Otávio Vacari Martins

MiniMoni: sistema de micropagamentos para streaming de vídeo

São Paulo, SP

2024

Otávio Vacari Martins

MiniMoni: sistema de micropagamentos para streaming de vídeo

Trabalho de conclusão de curso apresentado ao Departamento de Engenharia de Computação e Sistemas Digitais da Escola Politécnica da Universidade de São Paulo para obtenção do Título de Engenheiro.

Universidade de São Paulo – USP

Escola Politécnica

Departamento de Engenharia de Computação e Sistemas Digitais (PCS)

Orientador: Prof. Dr. Marcos Antonio Simplici Junior

São Paulo, SP

2024

Catalogação-na-publicação

Martins, Otávio
MiniMoni: sistema de micro pagamentos para streaming de vídeo / O. Martins -- São Paulo, 2024.
50 p.

Trabalho de Formatura - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Computação e Sistemas Digitais.

1.Desenvolvimento Web 2.Streaming de Vídeo 3.Pagamentos Digitais I.Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Computação e Sistemas Digitais II.t.

Dedicamos este trabalho aos que compreendem o erro não como falha, mas como essencial ao processo de aprendizagem.

Agradecimentos

Expressamos nossa profunda gratidão ao Prof. Dr Marcos Antonio Simplici Junior pela orientação e apoio constante ao longo deste projeto. Sua experiência não apenas iluminou nosso caminho, mas também nos incentivou a superar nossos limites.

Resumo

No contexto atual, a digitalização acelerada dos mercados globais e a expansão do e-commerce evidenciam a importância crítica dos sistemas de pagamento online. Porém, esse modelo causa frustração do lado do usuário, já que é necessário assinar diversos serviços simultaneamente. Um modelo que permitisse a cobrança de pagamentos baseada pelo de uso poderia resolver esse problema, uma vez que essa abordagem centrada no usuário levaria a um maior controle e transparência dos gastos. Entretanto, os métodos convencionais de pagamento enfrentam problemas significativos, incluindo alta carga cognitiva para os usuários, latência nas transações, limitações de escalabilidade para multivendedores e a inabilidade de realizar pagamentos internacionais de maneira eficiente. Diante desses desafios, este trabalho propõe uma solução que utiliza o esquema de micropagamentos Payword junto da programabilidade da Blockchain para a construção de um sistema de vídeo sob demanda. Dessa forma, a cobrança é realizada por cada segmento de vídeo assistido e, ainda assim, é possível realizar esses pagamentos, contornando as limitações anteriores.

Palavras-chave: tecnologia financeira, inovação em métodos de pagamento, escalabilidade de sistemas de pagamento

Abstract

In the current context, the accelerated digitalization of global markets and the expansion of e-commerce highlight the critical importance of online payment systems. However, this model causes user frustration, as it requires simultaneous subscription to various services. A model that allowed usage-based payment charging could solve this problem, since this user-centered approach would lead to greater control and transparency of expenses. However, conventional payment methods face significant problems, including high cognitive load for users, transaction latency, scalability limitations for multi-vendors, and the inability to conduct international payments efficiently. Given these challenges, this work proposes a solution that uses the Payword micropayment scheme along with Blockchain programmability to build a video-on-demand system. Thus, charging is performed for each video segment watched, and it is still possible to make these payments while circumventing the previous limitations.

Keywords: Blockchain, Micropayments, Video Streaming, Pay-per-use, Digital Payments

Lista de ilustrações

Lista de abreviaturas e siglas

API	Application Programming Interface
CDN	Content Delivery Network
CI/CD	Continuous Integration/Continuous Deployment
CSR	Client-Side Rendering
dApp	Decentralized Application
HLS	HTTP Live Streaming
IoT	Internet of Things
ISR	Incremental Static Regeneration
M2M	Machine-to-Machine
ORM	Object-Relational Mapping
SaaS	Software-as-a-Service
SSG	Static Site Generation
SSR	Server-Side Rendering

Sumário

1 Introdução

A internet modificou imensamente a maneira com qual a sociedade consome produtos. Atualmente, a Meta Platforms, anteriormente conhecida como Facebook, passou a incluir uma ampla gama de produtos e serviços, incluindo Facebook, Instagram, Messenger, WhatsApp. Essas soluções baseiam-se no modelo Software as a Service (SaaS), onde empresas conseguem fornecer serviços on-line de maneira escalável para milhões de usuários.

Essas empresas fornecem o serviço gratuitamente, porém com a exibição de anúncios, permitindo enormes lucros. Em 2022, o tamanho global do mercado de gastos com publicidade na Internet foi estimado em cerca de US$ 289,9 bilhões e espera-se que atinja aproximadamente US$ 892,2 bilhões até 2031 (Mordor Intelligence, 2023). Esse crescimento está sendo impulsionado por fatores como o uso generalizado de smartphones, o desenvolvimento da Internet de alta velocidade, a popularidade das plataformas de streaming e um aumento nos gastos com publicidade digital em vários setores.

Além disso, também existem aquelas empresas que se baseiam no modelo de cobrança mensal. Um relatório de 2021 mostrou que 60% dos americanos estão inscritos em pelo menos um serviço de streaming de música pago, com o número médio de assinaturas por pessoa sendo duas (CLOUDWARDS, 2021), demonstrando o sucesso do modelo de negócio. Esse sucesso também está presente nos serviços de streaming de vídeo, constatando que a média de assinaturas de serviços de streaming de vídeo sob demanda (SVOD) por domicílio americano é de quatro (KARRER, 2024).

Também, novas maneiras de pagamentos que possibilitam a interação ao vivo têm se popularizado com a popularização das plataformas de transmissão. A indústria do streaming de conteúdo ao vivo, em 2023, evidenciou um expressivo crescimento, com 7,6 bilhões de horas de conteúdo ao vivo visualizadas. O segmento de live commerce está projetado para atingir $31,7 bilhões ao fim de 2023, demonstrando um acentuado aumento em comparação a 2021. Este crescimento sinaliza a preferência crescente dos consumidores por interações ao vivo e conteúdo dinâmico em detrimento dos métodos tradicionais de consumo de mídia e compras online (DemandSage, 2023).

Outras tendências de pagamentos mais recentes se aliam a tecnologia mais recentes como a internet das coisas (IoT) (STRUGAR et al., 2018). Os pagamentos do tipo machine-to-machine (M2M) permitem que dispositivos conectados à internet realizem transações financeiras entre si sem intervenção humana. Esta tecnologia tem potencial em vários setores, incluindo automotivo, manufatura, e serviços públicos, onde pode simplificar e automatizar pagamentos por serviços consumidos ou oferecidos por dispositivos inteligentes.

1.1 Motivação

No cenário atual de comércio eletrônico e transações financeiras digitais, a eficiência e a segurança dos métodos de pagamento são de suma importância tanto para consumidores quanto para empresas. Entretanto, apesar dos avanços tecnológicos, muitas das soluções de pagamento existentes ainda apresentam desafios significativos que limitam sua eficácia e aceitação em larga escala.

1.1.1 Problemas com Pagamentos por Cartão de Crédito

Os pagamentos por cartão de crédito, embora amplamente utilizados, enfrentam questões críticas de latência, custos de intermediação e limitações de escalabilidade. A latência nos processos de autorização e liquidação podem comprometer a experiência do usuário, especialmente em ambientes de comércio eletrônico, onde a agilidade é frequentemente um fator crítico. Além disso, as taxas impostas por terceiros, como bancos e operadoras de cartão, aumentam os custos para os comerciantes e, muitas vezes, são repassadas aos consumidores, afetando a competitividade do mercado. Por fim, a escalabilidade dos sistemas de pagamento por cartão é desafiada pelo crescente volume de transações globais, exigindo infraestruturas robustas e custosas para manter a eficiência operacional.

1.1.2 Desafios do Pix e Outros Pagamentos Instantâneos

Por outro lado, soluções de pagamento instantâneo, como o Pix no Brasil, embora representem um avanço significativo na redução de custos e tempo de transação, ainda enfrentam barreiras de adoção devido à carga intelectual exigida aos usuários. A necessidade de escanear um QR Code ou acessar um aplicativo bancário para concluir transações introduz etapas adicionais que podem desencorajar o uso, especialmente em contextos de compra rápida ou quando a simplicidade é essencial. Além disso, a dependência de dispositivos móveis e conectividade à internet pode ser um limitador em regiões com infraestrutura digital precária.

1.1.3 Limitações nos Pagamentos Internacionais

Os pagamentos internacionais representam outra área de preocupação. A incompatibilidade entre diferentes sistemas de pagamento e as regulamentações variadas entre países tornam as transações internacionais complexas, demoradas e custosas. Essa dificuldade ocorre devido diferente infraestrutura de pagamentos utilizada por cada país, como sistemas bancários, diferentes moedas correntes e limitações legais para a operação. A incapacidade de realizar pagamentos internacionais com facilidade limita significativamente o potencial de crescimento das empresas que buscam operar em uma escala global.

1.1.4 Necessidade de Pagamentos M2M

À medida que avançamos para uma era de maior conectividade, impulsionada pela Internet das Coisas (IoT), surge a necessidade de facilitar pagamentos M2M. Essa tecnologia tem o potencial de revolucionar setores inteiros, permitindo que dispositivos conectados realizem transações financeiras autonomamente. No entanto, a falta de sistemas de pagamento capazes de suportar essas transações sem intervenção humana e de forma segura é um obstáculo significativo. A capacidade de programar pagamentos M2M abre novos horizontes para a automação e a eficiência operacional em diversas indústrias.

1.2 Objetivos

O presente trabalho propõe o desenvolvimento de um sistema de pagamentos online inovador, cujo desenho visa abordar as principais limitações identificadas nos métodos de pagamento existentes. O foco está em quatro objetivos principais que, em conjunto, pretendem remodelar a experiência de pagamento tanto para consumidores quanto para comerciantes, facilitando transações mais rápidas, seguras e acessíveis em um ambiente globalizado. Estes objetivos são detalhados a seguir:

- **Redução da Carga Cognitiva**: O sistema é projetado para minimizar a carga cognitiva dos usuários durante o processo de pagamento. Isso implica em uma interface intuitiva e um processo de transação simplificado, eliminando a necessidade de etapas complexas como a leitura de QR codes ou a navegação por múltiplos aplicativos e plataformas. A intenção é proporcionar uma experiência de pagamento fluida, que possa ser completada com um mínimo de esforço cognitivo por parte do usuário.

- **Baixa Latência nas Transações**: Priorizando a eficiência, o sistema visa reduzir significativamente a latência em transações de pagamento. Ao otimizar os processos de processamento e verificação de pagamentos, busca-se garantir que as transações sejam concluídas quase em tempo real. Isso não apenas melhora a experiência do usuário, mas também beneficia os comerciantes ao acelerar o fluxo de caixa e reduzir o tempo de espera para a confirmação de pagamentos.

- **Solução Multi-vendedor** O sistema é projetado para ser altamente escalável e flexível, suportando uma ampla gama de comerciantes e plataformas de venda. Isso inclui a capacidade de integrar-se facilmente com diferentes websites de e-commerce, sistemas de ponto de venda e plataformas de marketplace, proporcionando uma solução de pagamento versátil que pode ser adotada em diversos contextos comerciais. A meta é oferecer uma solução única que atenda às necessidades de pagamento de uma variedade de vendedores, desde pequenos negócios até grandes corporações.

