UNIVERSIDADE FEDERAL DO ABC

TRABALHO DE GRADUAÇÃO EM

ENGENHARIA DE INFORMAÇÃO

# Uso de Canais de Pagamento em Blockchain na Implementação de uma Loja Online

Victor Hugo da Costa Leite

Santo André, SP

2021

*Victor Hugo da Costa Leite*

# Uso de Canais de Pagamento em Blockchain na Implementação de uma Loja Online

Monografia apresentada ao curso de Engenharia de Informação da Universidade Federal do ABC como parte dos requisitos para a obtenção do grau de Engenheiro.

UNIVERSIDADE FEDERAL DO ABC

Orientador:
Prof. Dr. João Henrique Kleinschmidt

Santo André, SP

2021

*À minha mãe que mesmo depois tanto tempo sinto como se cuidasse de mim e me guiasse até hoje.*

# Agradecimentos

Agradeço a Deus por ter me dado forças quando eu precisei; à minha namorada Anne pela parceira, motivação e paciência; a minha família pelo suporte e ao Prof. Dr. João Henrique Kleinschmidt por toda orientação, disponibilidade e compreensão para a conclusão deste trabalho.

# Resumo

O uso de moedas digitais como depósito de valor e especulação tem se tornado cada vez mais habitual por parte da população desde a criação do bitcoin em 2008. Tendo blockchain como a base de moedas digitais, a descentralização e consequente independência de governos é um dos pontos mais disruptivos e de agregação de valor comparado a moedas tradicionais vinculadas à economia de países. Apesar do seu uso ter se mostrado seguro na última década, moedas digitais ainda são pouco usadas como meio de pagamento. Limitações intrínsecas à tecnologia blockchain contribuem para isso, como a falta de escalabilidade, taxas de transações e tempo de confirmação. Visando superar essas limitações, o conceito de *Payment Channel Networks* (PCN) foi criado. PCN funciona como uma segunda camada em cima da blockchain onde transações podem ocorrer de maneira segura somente entre as partes envolvidas sem a necessidade de propagar ela por toda a blockchain resolvendo, de certa forma, o problema de escalabilidade. Este trabalho tem como objetivo a implementação uma loja online que aceita pagamentos através de PCNs da blockchain Ethereum. Para isto, foram estudados os conceitos fundamentais de blockchain e Raiden que é o framework que implementa o conceito de PCN na blockchain Ethereum. Dos experimentos, pode-se averiguar que os pagamentos são tão rápidos como métodos tradicionais, mas que ainda assim tem suas limitações pois há a necessidade de submissão de transações na blockchain para fazer depósitos e saques de tokens em canais de pagamento as quais necessitam passar pelo consenso da rede.

# Abstract

The use of digital currencies as a deposit of value and speculation has become increasingly common among the population since the creation of bitcoin in 2008. Having blockchain as the basis of digital currencies, decentralization and consequent independence of governments is one of the most disruptive and value-adding points in relation to traditional currencies linked to the economy of countries. Despite its use having been defined as safe in the last decade, coins are still little used as a means of payment. Intrinsic limitations to blockchain technology contribute to this as the lack of scalability, transaction rates and confirmation time. In order to overcome these limitations, the concept of Payment Channel Networks (PCN) was created. PCN works as a second layer on top of the blockchain where transactions can occur securely only between the parties involved without the need to propagate it throughout the blockchain, solving, in a way, the scalability problem. This work aims to implement an online store that accepts payments through PCNs from the Ethereum blockchain. For this, the fundamental concepts of blockchain and Raiden as the framework that implements the PCN concept in Ethereum blockchain were studied. From the experiments it can be seen that payments are as fast as traditional methods but that still has its limitations as there is a need to submit transactions on the blockchain to make deposits and withdrawals of tokens from payment channels which need to go through the consensus of the network.

# Lista de Figuras

# Lista de Tabelas

# Lista de Abreviaturas e Siglas

| | |
|---|---|
| PCN | Payment Channel Network |
| DApps | Decentralized Applications |
| EVM | Ethereum Virtual Machine |

# Sumário

# CAPÍTULO 1

# Introdução

Blockchain nasceu em 2008 com a introdução do Bitcoin [3] e na última década ganhou repercussão e importância a ponto de ser a plataforma de diversas aplicações utilizadas por empresas e governos. Blockchain é o nome da tecnologia de blocos encadeados que são compartilhados por nós em uma rede *peer-to-peer* (P2P). Cada bloco possui um conjunto de informações que depois de construído e inserido na blockchain, suas informações não podem ser alteradas e estarão replicadas e acessíveis por todos os nós da rede P2P [4].

