. *arXiv*, arXiv, 2022. Disponível em: <https://arxiv.org/abs/2202.06315>. Citado na página 23.

Elixir Games. *Terms of Use for Elixir Game Launcher*. 2024. Disponível em: <https://about.elixir.games/terms>. Citado na página 26.

Elixir Games. *Elixir Launcher*. 2025. Disponível em: <https://launcher.elixir.games>. Citado na página 26.

FERDINAND, P. K. *Ubisoft Reportedly Removing Access to The Crew From Buyer's Accounts*. 2024. Disponível em: <https://gamerant.com/the-crew-ubisoft-connect-revoke-access/>. Citado na página 24.

FLANAGAN, D. *JavaScript: O Guia Definitivo*. [S.l.]: Bookman Editora, 2012. ISBN 9788565837484. Citado na página 27.

Fractal. *Fractal: The Marketplace for Web3 Games*. 2025. Disponível em: <https://fractal.is>. Citado na página 25.

Fractal. *Uploading binaries to Fractal launcher*. 2025. Disponível em: <https://docs.fractal.is/launcher/upload-your-game>. Citado na página 25.

GOG. *Benefits of Releasing Your Game on GOG – A Guide for Indie Game Developers*. 2023. Disponível em: <https://www.gog.com/blog/benefits-of-releasing-your-game-on-gog-a-guide-for-indie-game-developers/>. Citado na página 62.

GOLDBERG, J. *Learning typescript*. Sebastopol, CA: O'Reilly Media, 2022. ISBN 978-1098110338. Citado na página 27.

KLABNIK, S.; NICHOLS, C. *The rust programming language*. San Francisco, CA: No Starch Press, 2018. ISBN 978-1-59327-828-1. Citado na página 28.

LEDUC, G.; KUBLER, S.; GEORGES, J.-P. Innovative blockchain-based farming marketplace and smart contract performance evaluation. *Journal of Cleaner Production*, Elsevier BV, v. 306, p. 127055, jul. 2021. ISSN 0959-6526. Disponível em: <http://dx.doi.org/10.1016/j.jclepro.2021.127055>. Citado na página 16.

LIMA, R. V. A. de. *Análise comparativa de gerenciadores de pacotes javascript: npm, pnpm e yarn*. Dissertação (Monografia de Graduação em Ciência da Computação) — Campus Quixadá da Universidade Federal do Ceará, 2025. Disponível em: <https://repositorio.ufc.br/bitstream/riufc/83080/1/2025_tcc_rvalima.pdf>. Citado na página 28.

MAGALHãES, R. A.; FANTINI, L. M. C. Blockchain e os ativos digitais em jogos. *Revista Jurídica da FA7*, Educadora Sete de Setembro, v. 19, n. 2, p. 29–44, dez. 2022. ISSN 1809-5836. Disponível em: <http://dx.doi.org/10.24067/rjfa7;19.2:1323>. Citado 2 vezes nas páginas 18 e 20.

MASTORAKIS, N. E. *Tokenization of a Property on the Ethereum and Solana Blockchains: A Comparative Study*. 2025. 45–57 p. Disponível em: <https://www.iaras.org/iaras/filedownloads/ijems/2025/001-ijems-2025-a1.pdf>. Citado na página 29.

Mozilla Developer. *CSS: Cascading Style Sheets*. 2025. Disponível em: <https://developer.mozilla.org/en-US/docs/Web/CSS#cookbook>. Citado na página 27.

Phantom. 2025. Disponível em: <https://phantom.com/>. Citado na página 53.

POKHREL, S. et al. Decentralized File Storage System for Web3 with IPFS. In: INSTITUTE OF ENGINEERING, TRIBHUVAN UNIVERSITY, NEPAL. *Proceedings of 14th IOE Graduate Conference*. [S.l.], 2023. v. 14, p. 174 – 179. ISSN 2350-8914 (Online), 2350-8906 (Print). Citado na página 29.