- **Habilitação de Pagamentos Internacionais** Reconhecendo a importância do comércio global, o sistema será capaz de facilitar pagamentos internacionais. Isso envolve superar os desafios associados às diferenças entre moedas, taxas de câmbio e regulamentações bancárias internacionais. O objetivo é proporcionar uma maneira segura, confiável e econômica para que consumidores e comerciantes realizem transações além das fronteiras nacionais, sem as taxas proibitivas e a complexidade que muitas vezes caracterizam os pagamentos internacionais atualmente.

1.3 Metodo

No desenvolvimento do trabalho, foi realizada uma busca extensa na literatura para identificar soluções existentes que buscam resolver o problema de micro-pagamentos. Elsheikh, Clark e Youssef (2020a) discute a implementação do sistema de pagamento Payword na plataforma Ethereum, focando na integração das tecnologias para melhorar as transações financeiras; Néto (2002) aborda uma implementação prática do PayWord, que inclui a construção de um protótipo para testar a viabilidade do protocolo em um ambiente controlado; Rivest e Shamir (1996) apresenta duas propostas de sistemas de micro-pagamentos destinados a facilitar transações de baixo valor na internet. O Payword e o Micromint: um esquema que utiliza a geração de moedas virtuais através de um processo criptográfico. A proposta do projeto surgiu após estudos e análises dessas diversas soluções existentes apresentadas nos artigos.

Durante esta fase de pesquisa, investigamos uma variedade de métodos e tecnologias empregadas nesse setor, avaliando características, vantagens e limitações possíveis. Dentre as vantagens investigadas analisamos a maior eficiência e menor custo como nas plataformas Lightning Network e Ripple XRP que oferecem transações rápidas e com custos significativamente menores comparados aos métodos tradicionais. Além disso, vimos uma simplificação do processo de pagamento, como visto no uso de sistemas *pay-as-you-go*, um modelo de pagamento onde os usuários pagam por um serviço ou produto conforme o utilizam, ao invés de pagamentos periódicos fixos, melhora a experiência do usuário. Esse modelo facilita a cobrança, pois permite que os consumidores tenham maior controle sobre seus gastos, pagando apenas pelo que realmente consomem.

Contudo, apesar das melhorias, alguns dos sistemas discutidos ainda enfrentam desafios de escalabilidade quando aplicados a um grande número de transações simultâneas. Sistemas, como PIX, requerem interações complexas ou múltiplas etapas de autenticação podem também serem limitadores pois os mesmos aumentam a carga cognitiva dos usuários, o que pode desencorajar a adoção em cenários onde a velocidade é crítica.

Assim, identificamos que o PayWord é uma solução é uma solução que também resolve as limitações de velocidade das transações e escalabilidade pois a maior parte

do processamento é feita sem a necessidade da interação com o agente intermediário de pagamentos. A comunicação dos micro-pagamentos fica restrita aos vendedor e ao consumidor, enquanto a interação com o terceiro é realizada apenas na validação inicial e ao final, quando o vendedor realiza o saque.

2 Fundamentação teórica

Este capítulo apresenta os conceitos e tecnologias para desenvolvimento de aplicações descentralizadas. A Seção 2.1 descreve a evolução dos pagamentos digitais, desde o E-cash até as blockchains modernas como Bitcoin e Ethereum, e introduz o protocolo Payword para micropagamentos. A Seção 2.5 implementa as bibliotecas Viem e Wagmi para desenvolvimento em blockchain. A Seção 2.3 aborda o React e Next.js como frameworks para interfaces web. A Seção 2.4 define os protocolos de streaming HLS e a biblioteca HLS.js. A Seção 2.6 expõe as ferramentas de design Tailwind CSS.

2.1 Tecnologias para pagamento digitais

O final do século XX ficou marcado pela inauguração da criptografia. Com isso, diversas tecnologias surgiram, como assinatura digitais e funções de *hash*, possibilitando um aumento na privacidade e segurança no mundo digital. Também, surgiram algumas tentativas de implementar métodos de pagamento digitais usando essas inovações.

2.1.1 E-cash

Por exemplo, o e-cash (CHAUM; FIAT; NAOR, 1988) representava uma forma pioneira de dinheiro digital, concebido para proporcionar anonimato e segurança nas transações online. Surgiu nos anos 90, criado pelo cientista da computação David Chaum, como uma resposta direta à crescente necessidade de privacidade nas transações financeiras digitais. Seu objetivo era permitir pagamentos anônimos, preservando a segurança e a privacidade dos usuários, um avanço considerável na época.

Porém, o e-cash enfrentou desafios significativos que contribuíram para sua eventual falha. Entre esses desafios, destacam-se a dificuldade de implementação em larga escala e a limitação de sua infraestrutura em lidar com a questão do *double spending* (gasto duplo), que é a possibilidade de uma mesma unidade monetária digital ser gasta mais de uma vez. Para resolver esse problema, é necessário utilizar uma autoridade verificadora.

2.1.2 Bitcoin

Mais recentemente, o Bitcoin (NAKAMOTO, 2008), apresentado em 2008 por uma pessoa ou grupo sob o pseudônimo Satoshi Nakamoto, emergiu como a primeira criptomoeda descentralizada, oferecendo uma nova forma de realizar transações financeiras sem a necessidade de autoridades. Sua origem está ligada ao desejo de criar um sistema de pagamento eletrônico baseado em prova matemática, permitindo duas partes transacio-

narem si de maneira segura e anônima. O Bitcoin resolve vários problemas associados às moedas tradicionais e sistemas de pagamento, incluindo a questão da confiança, os custos de transação e a liberdade para enviar ou receber dinheiro globalmente.

Assim, sua implementação baseia-se fortemente em assinaturas digitais. Uma das funcionalidades críticas de carteiras é a capacidade de gerar e gerenciar assinaturas digitais. Assinaturas digitais são fundamentais para a segurança e integridade das transações em blockchains. Elas são usadas para verificar a autenticidade de uma mensagem ou transação, assegurando que não foram alteradas e confirmando a identidade do remetente.

O Bitcoin também introduziu a possibilidade de programar dinheiro através de scripts, uma funcionalidade que se expandiu com o desenvolvimento do Miniscript. O Miniscript permite uma programação mais acessível e segura dentro da rede Bitcoin, facilitando a criação de contratos inteligentes que podem executar funções complexas baseadas em condições pré-determinadas. Isso amplia significativamente as possibilidades de uso do Bitcoin, indo além das simples transações de pagamento para incluir acordos com múltiplas assinaturas e funções financeiras sofisticadas. Esse avanço não apenas fortalece a utilidade e a flexibilidade do Bitcoin como uma criptomoeda, mas também abre inspirou novas inovações financeiras descentralizadas.

2.1.3 Ethereum

O Ethereum (BUTERIN et al., 2013) inovou no universo das criptomoedas ao introduzir a ideia de executar programas diretamente na blockchain, que são conhecidos por serem Turing complete, ao contrário do que está presente no Bitcoin. Esses programas no são chamados de contratos inteligentes e para criá-los utiliza-se uma linguagem de programação chamada Solidity. A linguagem é estaticamente tipada, suporta herança, bibliotecas e tipos complexos definidos pelo usuário, o que facilita a escrita de códigos que são seguros e previsíveis dentro do ambiente de execução da Ethereum. Os contratos inteligentes compilados em Solidity são implantados na blockchain, onde qualquer pessoa pode interagir com eles de acordo com as regras definidas no contrato, promovendo um ambiente robusto para automação de processos financeiros.

2.1.4 Applicações com Blockchain: cofres e canais de pagamentos

Um exemplo prático de contrato inteligente é o *contrato de cofre*. Tradicionalmente, implementar um cofre digital exigiria APIs de pagamento, um servidor *backend* e um banco de dados para gerenciar as transações e estados. No entanto, com a tecnologia *blockchain*, um contrato de cofre pode ser implementado diretamente na rede e dessa forma possui fácil integração com a moeda nativa da rede. Esse contrato pode conter lógica para depósitos, saques e regras de acesso, todas executadas na blockchain, reduzindo a complexidade e

custos operacionais de possíveis integrações com terceiros (e.g, processadoras de pagamentos e outras instituições financeiras).

Avançando para aplicações mais complexas, os *contratos de canais de pagamento* representam uma evolução significativa na maneira como as transações são processadas. Estes contratos permitem pagamentos *off chain*, ou seja, transações que ocorrem fora da cadeia principal da blockchain, mas que são seguradas e validadas por ela. Essa abordagem não só aumenta a escalabilidade da rede ao reduzir o volume de transações que precisam ser processadas e confirmadas na *blockchain* principal, mas também demonstra a versatilidade dos contratos inteligentes em criar camadas adicionais de transação. Tais canais de pagamento podem facilitar transações quase instantâneas entre as partes, com a *blockchain* atuando como árbitro final apenas em caso de disputas. Isso ilustra claramente como a tecnologia de contratos inteligentes pode ser utilizada para construir sistemas financeiros mais eficientes e escaláveis.

2.1.5 Canais de Micropagamento no Ethereum

A literatura apresenta diferentes implementações de canais de micropagamento no Ethereum. Elsheikh, Clark e Youssef (2020b) propõe o EthWord, que adapta o Payword, detalhado na Seção 2.1.6, utilizando apenas funções *hash* e contratos inteligentes. O protocolo elimina a necessidade de confiança entre o Vendedor e Cliente ao usar contratos inteligentes para fixar o valor em aos *tokens* associados a moeda nativa da rede. Assim, o EthWord é capaz de reduz custos de transação para em cenários com mais de 16 pagamentos entre as mesmas partes, sendo viável até aproximadamente 1700 transações.

A implementação base do EthWord executa três etapas: inicialização do canal com smart contract e hash chain, envio de preimagens para pagamentos off-chain, e encerramento do canal para liquidação. O protocolo integra o paradigma "claim-or-refund"onde o pagador deposita fundos em um contrato que o recebedor pode sacar quando condições específicas são atendidas.

2.1.6 Payword: uma solução para micropagamentos

O conceito de *PayWord* (RIVEST; SHAMIR, 1996), resume-se em um protocolo de micropagamento criptográfico projetado para facilitar transações on-line de baixo valor de maneira eficiente e segura.

No núcleo do protocolo *PayWord*, encontra-se um mecanismo baseado em criptografia que permite a um usuário criar uma série de *paywords* a partir de um compromisso inicial. Este compromisso, juntamente com o primeiro *token*, é enviado ao vendedor como um compromisso de pagamento. As paywords subsequentes são utilizadas para realizar pagamentos incrementais sem a necessidade de comunicação direta com o a instituição

financeira em cada transação. Tudo isso é realizado através de uma função de *hash* unidirecional, onde cada *payword* subsequente é gerada a partir da anterior, garantindo segurança e permitindo a verificação rápida da validade das *paywords* pelo vendedor.

No protocolo PayWord, a *hashchain* é fundamental para gerar e validar micropagamentos, oferecendo um método eficiente e ao mesmo tempo seguro para a execução de transações de pequeno valor. A *hashchain* é uma sequência de valores onde cada valor é derivado pela aplicação de uma função de *hash* ao valor anterior. Uma *hashchain* é gerada a partir de um valor inicial aleatório, chamado *seed*. Os valores da *hashchain*, $H_n, H_{n-1}, \ldots, H_1$, são calculados como:

$$\begin{aligned} H_n &= \text{hash}(\textit{seed}), \\ H_{i-1} &= \text{hash}(H_i), \quad \text{para } i = n, n-1, \ldots, 2. \end{aligned}$$

Uma *hashchain* é iniciada com a geração de um valor aleatório, denominado *seed*, a partir do qual o usuário gera o último hash da cadeia, $H_n = hash(seed)$. O usuário então gera os hashes anteriores de forma recursiva, $H_{n-1} = hash(H_n), H_{n-2} = hash(H_{n-1}), \ldots, H_1 = hash(H_2)$. O primeiro hash H_1 é enviado ao vendedor como parte do compromisso inicial, juntamente com a assinatura digital do usuário sobre H_1 e outras informações relevantes (como a identidade do usuário e do vendedor, a data, etc.).