Uma das aplicações blockchain mais famosas e mais utilizadas atualmente são as criptomoedas ou moedas digitais. Blockchain foi a primeira tecnologia que permitiu a existência de um ativo digital, pois evita a replicação do ativo e seu gasto duplo, sendo que antes do surgimento da tecnologia era complexo de ser evitado[4]. A blockchain tem um processo de validação de blocos com novas transações chamado de mecanismo de consenso. O consenso é alcançado quando a maioria dos nós da rede P2P validam que a transação que se está tentando inserir não é uma transação maliciosa, como por exemplo fazendo um gasto duplo[1].

Moedas físicas, geralmente estão sob o controle de uma entidade governamental podendo ter o seu valor alterado com a impressão de dinheiro. Além disto, a sua movimentação financeira em larga escala necessita de instituições financeiras de confiança para intermediar as transações onde há a incidência de taxas devido custo operacional diminuindo a liberdade de uso do dinheiro. Por funcionarem em cima de uma rede P2P, moedas digitais tem como característica a descentralização[3]. Transações financeiras com moedas digitais podem ser executadas sem entidades de confiança, como bancos e o banco central, pois a própria rede P2P irá validar as transações pelo mecanismo de consenso e impedir ações maliciosas[3].

Apesar de moedas digitais possuírem valor financeiro, atualmente o seu uso como meio de pagamento não é muito corriqueiro. Uma transação financeira utilizando moeda digital é demorada, pois precisa esperar o mecanismo de consenso agir numa escala global na rede P2P, não está livre de custo pois, a inserção de uma transação na blockchain precisa remunerar os mineradores e não é escalável uma vez que as blockchains tradicionais conseguem processar no máximo 15 transações por segundo, o que é muito menor do que a média de processamento da Visa de 1700 transações por segundo[5–7].

Como uma solução para essas limitações inerentes à tecnologia de blockchain foi criada uma segunda camada em cima da blockchain chamada de *Payment Channel Network* (PCN)[5]. Por estar sobre a blockchain, PCN possui todas as qualidades da blockchain como descentralização, integridade e auditabilidade das mensagens. PCN retira a necessidade de toda transação financeira precisar ser validada pelo processo de consenso, podendo ser feita apenas entre as entidades envolvidas. Sem a necessidade de passar pelo processo de consenso, um pagamento pode ser feito quase instantaneamente, de forma escalável e, se as partes concordarem, sem custo. O único custo envolvido é o de abrir e fechar um canal de pagamento com um remetente que precisa ser feito por meio de uma transação na blockchain[8].

A tecnologia PCN, permitindo pagamentos rápidos e sem custo, torna viável o conceito de micropagamentos. Um micropagamento é definido como um pagamento de baixo valor financeiro. Tradicionalmente, micropagamentos tem o problema do valor do pagamento ser muitas vezes próximo ou menor do que o valor das taxas envolvidas cobradas por entidades bancárias em uma operação. Com a possibilidade de pagamentos serem feitos sem custo, micropagamentos podem ser mais usados possibilitando que o consumo de produtos possam ser feitos com uma granularidade maior.

Este trabalho tem o objetivo de implementar uma loja online que aceita pagamentos via canais de pagamento sobre a blockchain Ethereum. Para isto, foram estudados os fundamentos de uma blockchain e o funcionamento do framework Raiden que implementa o conceito de canais de pagamento.

No capítulo 2 é apresentada uma fundamentação teórica a respeito de blockchain. No capítulo 3 é feito um descritivo de como blockchain é usado em aplicações financeiras com foco na blockhain Ethereum, apontando as limitações para uso de moedas digitais como meio de pagamento e a introdução do conceito de PCN com a implementação feito pelo Raiden Network. No capítulo 4 está descrito o projeto de uma loja virtual que aceita como pagamento tokens da Ethereum transferidos via Raiden. No capítulo 5 é mostrado o fluxo de compra de um produto usando a interface gráfica da loja implementada e o pagamento usando a interface gráfica do Raiden. Este capítulo também faz uma breve análise dos custos operacionais e tempos para a realização do pagamento. Por fim, no capítulo 6

são discutidas considerações a respeito do sistema desenvolvimento e dos experimentos realizados, destacando os resultados obtidos e as limitações superadas e não superadas.