POP, C. et al. Blockchain based decentralized applications: Technology review and development guidelines. *arXiv*, arXiv, 2020. Disponível em: <https://arxiv.org/abs/2003.07131>. Citado 2 vezes nas páginas 18 e 20.

React. Disponível em: <https://react.dev/>. Citado na página 28.

REKHI, G. *Peer-to-Peer File Sharing*. 2025. Disponível em: <https://udayton.teamdynamix.com/TDClient/1868/Portal/KB/ArticleDet?ID=46760>. Citado na página 25.

ROBBINS, J. N. *Learning Web Design 5e*. Sebastopol, CA: O'Reilly Media, 2018. ISBN 978-1-491-96020-2. Citado na página 27.

RUSTAMOV, S.; BEKHZOD, N. Comparison – centralized, decentralized and distributed systems. *International Journal of Academic Engineering Research (IJAER)*, v. 8, p. 1–5, 6 2024. ISSN 2643-9085. Disponível em: <http://ijeais.org/wp-content/uploads/2024/6/IJAER240601.pdf>. Citado na página 19.

Solana Foundation. *Contas*. 2025. Disponível em: <https://solana.com/pt/docs/core/accounts>. Citado 2 vezes nas páginas 20 e 21.

Solana Foundation. *Programas*. 2025. Disponível em: <https://solana.com/pt/docs/core/programs>. Citado 2 vezes nas páginas 21 e 22.

Solscan. 2025. Disponível em: <https://solscan.io/>. Citado 3 vezes nas páginas 54, 55 e 56.

SUBRAMANIAN, H. Decentralized blockchain-based electronic marketplaces. *Communications of the ACM*, Association for Computing Machinery (ACM), v. 61, n. 1, p. 78–84, dez. 2017. ISSN 1557-7317. Disponível em: <http://dx.doi.org/10.1145/3158333>. Citado 2 vezes nas páginas 16 e 20.

SUBRAMANIAN, H. A decentralized marketplace for patient-generated health data: Design science approach. *Journal of Medical Internet Research*, JMIR Publications Inc., v. 25, p. e42743, fev. 2023. ISSN 1438-8871. Disponível em: <http://dx.doi.org/10.2196/42743>. Citado na página 16.

SUSNJARA, S.; SMALLEY, I. *What is blockchain?* 2021. Disponível em: <https://www.ibm.com/think/topics/blockchain>. Citado na página 20.

TailwindCSS. 2025. Disponível em: <https://tailwindcss.com/>. Citado na página 27.

Tauri. *Tauri Documentation*. 2025. Disponível em: <https://tauri.app/start/>. Citado na página 28.

Valve Corporation. *ACORDO DE ASSINATURA DO STEAM*. 2025. Disponível em: <https://store.steampowered.com/subscriber_agreement/brazilian>. Citado na página 24.

Valve Corporation. *Steam DRM*. 2025. Disponível em: <https://partner.steamgames.com/doc/features/drm>. Citado 2 vezes nas páginas 25 e 62.

WANG, Q. et al. Non-fungible token (nft): Overview, evaluation, opportunities and challenges. *arXiv*, arXiv, 2021. Disponível em: <https://arxiv.org/abs/2105.07447>. Citado na página 22.

WILDE, T. *Most game devs don't think Steam earns its 30% revenue cut*. 2021. Disponível em: <https://www.pcgamer.com/most-game-devs-dont-think-steam-earns-its-30-revenue-cut>. Citado na página 24.

Apêndices

A Lista de variáveis

A tabela a seguir define as variáveis a serem utilizadas nos comandos para baixar os programas do metaplex da blockchain Solana.