No ciclo de vida de um pagamento PayWord, o compromisso inicial representa a autorização do usuário para realizar pagamentos ao vendedor até um certo valor total. O compromisso inicial inclui o primeiro *hash* da cadeia (H_1) e é assinado digitalmente pelo usuário. A assinatura, σ, é calculada sobre o compromisso:

$$\sigma = \text{sign}_{\text{priv}}(H_1 || \text{info}),$$

onde $\text{sign}_{\text{priv}}()$ denota a função de assinatura digital usando a chave privada do usuário e info representa os dados como valor e o tamanho da cadeia.

Após o compromisso inicial, o usuário envia ao vendedor progressivamente os valores intermediários da *hashchain* (por exemplo, H_i), que serve como uma moeda, já que a qualquer momento é possível realizar a liquidação dessa promessa de pagamento. O vendedor, que já possui o primeiro *hash* H_1 e a assinatura digital do usuário, pode validar o pagamento seguinte verificando se $hash(H_i) = H_{i-1}$. Esta propriedade de pre-imagem da função de *hash* garante que apenas o detentor do *seed* original pode gerar os valores válidos da sequência, assegurando a integridade e a autenticidade dos pagamentos.

2.2 Desenvolvimento de Aplicações Web

Uma aplicação web é um programa ou software que é executado em um servidor web, em vez de ser instalado no dispositivo local do usuário. Este tipo de aplicação é acessado

por meio de uma rede, geralmente a Internet, usando um navegador web. As aplicações web oferecem a vantagem de não necessitar de instalação específica nos dispositivos dos usuários, permitindo acesso imediato e atualizações centralizadas.

A função principal das aplicações web é fornecer uma interface para usuários executarem tarefas específicas, como assistir a *streaming* e também realizarem pagamentos. Estas tarefas são realizadas por meio de um *front-end*, construído com linguagens nativas do navegador como HTML, CSS e Javascript, que por sua vez é capaz de se comunicar com o servidor, chamado de *back-end*. Do lado do servidor, os dados são validados, processados e armazenados. Essa arquitetura permite que aplicações web funcionem em qualquer plataforma ou dispositivo que possua um navegador compatível, destacando-se por sua flexibilidade e acessibilidade.

2.2.1 Construção dos Navegadores Web Modernos

Os navegadores web modernos implementam uma arquitetura multi-processo que separa a interface do usuário, o processamento de páginas e o armazenamento de dados. Esta arquitetura estabelece isolamento entre diferentes contextos de execução, garantindo segurança e estabilidade durante a navegação.

2.2.1.1 Extensões de Navegador

As extensões de navegador representam programas que estendem ou modificam as capacidades do navegador. Este modelo de extensibilidade implementa diferentes contextos de execução para garantir isolamento e segurança:

- **Content Scripts**: Executam no contexto da página web, permitindo o acesso e interação com o conteúdo da página.
- **Background Scripts**: Operam em um contexto persistente e isolado, gerenciando estado e processamento em segundo plano.
- **Local Storage**: Implementa uma API de armazenamento persistente que permite às extensões manter dados entre sessões do navegador.

A comunicação entre estes diferentes contextos ocorre através de um sistema de mensagens baseado em eventos, permitindo troca segura de dados entre os componentes da extensão.

2.2.2 TypeScript

O TypeScript, uma linguagem de programação desenvolvida pela Microsoft e lançada em 2012, é um superconjunto de JavaScript que adiciona tipagem estática opcional à

linguagem. A introdução do TypeScript visou melhorar o desenvolvimento de aplicações complexas, proporcionando uma ferramenta que facilita a detecção de erros e aumenta a produtividade do desenvolvedor (MICROSOFT, 2024). Desde a sua introdução, TypeScript tem visto uma adoção crescente, especialmente em projetos de larga escala que se beneficiam de suas características robustas de tipagem (SURVEY, 2023). A escalabilidade, segurança e facilidade de manutenção do código são características que têm contribuído significativamente para a popularização do TypeScript entre os desenvolvedores de software moderno (CHANDRAKAR, 2023). Este avanço torna o TypeScript uma escolha prevalente para o desenvolvimento *frontend*, reforçando sua posição como uma linguagem essencial para a construção de aplicações web que consultam API.

2.3 React e Next.js

React[1] é uma biblioteca JavaScript para construção de interfaces de usuário que revolucionou o desenvolvimento web através de seu modelo declarativo e baseado em componentes. Sua arquitetura eficiente utiliza o Virtual DOM para otimizar atualizações de renderização e facilita a criação de aplicações web complexas.

O desenvolvimento com React promove a reutilização de código através de componentes, que são blocos de construção independentes que encapsulam lógica e interface. Esta modularidade permite maior manutenibilidade e escalabilidade das aplicações. Com a introdução dos Hooks no React 16.8, a escrita de componentes funcionais tornou-se o padrão da indústria, simplificando o gerenciamento de estado e ciclo de vida.

2.3.1 Padrões de Gerenciamento de Estado em React

O React implementa diferentes padrões para gerenciamento e compartilhamento de estado entre componentes. O Context API estabelece um mecanismo para passar dados através da árvore de componentes sem necessidade de passar parâmetros manualmente em cada nível.

2.3.1.1 Context API

O Context API permite criar "contextos"que encapsulam dados e funcionalidades, disponibilizando-os para componentes descendentes sem acoplamento explícito. Este padrão implementa três conceitos principais:

- **Provider**: Componente que disponibiliza o contexto para seus descendentes
- **Consumer**: Componentes que consomem os dados do contexto

[1] Biblioteca JavaScript desenvolvida pelo Meta (anteriormente Facebook), <https://react.dev>

- **Hook useContext**: Interface para acessar o contexto em componentes funcionais

Esta arquitetura facilita a criação de módulos reutilizáveis e a composição de funcionalidades complexas em aplicações React.

2.3.2 Next.js

Next.js[2] emerge como uma extensão natural do React, oferecendo uma estrutura robusta para aplicações web full-stack. Ele resolve desafios comuns de desenvolvimento web moderno através de recursos incorporados como renderização híbrida, que inclui Server-Side Rendering (SSR), Static Site Generation (SSG), Incremental Static Regeneration (ISR) e Client-Side Rendering (CSR).

O framework implementa um sistema de roteamento baseado em arquivos, onde a estrutura de diretórios do projeto define automaticamente as rotas da aplicação. Além disso, oferece otimização automática de imagens e assets, melhorando significativamente o desempenho das aplicações em produção.

No frontend, o Next.js aprimora a experiência de desenvolvimento React com recursos como Hot Reloading e suporte integrado para diversas opções de estilização[3]. A estrutura do projeto segue convenções que promovem boas práticas de desenvolvimento e organização de código.

Para o backend, o framework implementa o conceito de API Routes, permitindo a criação de endpoints serverless diretamente no projeto. Esta integração seamless entre frontend e backend simplifica o desenvolvimento fullstack e permite deployments mais eficientes em plataformas modernas de hospedagem.

2.3.3 Sistemas de Gerenciamento de Banco de Dados

Os sistemas de gerenciamento de banco de dados (SGBD) fornecem uma interface sistemática para armazenar, recuperar e modificar dados. Estes sistemas implementam mecanismos de controle de concorrência, recuperação de falhas e segurança, garantindo a integridade e consistência dos dados armazenados.

O modelo relacional, proposto por Codd (1970), organiza dados em tabelas (relações) com linhas (tuplas) e colunas (atributos). Este modelo implementa a álgebra relacional para manipulação de dados e garante propriedades ACID (Atomicidade, Consistência, Isolamento e Durabilidade) nas transações.

[2] Framework React mantido pela Vercel, <https://nextjs.org>
[3] Suporta CSS Modules, Sass, Tailwind CSS e styled-components nativamente

O PostgreSQL[4] representa um SGBD objeto-relacional que estende o modelo relacional tradicional com recursos avançados como tipos de dados customizados, herança de tabelas e funções armazenadas. O sistema implementa completamente as propriedades ACID e suporta chaves estrangeiras, junções, triggers e procedimentos armazenados.

2.3.3.1 Object-Relational Mapping

O Object-Relational Mapping (ORM) estabelece uma técnica de programação que converte dados entre sistemas de tipos incompatíveis usando programação orientada a objetos. O Prisma[5] implementa uma camada de abstração que permite aos desenvolvedores interagir com bancos de dados usando uma API tipada, eliminando a necessidade de escrever SQL diretamente.

2.4 Streaming de Vídeo com HLS e HLS.js

O HTTP Live Streaming (HLS)[6] revolucionou a distribuição de conteúdo de vídeo na web ao introduzir um método adaptativo de streaming baseado em HTTP. Este protocolo segmenta o conteúdo de vídeo em pequenos fragmentos, normalmente de 2 a 10 segundos, e os disponibiliza através de requisições HTTP padrão, permitindo uma adaptação dinâmica à qualidade da conexão do usuário.

O HTTP Live Streaming (HLS)[7] é um protocolo de *streaming* adaptativo desenvolvido pela Apple Inc. Este protocolo divide o conteúdo de vídeo em pequenos segmentos, geralmente com duração de alguns segundos, e os disponibiliza através de requisições HTTP padrão. O HLS utiliza um arquivo de manifesto (*playlist*) no formato M3U8, que contém metadados sobre os segmentos de vídeo disponíveis. Este arquivo lista as URLs dos segmentos e inclui informações como resolução, bitrate e outras características do *stream*. O cliente HLS interpreta o manifesto e solicita os segmentos apropriados com base nas condições da rede e capacidades do dispositivo.

Para garantir a compatibilidade entre diferentes navegadores e fornecer uma experiência consistente de streaming, o projeto utiliza a biblioteca HLS.js[8]. Esta biblioteca implementa o cliente HLS em JavaScript, permitindo a reprodução de conteúdo HLS em navegadores que não possuem suporte nativo ao protocolo. O HLS.js gerencia automaticamente a adaptação da qualidade do vídeo, buffer de mídia e recuperação de erros, garantindo uma experiência de streaming robusta e adaptativa, essencial para o sistema de micropagamentos baseado em tempo de visualização.

[4] Sistema de banco de dados relacional de código aberto, <https://www.postgresql.org>
[5] ORM moderno para Node.js e TypeScript, <https://www.prisma.io>
[6] Protocolo de streaming desenvolvido pela Apple Inc., documentação oficial: <https://developer.apple.com/streaming/>
[7] Protocolo de streaming desenvolvido pela Apple Inc., <https://developer.apple.com/streaming/>
[8] Biblioteca JavaScript mantida pela Dailymotion, <https://github.com/video-dev/hls.js>

Esta solução permite que o sistema forneça uma experiência de streaming robusta e adaptativa, crucial para o funcionamento do sistema de micropagamentos por tempo de visualização. A capacidade do HLS de adaptar-se às condições da rede garante uma experiência contínua para o usuário, mesmo em conexões instáveis, enquanto a biblioteca HLS.js assegura a compatibilidade entre diferentes navegadores e dispositivos.