# CAPÍTULO 2

# Fundamentação Teórica

## 2.1 Blockchain

Blockchain é uma tecnologia emergente que se baseia no uso de uma estrutura de dados distribuída e compartilhada entre nós de uma rede *peer-to-peer* (P2P) para a criação e armazenamento de transações feitas entre usuários da aplicação blockchain [1]. Cada nó possui a mesma lista de blocos da blockchain em que cada bloco possui um conjunto de transações. Os nós podem inserir e validar novos blocos na blockchain sem nunca poderem remover ou alterar blocos já existentes o que garante um histórico íntegro e acessível de dados em todos os nós.

Os nós da blockchain são encarregados de fazerem a criação e validação de novos blocos por meio de técnicas de consenso que garantem uma consistência e sincronização eventual dos dados da blockchain ao longo do tempo. É o mecanismo de consenso que permite que nós não mutuamente confiáveis cheguem a um comum acordo dado que a maioria dos nós concordem [1].

## 2.2 Estrutura de dados

A estrutura de dados do blockchain, como o nome sugestivamente diz, é uma lista ligada de blocos em que o bloco mais recente aponta para o bloco anterior e assim sucessivamente até o primeiro bloco da blockchain que é chamado de *genesis* [1].

Cada bloco da blockchain é constituído por um cabeçalho e por um conjunto de transações organizados em uma árvore de Merkle. Um cabeçalho simplificado de um bloco possui

informações como: o hash do bloco anterior, um número pseudoaleatório chamado de nonce, e o hash da raiz da árvore de Merkle das transações do bloco. [1]. O *nonce* tem o propósito de dar variabilidade ao hash calculado do bloco.

Como já dito, o conjunto de transações é organizado em uma árvore de Merkle onde as folhas da árvore são as transações propriamente ditas. O segundo nível da árvore é composto pelo hash de cada uma das transações e os demais níveis são os hash de pares de hash dos níveis anteriores até que se chegue na raiz da árvores, cujo hash é armazenado no cabeçalho do bloco. A figura 2.1 mostra uma representação de um bloco da blockchain e a figura 2.2 representa um cadeia de blocos interligados pelo hash do bloco anterior.

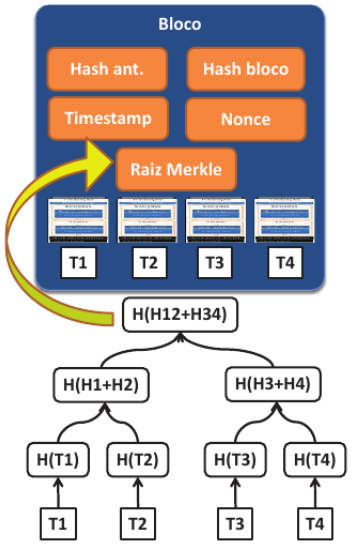


Figura 2.1: Estrutura de um bloco da blockchain. Fonte: [1]