Tabela 8 – Variáveis utilizadas nos comandos de configuração

Variável	**Endereço na Blockchain**
CORE_ADDR	CoREENxT6tW1HoK8ypY1SxRMZTcVPm7R94rH4PZNhX7d
METADATA_ADDR	metaqbxxUerdq28cj1RbAWkYQm3ybzjb6a8bt518x1s
BUBBLEGUM_ADDR	BGUMAp9Gq7iTEuizy4pqaxsTyUCBK68MDfK752saRPUY
CANDY_MACHINE_ADDR	CMACYFENjoBMHzapRXyo1JZkVS6EtaDDzkjMrmQLvr4J
CANDY_GUARD_ADDR	CMAGAKJ67e9hRZgfC5SFTbZH8MgEmtqazKXjmkaJjWTJ

Fonte: Próprio autor

A tabela a seguir define as variáveis para o diretório no qual os programas serão armazenados após serem baixados da rede Solana.

Tabela 9 – Variáveis de diretório

Variável	**Diretório**
CORE_DIR	./test-programs/mpl-core.so
METADATA_DIR	./test-programs/mpl-token-metadata.so
BUBBLEGUM_DIR	./test-programs/mpl-bubblegum.so
CANDY_MACHINE_DIR	./test-programs/mpl-core-candy-machine.so
CANDY_GUARD_DIR	./test-programs/mpl-core-candy-guard.so

Fonte: Próprio autor

B Comandos para dump dos programas Metaplex utilizados

É possível baixar o binário dos programas a serem executados por meio do comando com a seguinte estrutura:

```
solana program dump -u <URL_OR_CLUSTER> <ADDRESS> <DIRECTORY>
```

URL ou cluster representam o cluster que será utilizado para fazer o download do programa, a letra "m" indica que será utilizado o cluster mainnet-beta da solana. Logo após, é necessário prover o endereço do contrato a ser baixado e o diretório onde será armazenado.

Para realizar o download dos programas utilizados no projeto é necessário utilizar o comando *bash*:

```
solana program dump -u m $METADATA_ADDR $METADATA_DIR

solana program dump -u m $BUBBLEGUM_ADDR $BUBBLEGUM_DIR

solana program dump -u m $CORE_ADDR  $CORE_DIR

solana program dump -u m $CANDY_MACHINE_ADDR $CANDY_MACHINE_DIR

solana program dump -u m $CANDY_GUARD_ADDR $CANDY_GUARD_DIR
```

C Comando de inicialização da test ledger

O comando a seguir pode ser utilizado para iniciar uma *ledger* de testes. Com ela é possível interagir com a blockchain localmente, enviar transações, interagir com programas e requisitar *airdrop*.

```
solana-test-validator \
--bpf-program $METADATA_ADDR $METADATA_DIR \
--bpf-program $BUBBLEGUM_ADDR $BUBBLEGUM_DIR \
--bpf-program $CORE_ADDR $CORE_DIR \
--bpf-program $CANDY_MACHINE_ADDR $CANDY_MACHINE_ADDR \
--bpf-program $CANDY_GUARD_ADDR $CANDY_GUARD_DIR
```

Após a execução do comando a *ledger* estará pronta para uso e pode ser acessada via RPC url: `http://127.0.0.1:8899`.

D Apêndice C – Vídeos Demonstrativos

Este apêndice contém links para vídeos demonstrativos do sistema implementado, disponíveis no YouTube. Os vídeos ilustram o funcionamento prático das funcionalidades descritas no Capítulo 6 (Resultados).

Tabela 10 – Vídeos demonstrativos do sistema

Funcionalidade	**Link**
Autenticação	<https://www.youtube.com/watch?v=lAGeIdI8U0s>
Marketplace	<https://www.youtube.com/watch?v=kiUNwAGvb7c>
Biblioteca	<https://www.youtube.com/watch?v=RqEGO290kGw>
Launcher	<https://www.youtube.com/watch?v=vbsyNnmdkWM>
Interoperabilidade	<https://www.youtube.com/watch?v=JzlyMIHo2a8>

Fonte: Próprio autor