2.5 Bibliotecas para o desenvolvimento em blockchain

O desenvolvimento de aplicações descentralizadas na rede Ethereum requer a compreensão das ferramentas existentes. A Seção 2.5.1 apresenta o Viem como uma interface TypeScript para interações com a blockchain Ethereum. A Seção 2.5.2 introduz o Wagmi como ferramenta para o ecossistema React. Os hooks da biblioteca executam a integração entre interfaces de usuário e a blockchain. O Wagmi opera sobre o Viem e implementa operações em aplicações descentralizadas através da API com tipagem.

2.5.1 Viem: Interface TypeScript para Ethereum

O desenvolvimento de aplicações descentralizadas (dApps) na rede Ethereum requer ferramentas robustas e tipadas para garantir a integridade e segurança das interações com a blockchain. Neste contexto, o Viem[9] emerge como uma biblioteca moderna que oferece uma interface TypeScript completa para interações com a rede Ethereum. O Viem se destaca por sua arquitetura modular e forte tipagem, que proporciona maior segurança no desenvolvimento e melhor experiência para os desenvolvedores. A biblioteca suporta funcionalidades essenciais para dApps, incluindo gerenciamento de contas, interação com contratos inteligentes e integração com carteiras digitais[10]. Entre as principais características do Viem, destacam-se:

- Suporte completo a TypeScript com tipos nativos para ABI e dados on-chain
- Integração com múltiplas chains através de módulos específicos
- Implementação do padrão EIP-1193 para providers Ethereum
- Ferramentas para manipulação de ENS (Ethereum Name Service)
- Suporte a WebSocket para eventos em tempo real

A biblioteca também oferece recursos avançados para casos de uso específicos, como Account Abstraction e integração com SIWE (Sign-In with Ethereum). Esta flexibilidade

[9] Viem é uma interface TypeScript para Ethereum, desenvolvida por awkweb.eth e jxom.eth. Disponível em: <https://viem.sh>

[10] Documentação completa das funcionalidades disponível em: <https://viem.sh>, versão 2.21.35.

torna o Viem uma escolha versátil para diferentes tipos de aplicações blockchain, desde carteiras digitais até plataformas DeFi completas. Para garantir a segurança das operações, o Viem implementa as melhores práticas de criptografia e manipulação de chaves[11], além de fornecer utilidades para validação de inputs e sanitização de dados. A biblioteca é mantida ativamente pela comunidade e possui licença MIT[12], permitindo seu uso em projetos comerciais e de código aberto.

2.5.2 Wagmi: Hooks React para Ethereum

O desenvolvimento de interfaces Web3 requer integração entre React e Ethereum. As aplicações descentralizadas precisam gerenciar estados complexos, interagir com carteiras digitais e processar dados da blockchain de forma consistente.

O Wagmi[13] implementa hooks React para conectar interfaces com a blockchain Ethereum. A biblioteca automatiza tarefas como reconexão de carteiras, tratamento de erros e cache de dados. O suporte TypeScript garante detecção de erros e autocompleção em editores de código. A biblioteca implementa abstrações para operações blockchain através de hooks específicos. O useConnect gerencia conexões com carteiras, o useContract interage com contratos inteligentes, e o useTransaction processa transações.

A arquitetura do Wagmi complementa o Viem na construção de interfaces Web3. O Wagmi controla o ciclo de vida dos componentes React enquanto o Viem executa operações na blockchain. Esta separação de responsabilidades permite que desenvolvedores React implementem interfaces blockchain seguindo práticas do desenvolvimento web.

2.6 Ferramentas para hospedagem, estilo e autenticação

O desenvolvimento de interfaces modernas exige ferramentas que proporcionem flexibilidade e manutenibilidade ao código. O Tailwind CSS[14] revoluciona a estilização de aplicações web através de sua abordagem baseada em classes utilitárias. A aplicação de estilos diretamente no HTML, utilizando classes predefinidas, elimina a necessidade de arquivos CSS separados e reduz a complexidade do código. Esta metodologia resulta em um desenvolvimento mais ágil e uma base de código consistente, que facilita a manutenção e evolução do sistema.

[11] A biblioteca utiliza dependências confiáveis para operações criptográficas, como @noble/curves e @noble/hashes.
[12] Código fonte disponível em: <https://github.com/wevm/viem>
[13] Biblioteca React para desenvolvimento de dApps, mantida pela equipe wevm. Disponível em: <https://wagmi.sh>
[14] Framework CSS de código aberto, <https://tailwindcss.com>

2.6.1 Hospedagem e CDN

A infraestrutura da Vercel distribui o conteúdo globalmente através de sua rede de CDN, otimizando a entrega de recursos estáticos e dinâmicos. O sistema de cache inteligente reduz a latência e melhora a experiência do usuário em diferentes regiões geográficas. A plataforma executa otimizações automáticas de assets e implementa estratégias avançadas de caching.

2.6.2 Hospedagem, Integração e Deploy Contínuos

A infraestrutura moderna demanda soluções que integrem desenvolvimento e operações de forma eficiente. A Vercel[15] estabelece um ambiente completo de DevOps através de sua plataforma especializada em aplicações JavaScript.

O pipeline de CI/CD automatiza o processo de deploy e garante a qualidade do código em produção. A integração com repositórios Git permite previews automáticos para cada branch, facilitando a revisão de código e testes. O sistema de rollback protege contra deployments problemáticos, enquanto o monitoramento contínuo identifica e alerta sobre possíveis problemas de performance. Esta automatização do processo de deploy reduz erros humanos e acelera o ciclo de desenvolvimento.

2.6.3 Sistema de Autenticação

A segurança de aplicações web depende fundamentalmente de um sistema robusto de autenticação e autorização. O Clerk[16] implementa múltiplos métodos de autenticação, incluindo email/senha, provedores sociais e verificação em duas etapas. A plataforma disponibiliza componentes de interface pré-construídos e APIs seguras para gerenciamento de sessões e usuários. O sistema de autenticação estabelece uma camada de segurança essencial para as operações de micropagamentos, garantindo a integridade das transações e a proteção dos dados dos usuários.

[15] Plataforma de deploy e hospedagem, <https://vercel.com>
[16] Plataforma de autenticação e gerenciamento de usuários, <https://clerk.com>

3 Proposta de solução

Tendo em vista o crescente consumo de produtos digitais, fica evidente uma demanda crescente de uma infraestrutura robusta pagamentos digitais. Porém, ao analisá-los, os principais métodos de pagamentos seguem sendo o cartões de crédito, e, mais recentemente no Brasil, o PIX. Aliado a isso, com o advento de inovações de tecnologias financeiras, como a Blockchain, existe a possibilidade ir além de puramente pagamentos e transações e de fato inserir uma lógica fortemente atrelada ao dinheiro. Assim, o presente trabalho propõe um novo sistema de pagamentos que contribui com a implementação e também com a implementação de uma infraestrutura que possibilita micro-pagamentos escaláveis e com baixa carga cognitiva.

3.1 Atores

A análise estruturada do sistema de micropagamentos revela uma arquitetura tripartite, onde cada ator desempenha um papel crucial no fluxo de transações. O modelo proposto distribui responsabilidades específicas entre três entidades principais, cada uma com suas próprias capacidades e limitações, garantindo assim a integridade e eficiência do sistema como um todo. Os atores fundamentais são:

- **Usuário**: Consumidor final que adquire *tokens* do corretor e os utiliza para realizar micropagamentos em streaming, podendo executar múltiplas transações sequenciais com baixa latência
- **Vendedor**: Entidade que aceita *tokens* como forma de pagamento, pode acumulá-los e posteriormente convertê-los em moeda tradicional através do corretor, operando como beneficiário final das transações
- **Corretor**: Autoridade central do sistema responsável pela emissão, validação e conversão dos *tokens*, atuando como garantidor da integridade das transações e prevenindo gastos duplos

3.2 Requisitos

Em sistemas de micropagamentos, a clara definição dos papéis e responsabilidades dos atores é fundamental para garantir transações seguras e eficientes. O sistema proposto utiliza *tokens* como unidade de valor digital, emitidos e gerenciados por um corretor centralizado. Diferente do dinheiro tradicional, estes *tokens* operam sob um conjunto específico

de regras que permitem micropagamentos em alta frequência com validação síncrona. Esta abordagem possibilita *streaming* de pagamentos com baixa latência, mantendo a integridade do sistema através de validações em lote. O corretor atua como autoridade central, emitindo *tokens*, prevenindo gastos duplos e facilitando a conversão entre moeda digital e tradicional.

Com esta arquitetura, vendedores podem receber múltiplos micropagamentos dos usuários de forma contínua, sem que a validação de cada transação impacte o desempenho do sistema. Os requisitos a seguir detalham as funcionalidades essenciais de cada ator neste ecossistema:

RF1: O usuário deve poder enviar micropagamentos (*tokens*) para o vendedor com latência máxima de 100ms

RF2: O usuário deve poder visualizar seu saldo de *tokens* em tempo real

RF3: O usuário deve poder realizar até 1000 micropagamentos por segundo para um mesmo vendedor

RF4: O vendedor deve poder receber micropagamentos dos usuários sem necessidade de validação imediata

RF5: O vendedor deve poder solicitar a validação em lote dos tokens recebidos

RF6: O corretor deve poder emitir *tokens* com identificadores únicos e assinatura digital

RF7: O corretor deve validar lotes de *tokens* em até 5 segundos, independente do tamanho do lote

RF8: O corretor deve detectar e impedir tentativas de gasto duplo do mesmo *token*

RF9: O corretor deve processar solicitações de troca *token* em até 1 dia útil

3.3 Trabalhos relacionados

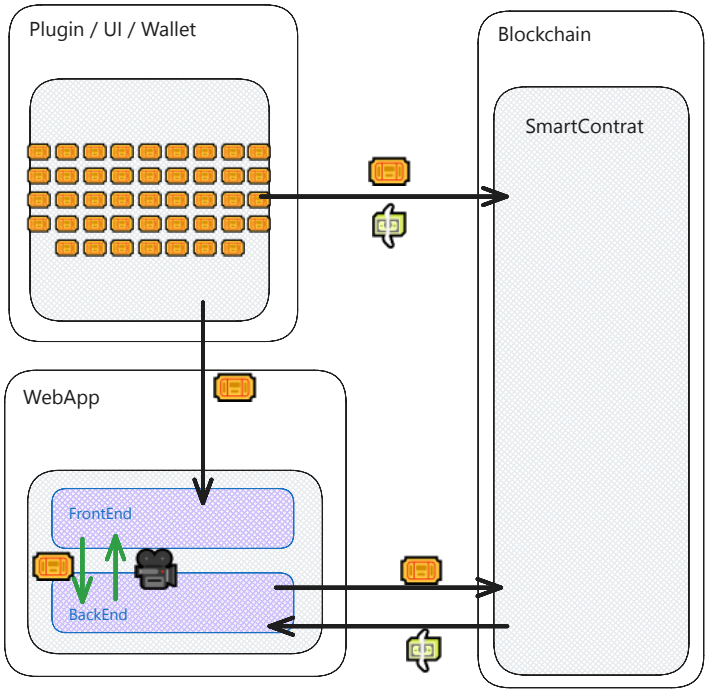
Néto (2002) explora o uso do PayWord para resolver problemas de escalabilidade e segurança em sistemas de pagamento online. Este estudo focou no PayWord como uma solução para facilitar micropagamentos eficientes e que possuem segurança, destacando sua potencialidade para simplificar transações de baixo valor para minimizar custos operacionais. A pesquisa desenvolveu uma prova de conceito usando a tecnologia Microsoft Access e teve sucesso em demonstrar a viabilidade de um sistema de micropagamentos, destacando a redução da carga cognitiva para os usuários e a simplificação do processo de transação.