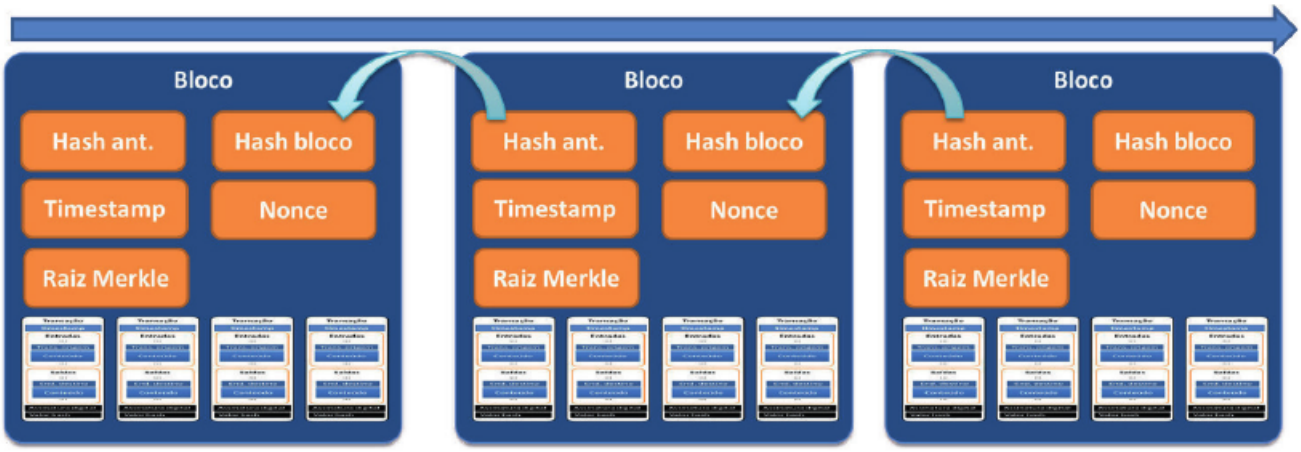


Figura 2.2: Representação da cadeia de blocos da blockchain. Fonte: [1]

Uma função *hash* possui a propriedade de **resistência à colisão** em que idealmente é impossível encontrar dois *hashs* iguais de tal forma que $H(a) = H(b)$ onde $a$ e $b$ são mensagens

diferentes e propriedade de **ocultação** em que dado $H(a)$ é inviável encontrar $a$. O uso de funções hash na árvore de Merkle das folhas até a sua raiz faz com que possamos conferir a integridade das transações presentes no bloco, isto é, conferir se não houve alteração na transação depois de inserida no bloco. Para isso basta ir calculando o hash das folhas até o hash da raiz e comparar com o hash presente no cabeçalho do bloco. Para certificar que as transações estão íntegras basta que o hash resultante da raiz seja igual ao hash presente no cabeçalho.

Além disso, também é calculado o hash do bloco utilizando todos os dados do cabeçalho. Como o bloco seguinte conterá o hash do bloco atual, para verificar a integridade do bloco como um todo basta comparar o hash do bloco atual contido no bloco seguinte com o hash calculado do bloco atual. Assim, com os cálculos de hash de um bloco feito em $O(log(n))$ pode-se verificar a integridade de um bloco e de toda uma cadeia de blocos [1].

O conjunto de transações armazenadas em uma blockchain, também chamadas de livro razão [4], são texto livre onde a semântica vai de acordo com as necessidades da aplicação que está sendo construída [1]. No caso de uma aplicação de pagamentos muito provável que o conjunto de transações se pareça com um livro contábil que contém: "um timestamp, o identificador (hash) da transação anterior de onde vem o valor de entrada (pode haver mais de um), o valor de entrada, o valor de saída, o endereço de destino (que vai receber o crédito), e uma assinatura digital feita com a chave privada do criador da transação (o debitado)" [1]. A blockchain somente garante que os dados inseridos nela não podem ser alterados ou trocados de ordem. Quaisquer outras validações sobre os dados que sejam mais específicas devem ser de responsabilidade da camada de aplicação.

A assinatura de uma transação confere a autorização, autenticidade e irrefutabilidade da pessoa que está transferindo [1]. Geralmente as assinaturas são feitas utilizando assinaturas de chave pública e chave privada o que torna possível o requisitante da transação assinar uma transação com a chave privada e outros validarem a assinatura utilizando a chave pública.

No caso de uma aplicação financeira, uma última etapa de validação ocorre verificando se a mesma transação com os mesmos valores referenciados já não foi utilizada, o que implicaria em gasto duplo. Para isso, é necessário efetuar uma busca na blockchain começando no bloco da transação referenciada até o último bloco da estrutura.

Outros campos que podem estar presentes nos blocos de uma blockchain são um campo de timestamp do momento que o bloco é criado e metadados em geral que podem conter a chave pública de quem fez a *mineração* do bloco e outras informações que dependem da blockchain[9].