O projeto ofereceu uma visão prática valiosa na época, mas, dado o avanço rápido da tecnologia, há agora oportunidades para adaptar e melhorar essa prova de conceito com ferramentas mais modernas e eficientes. Naquele período, a tecnologia blockchain não estava disponível, o que limitava a capacidade de testar o sistema em um ambiente mais amplo e avaliar seu desempenho em escala real. Portanto, nos dias atuais tal dissertação não apresenta muitas inovações, o que a torna limitada nesse contexto.

3.4 Solução técnica

O presente trabalho apresenta como proposta de solução a implementação de uma aplicação web que aceite pagamentos em fluxo. Para atingir os requisitos esperados, é utilizado uma abordagem baseada no PayWord, que utiliza o contrato inteligente na *blockchain* como emissor. Por fim, para possibilitar que o usuário utilize os *tokens*, é utilizado uma carteira virtual integrada no navegador. A visão geral da solução é definida pela Figura 2.

Figura 1 – Fluxo para compra de tokens



Fonte: os autores

3.4.1 Infraestrutura necessária

Para começar o processo de pagamentos, o Usuário deve adquirir os *tokens*. Essa aquisição é feita por uma ação em conjunto através da criação de um contrato inteligente,

na *blockchain*, e também por uma carteira digital, que é utilizada para gerenciar os *tokens*. Com isso o usuário utiliza uma interface que permite a interação com os contratos inteligentes na *blockchain*. Esse contrato é responsável por executar a lógica de validação dos *tokens* e permitir o saque pelo vendedor.

Para o Usuário, a carteira digital é utilizada para gerenciar os *tokens*. Também, permite a comunicação local com o *WebApp* sem a necessidade de um terceiro intermediando. Portanto, permite que o Usuário realize e pagamentos com a menor atrito possível. No que diz respeito ao Vendedor, é necessário um *WebApp*, contendo naturalmente o produto a ser oferecido e também a integração com restante do sistema. Por outro lado, o *WebApp* também permite verificação automatica a validade de cada *token* apresentadas pelos compradores, facilitando a aceitação e entrega do produto.

3.4.1.1 Ciclo de vidas dos *Tokens*

O ciclo de vida dos *tokens* é analogo ao ciclo de vida das Payword, apresentando na Seção 2.1.6. Todo o fluxo concentra-se na criação da *hashchain*, seguido na publicação do contrato inteligente, transmissão dos *tokens* e por fim o encerramento do canal de pagamentos. Assim, esse fluxo ser separado em 5 partes:

1. **Preparação para o compromisso**: Inicialmente, é necessária a criação da *hashchain*, utilizada no conceito de PayWord. Para facilitar a geração e gerenciamento desses dados, é o Usuário utiliza uma carteira digital que também é responsável pela comunicação com o *WebApp*.

2. **Realização do compromisso**: Em seguida, é necessário atribuir valor aos *tokens*. Para isso, é necessário utilizar uma agente intermediário que forneça e garanta o crédito. A utilização de um contrato inteligente, na blockchain, permite que isso ocorra. Elsheikh, Clark e Youssef (2020a) propõe um contrato inteligente, na rede Ethereum, onde é implementado a lógica do PayWord. Assim, é possível utilizas os *tokens* para cria-lo e ao final, executar a lógica de saque. Assim, os *tokens* criados pela carteira, possui valor, já que é possível converte-los para alguma outra uma moeda.

3. **Publicação ao vendedor**: o Usuário escolhe algum Vendedor que aceite a forma de pagamento e assim ao receber o *token* do usuário, juntamente com outras informações necessárias para fazer a validação do seu valor.

4. **Transmissão em fluxo**: Após a validação do valor, os próximos *tokens* é necessário validar apenas a propiedade da *hashchain*, assim como definido no PayWord.

5. **Término da transmissão**: Por fim, para que o Vendedor liquide o pagamento e receba seu devido pagamento, basta enviar o último *token* recebido ao contrato inteligente.

3.4.2 Fluxo de dados

A presente seção apresenta uma descrição detalhada de cada um dos processos envolvidos entre cada um dos componentes do sistema. Inicialmente é descrito o fluxo para a compra do *token*, passando pelo fluxo de transmissão para a aplicação web e finalmente o fluxo de saque do vendedor.

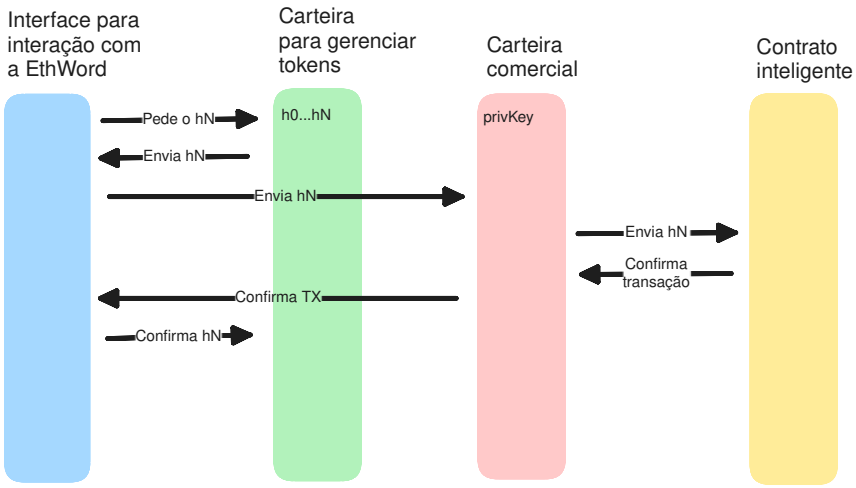
3.4.2.1 Fluxo para compra de *tokens*

Primeiro, o usuário acessa uma página de interface destinada à compra de *tokens*. Essa interface serve como o primeiro ponto de contato entre o usuário e o sistema de transações. A compra de *tokens* digitais é ilustrado pela Figura 2.

Assim, a página solicita um *token* para a carteira de PayWords, e após o envio, a interface procede para a construção da transação. Tendo a transação, a interface recorre a auxilio de alguma carteira comercial que, por sua vez, envia essa transação para a *blockchain*, onde interage com o contrato inteligente pré-estabelecido para governar essas operações.

Uma confirmada a transação na *blockchain*, a carteira comercial recebe a confirmação e a transmite de volta para a interface de usuário. A interface, ao receber essa confirmação, envia um sinal de sucesso para a carteira que gerencia os *tokens* do usuário, indicando que a compra foi realizada com êxito. Assim, após esse processo, os *tokens* terão algum valor atribuído, já que o contrato inteligente irá permitir trocá-los por alguma moeda real.

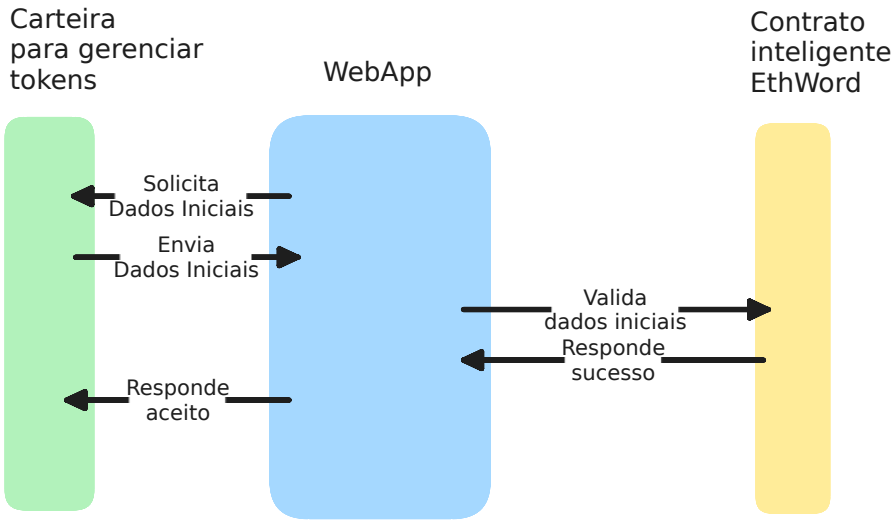
Figura 2 – Fluxo para compra de *tokens*



3.4.2.2 Fluxo para envio do primeiro token

Para que o Usuário comece a consumir o produto disponível no *WebApp*, é necessário um fluxo para tratar o primeiro *token*, que é ilustrado pela Figura 3. Ao acessar o *WebApp*, a página solicita ao usuário o *token* inicial. O usuário utiliza sua carteira de PayWords e autoriza o envio. Por fim, do lado do servidor, há a consulta dos dados recebidos com o Emissor. Após a validação, o *WebApp* finalmente responde informando para a carteira que o *token* está validado.

Figura 3 – Fluxo para envio do primeiro token

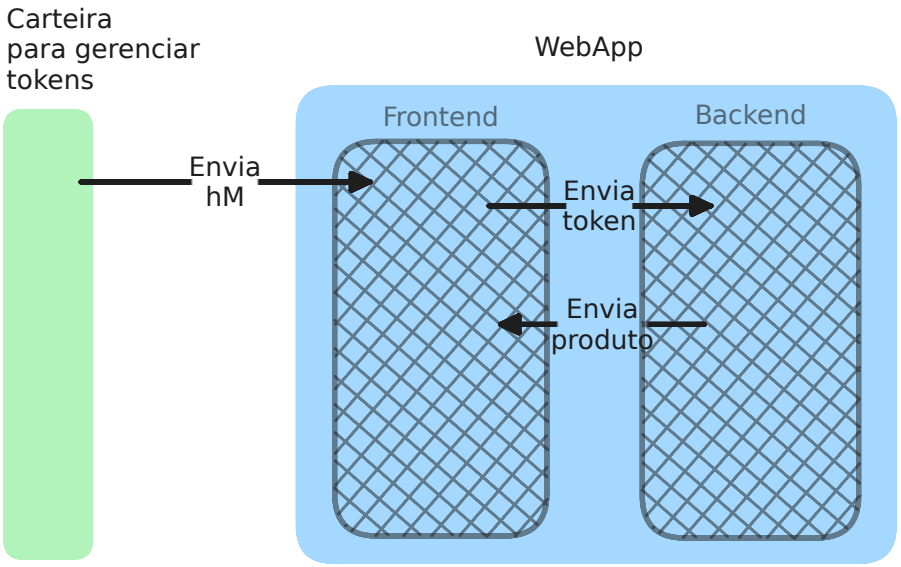


Fonte: os autores

3.4.2.3 Fluxo de micro-pagamentos

Após o primeiro *token*, o sistema guarda as informações referentes ao primeiro pagamento, assim seu fluxo fica reduzido, sem a necessidade de consultar o Emissor a todo instante, conforme ilustrado pelo fluxograma da Figura 4. Inicialmente, a página solicita a carteira o próximo *token*, e este é enviado e validado, assim como no passo anterior. Caso tenha sucesso, ira retornar o produto. Diferente do passo fluxo para enviar o primeiro *token*, agora não é mais necessário a comunicação com o emissor, já as propriedades da Payword garantem que a partir de um *token*, todos em sua sequencia podem ser validades localmente.

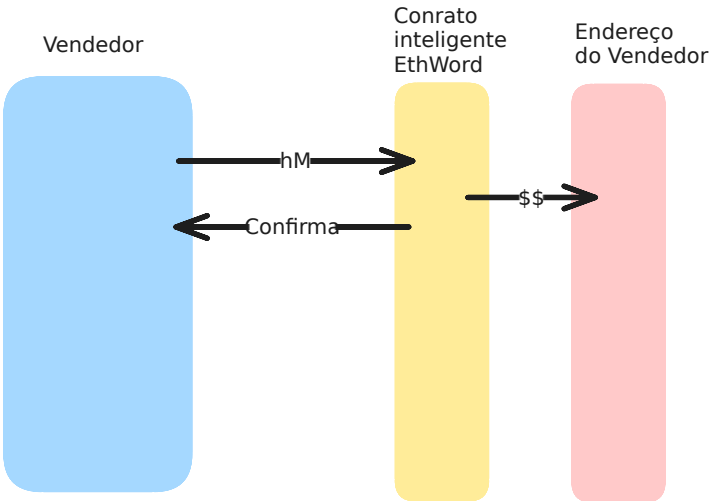
Figura 4 – Fluxo para micro-pagamentos



3.4.2.4 Fluxo de saque

Por fim, após encerrado os pagamentos em fluxo, o vendedor poderá realizar o saque, representado na Figura 5. Primeiro, o vendedor utiliza o último *token* recebido para interagir com o contrato inteligente, na *blockchain*. Após essa interação, o contrato automaticamente envia o dinheiro referente àquele *token* para o vendedor.

Figura 5 – Fluxo para compra de tokens



3.5 Plano de desenvolvimento

Para o desenvolvimento do projeto o seguinte cronograma foi utilizado, dividido nos 12 meses do calendário quadrimestral da Engenharia de Computação da Escola Politécnica da USP

- **Atividade 1: Introdução e Contextualização**
  - Descrição do cenário atual de sistemas de pagamento online e seus problemas.
  - Identificação das necessidades de inovação no campo de micro pagamentos.
- **Atividade 2: Proposta do Sistema**
  - Análise das soluções de pagamento existentes e suas limitações.
  - Seleção da tecnologia Payword e blockchain como soluções viáveis.
- **Atividade 3: Especificação de Requisitos**
  - Definição dos atores principais do sistema: usuário, vendedor e corretor.
  - Estabelecimento de requisitos mínimos para cada ator.
- **Atividade 4: Especificação Técnica**
  - Detalhamento do protocolo PayWord e sua aplicação em micropagamentos.
  - Detalhamento da criptografia e segurança das transações.
- **Atividade 5: Melhorias da biblioteca Typescript**
  - Implementação inicial do sistema de pagamentos PayWord.
- **Atividade 6: Integração com Blockchain**
  - Integração do protótipo com a tecnologia blockchain.
- **Atividade 7: Integração com o Payword**
  - Testes de funcionalidade e segurança do sistema integrado de micropagamentos.
- **Atividade 8: Desenvolvimento da Interface do Usuário no FrontEnd**
  - Criação de interfaces de usuário intuitivas para a gestão de micropagamentos com um escopo.
- **Atividade 9: Preparação para Publicação**
  - Finalização da documentação técnica e preparação para o deployment.
- **Atividade 10: Escrita e revisão do TCC**

– Escrita e revisão do trabalho de conclusão de curso (TCC).

Tabela 1 – Cronograma das Atividades

Atividades	Jan	Fev	Mar	Abr	Mai	Jun	Jul	Ago	Set	Out	Nov	Dez
At. 1	X											
At. 2	X	X										
At. 3			X									
At. 4		X	X									
At. 5				X	X	X						
At. 6					X	X						
At. 7							X	X				
At. 8								X	X			
At. 9										X	X	
At. 10			X	X	X	X	X	X	X	X	X	X

3.6 Implementação do Trabalho

Sistemas de streaming convencionais focam em fornecer um catálogo extenso de conteúdo, implementando funcionalidades como recomendações personalizadas, comentários e avaliações. O *MiniMoni*, entretanto, direciona seu foco para a implementação de um mecanismo de micropagamentos integrado ao streaming de vídeo, estabelecendo uma prova de conceito para monetização granular de conteúdo.

Para viabilizar o desenvolvimento e validação do sistema de micropagamentos, o projeto adota simplificações estratégicas no escopo da implementação. O sistema opera com um modelo reduzido onde existe apenas um vídeo disponível e um único provedor de conteúdo. Esta simplificação permite concentrar os esforços de desenvolvimento nos aspectos críticos do sistema de pagamentos, sem as complexidades adicionais de um catálogo completo.

Apesar das simplificações no catálogo e provedores, o sistema mantém suporte a múltiplos usuários consumidores, aproximando-se assim de um cenário real de operação. Para cada usuário, o sistema estabelece um canal de pagamento individual com o provedor único de conteúdo.

A implementação deste modelo simplificado, descrita em detalhes nas seções seguintes, demonstra a viabilidade técnica do conceito de micropagamentos integrados ao streaming de vídeo. O sistema estabelece as bases para futuras expansões que poderão incluir múltiplos provedores de conteúdo e um catálogo diversificado de vídeos.

3.6.1 Fluxo de Utilização

O fluxo de utilização do *MiniMoni* divide-se em três etapas principais: a abertura do canal de pagamento, a transmissão do conteúdo com micropagamentos e o fechamento do canal. Este processo implementa o protocolo *Payword* sobre canais de pagamento para viabilizar micropagamentos eficientes durante o streaming de vídeo.

1. **Abertura do canal:** A abertura do canal de pagamento inicia-se quando o usuário acessa a plataforma. O sistema gera localmente uma *hashchain* e implanta um contrato inteligente na blockchain. Este contrato estabelece os termos do canal de pagamento, incluindo o valor total bloqueado e o endereço do provedor de conteúdo. O provedor verifica a existência e a validade do contrato na blockchain antes de autorizar o início da transmissão.

2. **Transmissão do conteúdo:** Durante a transmissão do conteúdo, o sistema realiza micropagamentos através da liberação progressiva de hashes da *hashchain*. Para cada segmento de vídeo solicitado, o player anexa um hash específico ao cabeçalho da requisição HTTP. O provedor valida cada hash recebido reconstruindo a *hashchain* localmente e comparando com o valor inicial registrado no contrato.

3. **Fechamento do canal:** O fechamento do canal ocorre quando o provedor submete o último hash recebido ao contrato inteligente na blockchain. O contrato executa a validação final da *hashchain* e, em caso de sucesso, transfere o valor correspondente ao número de hashes utilizados para o endereço do provedor. Este processo de fechamento garante a finalidade dos pagamentos e libera os fundos remanescentes para o usuário.

A utilização de funções hash para validação, em substituição a assinaturas digitais tradicionais, reduz o custo computacional das operações de verificação. O processamento dos pagamentos ocorre fora da blockchain, minimizando taxas de transação e sobrecarga na rede. Esta arquitetura estabelece um mecanismo eficiente para micropagamentos em streaming de vídeo.

3.6.2 Implementação da Abertura de Canal

A implementação do *MiniMoni* segue a arquitetura de uma aplicação descentralizada (dApp), onde o *frontend* estabelece comunicação direta com a blockchain. Esta comunicação requer a integração com carteiras digitais como *MetaMask*[1].

O processo de abertura do canal inicia-se com a geração da *hashchain*. O sistema implementa uma extensão web que gerencia a criação e armazenamento das cadeias

[1] Interface de carteira digital que permite interações com aplicações Web3, <https://metamask.io>

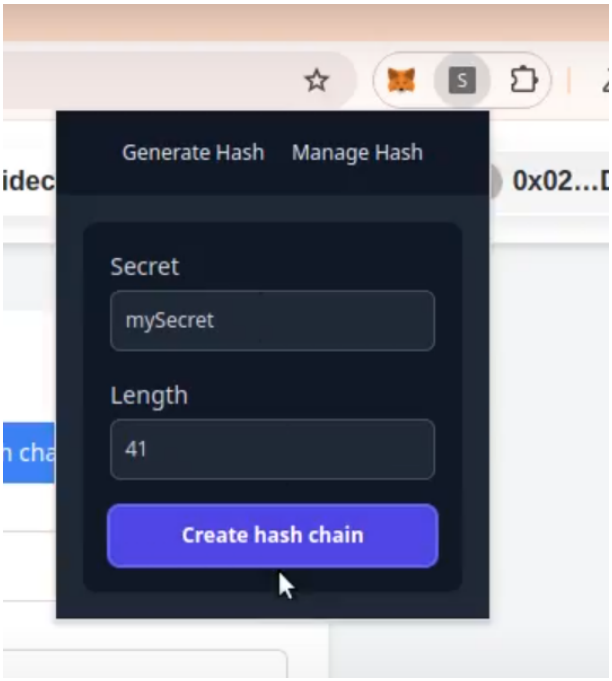


Figura 6 – Carteira para criação da cadeia de hashes

criptográficas. A interface da extensão, apresentada na Figura 6, permite que o usuário configure dois parâmetros fundamentais: o segredo inicial para geração da cadeia e o comprimento total desejado.

A extensão disponibiliza métodos de acesso à *hashchain* através de uma API JavaScript, permitindo a interação com as aplicações web. O usuário preenche um formulário com os parâmetros do canal: o valor total a ser bloqueado, o valor do último hash (cauda) e o tamanho da *hashchain*. O sistema executa a implementação do contrato inteligente na blockchain utilizando estes parâmetros. O endereço do contrato gerado armazena-se automaticamente na extensão, associado à *hashchain* correspondente.

O processo finaliza-se com o registro do canal no servidor de streaming. O usuário informa o endereço do contrato ao serviço, que executa um processo de validação utilizando a biblioteca *Viem*, apresentada na Seção 2.5. O *backend*, em NextJs, descrito na Seção 2.3.2, consulta o contrato na blockchain para verificar sua existência e parâmetros. Esta validação garante a integridade do canal antes de iniciar o streaming de conteúdo.

A estrutura do sistema distribui as responsabilidades entre componentes especializados: a extensão gerencia as *hashchains*, o contrato inteligente estabelece os termos do canal, e o servidor valida a configuração.

3.6.3 Pagamento em fluxo integrado com player de vídeo

O player de vídeo representa o núcleo da experiência do usuário, integrando streaming adaptativo com micropagamentos automáticos. A implementação utiliza a

biblioteca *HLS.js*, detalhada na Seção 2.4, para gerenciar o streaming de vídeo em segmentos. A integração com a biblioteca *Viem*, discutida na Seção 2.5, permite a geração e validação consistente de hashes criptográficos em todo o sistema.

O processo de streaming com micropagamentos inicia-se com a geração dos tokens criptográficos. Para isso, o player utiliza metadados essenciais guardado na extensção do navegador, como a semente da *hashchain* e seu comprimento. Assim, é possível produzir a sequência de tokens necessária para o streaming.

Durante a reprodução do vídeo, cada requisição de segmento implementa um mecanismo de autenticação baseado em tokens. O player anexa ao cabeçalho HTTP da requisição um token no formato `indexId:hash`, onde `indexId` representa a posição atual na *hashchain* e `hash` contém o valor criptográfico correspondente. A Figura 7 ilustra este processo de requisição na camada de rede.