## 2.3 Propriedades

A tecnologia blockchain permite que aplicações construídas em cima dela tenham algumas propriedades intrínsecas. As principais são encontradas abaixo conforme [4]:

- Descentralização: Todos os nós da rede P2P possuem uma cópia da base de dados distribuída e dependendo do modelo de blockchain todos os nós conseguem validar transações feitas entre nós, não confiáveis entre si, sem a necessidade de uma terceira entidade confiável.
- Disponibilidade: Como a base de dados de transações é distribuídas entre todos os nós da rede e a criação de blocos também consegue ser feita por todos os nós da rede sem a dependência de uma entidade única, caso alguns nós fiquem indisponíveis por algum motivo o restante da rede ainda consegue fazer todo o trabalho necessário para que a blockchain não pare de funcionar.
- Integridade: Todas as transações são assinadas com a chave privada do autor e validadas posteriormente por outros nós da rede P2P fazendo com que todas as transações inseridas no blockchain sejam íntegras. Além disso, existem a noção de ordem de criação de transações graças a verificação de hash do bloco anterior o que permite validar a ordem geral das transações também.
- Auditabilidade: Uma vez inserido o bloco em um blockchain ele não pode ser alterado ou removido e toda a base de dados está replicada nos diversos nós da rede P2P então é garantida alta auditabilidade na blockchain.
- Imutabilidade e Irrefutabilidade: A estrutura de dados da blockchain que conta com cálculo de hash de informações inseridas e assinaturas com chave privada do autor garantam nas validações a imutabilidade dos dados e a irrefutabilidade de autores de transações.

## 2.4 Mecanismo de Consenso

A inserção de um bloco na blockchain consiste em criar o bloco e distribuí-lo para outros nós da blockchain. Uma vez recebido um bloco, ele é verificado e, se estiver válido, repassado para os demais nós da blockhain até que todos os nós tenham um cópia do bloco. A consistência do livro-razão é alcançada no momento em que os blocos e a ordem de inserção deles é a mesma para todos os nós da rede.[9].

"Como esse processo é feito de forma assíncrona e sem coordenação central, cada nó pode ter uma visão diferente do estado do Blockchain em um dado instante. Entretanto, para

garantir a convergência dessas visões em um espaço de tempo não muito longo, o Blockchain utiliza um mecanismo de consenso" [9].

Esse mecanismo de consenso tem o objetivo de garantir que a blockchain permaneça consistente ao que diz respeito à ordenação de blocos e sobre integridade do seu conteúdo suportado por um conjunto de regras respeitadas e aplicadas por um conjunto de participantes atuando de forma organizada [9]. Ele é posto em prática por vários nós não mutuamente confiáveis e essa alta taxa de validação é o que garante a segurança e confiabilidade da rede.

### 2.4.1 Comunicação e troca de mensagens

Como a blockchain é um sistema distribuído, a troca de mensagens para propagar blocos está associado com mecanismo de distribuição de mensagens em que não há uma coordenação central e cada nó deve transmitir informação para outros nós que se sabe que estão participando do sistema. Geralmente cada nó possui uma lista, não necessariamente completa, de outros nós onde informações são transmitidas para por meio de protocolos de broadcasting[9].

Muitas blockchains, como Bitcoin e Ethereum, adotam a estratégia de existir nós que tem uma lista de todos os participantes da rede. Esses nós, na conexão de um novo nó, escolhem nós aleatórios para que sejam passados para o novo nó e esse novo nó possa fazer o broadcasting [9].

### 2.4.2 Regras de consenso

No mecanismo de consenso deve existir um conjunto de regras que visam permitir que cada nó independente do sistema possa aplicar para validar transações, resolver conflitos e lidar com *falhas bizantinas*. *Falhas Bizantinas* são "falhas que podem apresentar-se de forma diferente para observadores distintos, devido à ausência de uma visão global da rede, criando então inconsistências no sistema" [9].

As regas de consenso podem resultar em dois tipos de consenso diferente: determinístico e probabilístico. No consenso determinístico, uma vez alcançado o consenso, a informação armazenada na blockchain não pode ser alterada posteriormente. No consenso probabilístico, um consenso posterior pode ir contra um consenso feito em um momento de tempo passado. Um exemplo de consenso probabilístico é o do Bitcoin que aplica a regra de armazenar localmente a versão da blockchain que apresente o maior número de blocos durante um consenso mesmo que o consenso já diferente já tenha sido alcançado no final [9].

Na blockchain vários nós estão fazendo a mesma coisa e portanto há um dinamismo intrínseco ao processo de criação e inserção de blocos na blockchain onde todos os nós estão competindo para fazer a montagem de blocos e validação das transações visíveis a

cada nó antes dos concorrentes. Uma vez que um bloco consegue fazer uma validação pelo processo de consenso da blockchain toda a rede precisa ser informada para que o bloco seja armazenado na cadeia de blocos [1].