Figura 7 – Estrutura da requisição HTTP para segmentos de vídeo com token de micropagamento

O *backend* valida cada token recebido antes de liberar o segmento de vídeo correspondente. Esta validação garante a integridade da *hashchain* e confirma que o pagamento para aquele segmento específico foi devidamente registrado. A cada nova requisição de segmento, o valor do token (`indexId:hash`) é atualizado, criando uma sincronização natural entre o processo de streaming e o sistema de micropagamentos.

Esta arquitetura permite que o consumo do conteúdo e os micropagamentos ocorram de forma simultânea e transparente para o usuário, enquanto mantém a segurança e a consistência do sistema de pagamentos. A integração entre o *HLS.js* e o *Viem* possibilita uma experiência fluida onde cada segmento de vídeo entregue corresponde a um micropagamento verificável na blockchain.

3.6.4 Implementação do Fechamento de Canal

O fechamento do canal de pagamentos ocorre por iniciativa do vendedor, independentemente do consumo total dos tokens disponíveis. Para isso, o painel administrativo , apresentado na Figura 8, permite que o vendedor acesse a interface de gerenciamento de

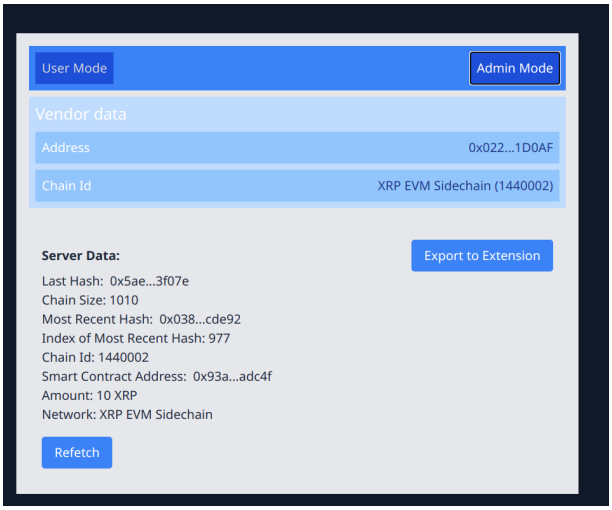


Figura 8 – Painel administrativo para exportação dos dados de liquidação

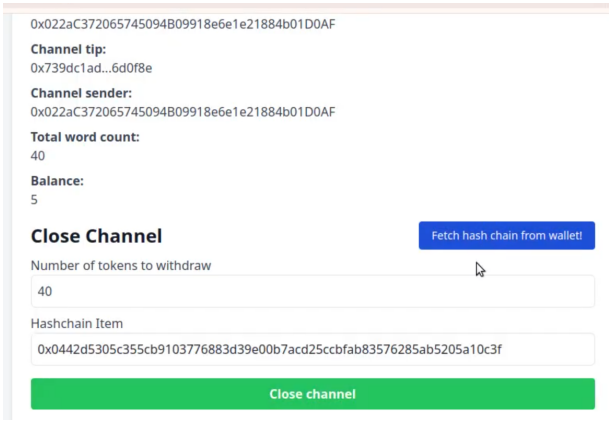


Figura 9 – Interface de fechamento do canal com validações de endereço

canais. O sistema exporta os seguintes dados para a extensão: o último token recebido, seu índice na *hashchain* e o endereço do contrato na blockchain.

Após armazenar os dado na extensão, o vendedor passa para uma nova interface de interação com o contrato. O sistema verifica se o endereço da carteira conectada corresponde ao endereço do beneficiário registrado no contrato inteligente. A Figura 9 demonstra o formulário de fechamento, que requer o endereço do contrato e o último token recebido.

O fechamento efetivo do canal ocorre através de uma transação na blockchain, assinada pela carteira do vendedor. O contrato inteligente executa a validação da *hashchain* e, em caso de sucesso, transfere o valor correspondente ao número de tokens utilizados para o endereço do provedor. Esta implementação, integrada com a biblioteca *Wagmi* apresentada na Seção 2.5, garante a segurança e a atomicidade da operação de fechamento.

O mecanismo de fechamento estabelece o ponto final do ciclo de micropagamentos, assegurando a correta distribuição dos valores entre as partes. A integração entre o painel

administrativo, a interface de fechamento e o contrato inteligente cria um processo seguro e auditável para a liquidação dos pagamentos.

3.6.5 Persistência de Dados

A implementação da persistência de dados no MiniMoni utiliza uma arquitetura em camadas para gerenciar informações dos usuários e seus pagamentos. O sistema adota o PostgreSQL, apresentado na Seção 2.3.3, como banco de dados principal, aproveitando suas capacidades de integridade referencial e suporte a transações ACID para garantir a consistência dos dados dos pagamentos.

A integração entre a aplicação Next.js e o banco de dados emprega o Prisma como ORM, discutido na Seção 2.3.3, estabelecendo uma interface tipada para operações no banco de dados. Esta camada de abstração permite que a aplicação interaja com o banco de dados de forma segura e eficiente, mantendo a tipagem estática do TypeScript em toda a pilha de desenvolvimento.

O modelo de dados implementa uma estrutura que unifica três aspectos fundamentais do sistema: autenticação do usuário, estado dos pagamentos e referências aos contratos inteligentes. Esta modelagem estabelece uma relação unívoca entre o usuário autenticado e seu histórico de transações, permitindo o rastreamento preciso do progresso dos pagamentos enquanto mantém as referências necessárias para validação na blockchain.

A estrutura de dados escolhida suporta o ciclo completo de pagamentos do sistema, desde o registro inicial do usuário até o encerramento do canal de pagamento. O PostgreSQL garante a durabilidade das transações, enquanto o Prisma fornece uma interface programática que simplifica as operações de leitura e escrita, mantendo a integridade referencial entre os diferentes aspectos do sistema.

3.6.6 Arquitetura da Extensão

A implementação da extensão do navegador segue uma arquitetura em camadas que permite interação segura com as aplicações web. O sistema utiliza Content Scripts e Background Scripts, apresentados na Seção 2.2.1, para estabelecer a comunicação entre o contexto da página web e o armazenamento persistente da extensão.

O Background Script atua como controlador central da extensão, gerenciando o estado das hashchains e a persistência de dados através do Local Storage do navegador. A integração com as aplicações web ocorre através do Content Script, que implementa listeners para eventos específicos e estabelece um canal de comunicação bidirecional entre a página e o Background Script.

Para unificar a interface de comunicação entre as diferentes aplicações e a extensão, o sistema implementa um módulo baseado no Context API do React, discutido na Seção 2.3.1.

Esta abstração encapsula toda a lógica de comunicação com a extensão em um contexto reutilizável, permitindo que tanto o dApp quanto a aplicação Next.js compartilhem a mesma interface de interação com a extensão.

O padrão de comunicação baseado em eventos estabelece um protocolo consistente para operações na hashchain, onde cada operação é representada por um evento específico que carrega os dados necessários. Esta arquitetura mantém o desacoplamento entre os componentes enquanto permite uma integração fluida entre a extensão e as aplicações web.

4 Conclusão

Primeiramente, o trabalho resultou na implementação do MiniMoni, um sistema de micropagamentos que integra o protocolo PayWord com contratos inteligentes na blockchain. Em essência, o sistema estabelece canais de pagamento através de contratos que executam a validação e liquidação das transações.

No que diz respeito ao desenvolvimento, o sistema envolveu a construção de três componentes principais. Especificamente, o primeiro utiliza TypeScript e a biblioteca Viem para estabelecer a comunicação com a blockchain. Por conseguinte, o segundo implementa a interface em React e Next.js que integra o streaming de vídeo ao sistema de pagamentos. Em seguida, o terceiro consiste em uma extensão para o navegador que gerencia a criação e validação das hashchains.

Em relação à arquitetura, o sistema executa três fluxos de operação distintos. Particularmente, o primeiro fluxo executa a abertura do canal de pagamento, onde o usuário realiza o depósito inicial. De fato, o segundo fluxo conduz a transmissão do conteúdo, onde o sistema processa os micropagamentos de forma automática. Como resultado, o terceiro fluxo executa o fechamento do canal, permitindo ao vendedor receber o valor correspondente ao conteúdo consumido.

No tocante à implementação do streaming, o sistema utiliza o protocolo HLS para transmissão de vídeo. Consequentemente, o sistema anexa os tokens de pagamento aos cabeçalhos HTTP de cada requisição de segmento de vídeo, mantendo a continuidade da reprodução sem interrupções para validação dos pagamentos.

4.1 Contribuições e Resultados Alcançados

Os resultados obtidos confirmam a viabilidade técnica do sistema proposto, atendendo aos principais requisitos funcionais estabelecidos. O sistema demonstrou capacidade de processar micropagamentos com baixa latencia (RF1), manter controle preciso do saldo de tokens (RF2), e garantir a segurança das transações através da blockchain (RF6, RF8).

Embora algumas funcionalidades inicialmente propostas, como a validação em lote (RF5), tenham sido implementadas de forma diferente da prevista inicialmente, as soluções alternativas desenvolvidas mostraram-se eficazes e mais adequadas ao contexto do streaming de vídeo.

O projeto alcançou seu objetivo principal de implementar um sistema funcional de micropagamentos para streaming de vídeo, adaptando o protocolo PayWord para tecnologias web modernas. As contribuições principais podem ser categorizadas em três

áreas fundamentais:

4.1.1 Implementação Blockchain do PayWord

Desenvolvemos uma implementação bem-sucedida do protocolo PayWord utilizando contratos inteligentes, que demonstrou ser capaz de:

- Prevenir efetivamente o gasto duplo através de validações no contrato inteligente
- Processar validações de *tokens* com custos de gás economicamente viável
- Executar micropagamentos com latência inferior a 100ms, compatível com streaming de vídeo

4.1.2 Inovação em Extensão Web

A extensão do navegador desenvolvida representa uma contribuição significativa para o ecossistema de pagamentos web, oferecendo:

- Gerenciamento eficiente de *hashchains* diretamente no navegador
- Integração transparente com players de vídeo HLS
- Sistema de acompanhamento em tempo real do saldo de tokens
- Capacidade de processar múltiplos micropagamentos por segundo

4.1.3 Arquitetura Web Moderna

A implementação estabeleceu uma arquitetura de referência para sistemas de micropagamento em aplicações web, demonstrando:

- Integração bem-sucedida entre TypeScript, React e contratos inteligentes
- Estrutura escalável para processamento de micropagamentos em streaming
- Sistema de validação eficiente com baixa latência
- Interface de usuário intuitiva para gerenciamento de pagamentos

4.2 Desafios superados

O desenvolvimento do sistema enfrentou diversos desafios técnicos significativos. Como há integração com diversos módulos independentes, como o próprio contrato específico para o Payword, extensão do navegador para geração e gerenciamento do Payword,

WebApp, etc, tudo isso demandou uma solução arquitetural complexa e com limitações devida as muitas interfaces de comunicação.

4.2.1 Integração com Streaming de Vídeo

A integração entre o player HLS.js e a extensão do navegador apresentou desafios únicos na implementação. Foi necessário desenvolver um mecanismo especializado para incluir *headers* de autenticação em cada requisição de vídeo, garantindo assim a continuidade do streaming sem comprometer a segurança do sistema. Para viabilizar esta solução, implementamos uma abordagem onde a *seed* é enviada inicialmente e toda a *hashchain* é gerada de forma dinâmica e declarado antes de iniciarmos o player de vídeo (i.e, iniciarmos a biblioteca *hls.js* .

Embora esta estratégia não apresente desafios significativos em termos de memória ou processamento, ela introduz vulnerabilidades potenciais do ponto de vista de segurança, uma vez que a *seed* poderia ser potencialmente interceptada através de métodos ainda não completamente estudados.

A solução ideal envolveria uma transmissão assíncrona, onde o player de vídeo requisitaria um *token* e aguardaria a emissão pela extensão. No entanto, a implementação atual gera toda a *hashchain* antes de iniciar o player, alternando subsequentemente o cabeçalho de autenticação, uma limitação que deriva diretamente desta escolha arquitetural. Durante o armazenamento do *hash*, adotou-se a representação em *string* com prefixo '0x', seguindo o padrão estabelecido pela biblioteca Viem. Esta biblioteca, que oferece compatibilidade universal com todos os ambientes de execução, simplificou significativamente a implementação através de sua padronização consistente.

4.2.2 Consistência Criptográfica Multi-ambiente

Um desafio particular surgiu da necessidade de manter consistência no cálculo de hashes através de diferentes ambientes de execução. A função Keccak, utilizada no Ethereum, precisa produzir resultados idênticos quando executada em três contextos distintos:

- No contrato inteligente (Solidity)
- No servidor backend
- No frontend da aplicação

Esta multiplicidade de ambientes trouxe à tona questões de compatibilidade na representação dos *hashes*, que podem variar entre formatos string, inteiro, binário e hexa-

decimal. A solução desenvolvida necessitou garantir a consistência destas representações em todas as camadas do sistema.

Um desafio adicional surgiu da própria implementação do Keccak no Ethereum, que difere do padrão SHA-3 estabelecido pelo NIST. Enquanto o SHA-3 padronizado pelo NIST (FIPS-202) utiliza um padding específico SHA3-256(M) = KECCAK [512] (M || 01, 256), o Ethereum manteve a implementação original da versão 3 da submissão Keccak. Esta diferença sutil mas crucial pode ser observada no seguinte exemplo:

```
Entrada: "testing"
Ethereum/Keccak-256:
 5f16f4c7f149ac4f9510d9cf8cf384038ad348b3bcdc01915f95de12df9d1b02
SHA3-256 (NIST):
 7f5979fb78f082e8b1c676635db8795c4ac6faba03525fb708cb5fd68fd40c5e
```

Esta divergência entre o Keccak do Ethereum e o padrão SHA-3 exigiu atenção especial durante a implementação, garantindo que todas as bibliotecas e ambientes utilizassem consistentemente a versão específica do Ethereum, evitando assim incompatibilidades na validação dos hashes através do sistema.

4.3 Limitações do Sistema Atual

Durante o desenvolvimento e implementação do sistema de micropagamentos para streaming de vídeo, diversas limitações e oportunidades de melhoria foram identificadas. Esta seção discute as principais restrições encontradas no sistema atual e propõe direções para trabalhos futuros que poderiam endereçar estas limitações e expandir as capacidades do sistema.

- **Custos de Blockchain Pública**: A dependência da rede introduz uma variabilidade significativa nos custos operacionais. As taxas de transação (*gas fees*) sofrem flutuações consideráveis dependendo do congestionamento da rede, podendo tornar o sistema economicamente inviável durante períodos de alto tráfego. Esta instabilidade nos custos representa um obstáculo significativo para a adoção em larga escala.

- **Latência de Validação**: O tempo necessário para confirmação de transações na blockchain pública pode comprometer a experiência do usuário, especialmente durante o estabelecimento inicial do canal de pagamento. Embora o sistema funcione com baixa latência após a validação inicial, este primeiro passo pode criar fricção desnecessária.

- **Barreira de Entrada**: A exigência de que usuários possuam e gerenciem criptomoedas representa um obstáculo significativo para adoção mainstream. O processo de aquisição e gestão de criptoativos ainda demanda conhecimento técnico além do razoável para um usuário típico de serviços de streaming.

- **Complexidade Operacional**: A interação com wallets e gestão de chaves criptográficas adiciona camadas de complexidade que poderiam ser simplificadas em uma solução mais centralizada. Esta complexidade afeta tanto o desenvolvimento quanto a experiência do usuário final.

4.4 Propostas para Trabalhos Futuros

Com base nas limitações identificadas e no potencial de evolução do sistema, várias direções para trabalhos futuros se apresentam como promissoras. Estas propostas visam tanto superar as restrições atuais quanto expandir as capacidades do sistema.

4.4.1 Blockchain Permissionada

Uma das principais direções para evolução do sistema envolve a exploração de alternativas à blockchain pública. A implementação em uma blockchain permissionada oferece possibilidades interessantes para otimização do sistema.

- **Controle Centralizado**: Uma rede permissionada permitiria que o emissor atuasse como autoridade central, mantendo controle sobre parâmetros críticos do sistema e otimizando o desempenho para o caso específico de micropagamentos.

- **Otimização de Custos**: As taxas de transação poderiam ser significativamente reduzidas ou mesmo eliminadas, tornando o sistema mais viável para conteúdo de baixo valor monetário.

- **Maior Previsibilidade**: O tempo de confirmação das transações se tornaria mais consistente e previsível, melhorando a experiência geral do usuário.

4.4.2 Otimizações Técnicas

A implementação atual realiza validações individuais dos *tokens*, o que pode ser ineficiente em termos de custos operacionais e processamento. Uma abordagem promissora seria a implementação de Árvores de Merkle para validação em lote. Esta estrutura permitiria a verificação de múltiplos *tokens* em uma única operação, reduzindo significativamente o número de interações com a blockchain. Por exemplo, um conjunto de mil *tokens* poderia ser validado através de uma única raiz Merkle, ao invés de mil validações individuais. Além

disso, as provas de Merkle forneceriam um método eficiente para demonstrar a pertinência de um *token* específico ao conjunto validado, mantendo a segurança do sistema enquanto reduz custos operacionais.

4.5 Considerações Finais

Este trabalho apresentou uma implementação moderna do protocolo PayWord para micropagamentos em streaming de vídeo, demonstrando sua viabilidade técnica através de uma arquitetura que combina contratos inteligentes, extensões de navegador e tecnologias web modernas. A implementação alcançou seus objetivos principais, estabelecendo um sistema funcional capaz de processar micropagamentos com baixa latência e manter a integridade das transações.

O sistema desenvolvido demonstrou que é possível adaptar protocolos clássicos de micropagamento para contextos contemporâneos, aproveitando a infraestrutura blockchain para garantir segurança e confiabilidade. A integração bem-sucedida com players de vídeo HLS mostrou que micropagamentos podem ser incorporados de forma transparente em aplicações de streaming, abrindo possibilidades para novos modelos de monetização de conteúdo.

No entanto, desafios significativos foram identificados, particularmente relacionados aos custos operacionais em blockchains públicas e à complexidade de interação para usuários não técnicos. Estas limitações sugerem direções importantes para pesquisas futuras, incluindo a exploração de blockchains permissionadas e otimizações técnicas através de estruturas de dados mais eficientes.

A experiência adquirida durante o desenvolvimento deste projeto indica que o sucesso de sistemas de micropagamento depende não apenas de sua correção técnica, mas também de considerações práticas sobre usabilidade e viabilidade econômica. O equilíbrio entre descentralização, eficiência e facilidade de uso emerge como um aspecto crucial para a adoção em larga escala de tais sistemas.

Por fim, este trabalho contribui para o campo de sistemas de pagamento não apenas através de sua implementação técnica, mas também pela identificação de desafios práticos e oportunidades de melhoria que podem guiar desenvolvimentos futuros nesta área.

Referências

BUTERIN, V. et al. Ethereum white paper. *GitHub repository*, v. 1, p. 22–23, 2013. Citado na página 26.

CHANDRAKAR, S. *Mastering TypeScript: 10 Essential Features for Writing Better Code*. 2023. <https://bootcamp.uxdesign.cc/mastering-typescript-10-essential-features-for-writing-better-code-c89719d6dcc7>. Acesso em: 21 abr 2024. Citado na página 30.

CHAUM, D.; FIAT, A.; NAOR, M. Untraceable electronic cash. In: *Advances in Cryptology - CRYPTO '88, 8th Annual International Cryptology Conference, Santa Barbara, California, USA, August 21-25, 1988, Proceedings*. [S.l.]: Springer, 1988. (Lecture Notes in Computer Science, v. 403), p. 319–327. Citado na página 25.

CLOUDWARDS. Music streaming services statistics 2022. *Cloudwards.net*, 2021. <https://www.cloudwards.net/streaming-services-statistics/>. Citado na página 19.

CODD, E. F. A relational model of data for large shared data banks. *Commun. ACM*, Association for Computing Machinery, New York, NY, USA, v. 13, n. 6, p. 377–387, jun. 1970. ISSN 0001-0782. Disponível em: <https://doi.org/10.1145/362384.362685>. Citado na página 31.

DemandSage. Live streaming and live commerce industry growth. 2023. <https://www.demandsage.com/live-streaming-statistics/>. Citado na página 19.

ELSHEIKH, M.; CLARK, J.; YOUSSEF, A. M. Short paper: Deploying payword on ethereum. In: SPRINGER. *Financial Cryptography and Data Security: FC 2019 International Workshops, VOTING and WTSC, St. Kitts, St. Kitts and Nevis, February 18–22, 2019, Revised Selected Papers 23*. [S.l.], 2020. p. 82–90. Citado 2 vezes nas páginas 22 e 40.

ELSHEIKH, M.; CLARK, J.; YOUSSEF, A. M. Short paper: Deploying payword on ethereum. In: BRACCIALI, A. et al. (Ed.). *Financial Cryptography and Data Security*. Cham: Springer International Publishing, 2020. p. 82–90. ISBN 978-3-030-43725-1. Citado na página 27.

KARRER, F. What is the average number of streaming services per u.s. household? *MNTN*, 2024. <https://mountain.com/average-number-of-streaming-services-per-us-household/>. Citado na página 19.

MICROSOFT. *Visão geral do TypeScript*. 2024. <https://learn.microsoft.com/pt-br/training/modules/typescript-get-started/2-typescript-overview>. Acesso em: 21 abr 2024. Citado na página 30.

Mordor Intelligence. *Online Advertising Market*. 2023. <https://www.mordorintelligence.com/pt/industry-reports/online-advertising-market>. [Accessed 17-03-2024]. Citado na página 19.

NAKAMOTO, S. Bitcoin whitepaper. v. 9, p. 15, 2008. Disponível em: <https://bitcoin.org/bitcoin.pdf>. Citado na página 25.

NéTO, J. C. *Criptografia: uma implementação do Protocolo de Micropagamento PayWord*. Dissertação (Dissertação de Mestrado) — Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 5 2002. Disponível em: <https://doi.org/10.11606/D.45.2002.tde-20210729-130315>. Citado 2 vezes nas páginas 22 e 38.

RIVEST, R. L.; SHAMIR, A. Payword and micromint: Two simple micropayment schemes. In: *Security Protocols Workshop*. [s.n.], 1996. Disponível em: <https://api.semanticscholar.org/CorpusID:3219009>. Citado 2 vezes nas páginas 22 e 27.

STRUGAR, D. et al. On m2m micropayments: A case study of electric autonomous vehicles. In: *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*. [S.l.: s.n.], 2018. p. 1697–1700. Citado na página 19.

SURVEY, D. *Global Usage of TypeScript*. 2023. <https://survey.stackoverflow.co/2023/>. Acesso em: 21 abr 2024. Citado na página 30.