Ao final da construção e validação de um novo bloco, os algoritmos de consenso pagam uma recompensa ao minerador pelo esforço computacional. Essa recompensa é feita pois "a segurança dos protocolos de consenso em blockchain é baseada na suposição de que a maioria dos operadores dos nós da rede P2P (os mineradores) está mais interessada em se beneficiar dos mecanismos de incentivo do protocolo e menos propensa a quebrar as regras" [1].

### 2.4.3 Algoritmos de inserção de blocos

Por se tratar de um sistema distribuído, a inserção de novos blocos precisa ser orientado por regras bem definidas para evitar que todo e qualquer nó consiga incluir um novo bloco de forma não organizada o que faria com os nós dificilmente chegassem a uma única visão global quanto ao conteúdo e ordem de blocos[9]. Portanto, deve existir um mecanismo de inserção de blocos que aumente a convergência do estado do sistema. A seguir estão alguns dos principais:

- Proof of Work(PoW): Neste processo de inserção todos os nós da rede podem competir para validar transações e inserir novos blocos. Os nós que querem fazer o processo de validação e inserção são chamadas de mineradores e devem realizar uma tarefa primeiro para que ganhem o direito de validar e inserir. A tarefa no processo de PoW é o de "encontrar uma entrada cujo valor de hash seja menor que um determinado alvo" [9]. Fazer a mineração é um processo altamente custoso computacionalmente e portanto uma desvantagem deste método é o tempo que leva para a mineração ser completada e o custo energético associado. Atualmente é o método utilizado pelo Bitcoin e Ethereum.

- Proof of Stake(PoS): No PoS apenas nós que possam oferecer garantias, em termos de tokens, podem participar. Nesse processo, validadores são escolhidos aleatoriamente proporcionalmente às garantias oferecidas para criar blocos e para validar blocos que eles não criaram. A garantia é para que o nó sofra penalidades com perdas financeiras caso haja alguma anomalia na criação ou validação de blocos ou seja percebido algum comportamento malicioso por parte do nó [9]. Deste modo espera-se que as consequências de um mau comportamento sejam mais caros do que agir corretamente. A blockchain de Ethereum está lançando um nova versão que migrará do atual uso de PoW para PoS [7]. Por não ter a mineração feita no PoW espera-se ganhar em termos de economia energética e tempo com a atualização.

- Proof of Authority(PoA): No PoA a blockchain possui alguns nós que possuem autoridade para fazer a validação de novos blocos. A autoridade que esses poucos nós confiáveis possuem são pré-configuradas pela administração da rede. "Para prevenir a personificação desses nós especiais por outros nós, as autoridades recebem certificados digitais válidos para assinar os novos blocos" [9]. Este mecanismo de consenso geralmente é usado por blockchains privadas ou redes de testes de blockchains públicas como Goerli Ethereum[10].

## 2.5 Modelos Blockchain

De acordo com [9], as blockchains de hoje em dia podem ser classificadas em respeito ao seu acesso em:

### 2.5.1 Pública

Modelo que pode ser considerado completamente descentralizada. Qualquer nó pode participar do processo de consenso distribuído, ler e enviar transações.

### 2.5.2 Privada

Modelo completamente centralizado no qual uma instituição possui o total controle sobre as operações.

### 2.5.3 Consórcio

Modelo híbrido parcialmente centralizado em que duas ou mais instituições possuem o controle sobre as operações.

## 2.6 Fluxo de inserção de blocos

De acordo com [1] o fluxo de inserção de blocos em uma blockchain pode ser resumido nos seguintes passos:

1. Alice e Bob criam uma conta cada na blockchain.
2. Alice divulga o endereço da sua conta para Bob.
3. Bob cria uma transação com a sua assinatura digital e envia para o endereço de Alice;
4. Bob faz transação entre os nós da rede P2P via um nó da rede.

5. Um nó qualquer inclui a transação em um bloco e com a propagação do bloco os nós da rede trabalham para obter um consenso sobre a criação do bloco de acordo com o mecanismo de consenso empregado da blockchain.

6. Mediante validação das transações, os nós da rede P2P propagam o seu resultado para outros nós.

7. Ao consultar o livro razão Alice vê a sua transação conclui que a sua transação foi processada.

Os passos descritos acima podem ser visualizados na figura 2.3. Um ponto importante de frisar aqui é que do momento que a transação é submetida para validação na blockchain até o momento que ela pode ser consultada há o delay do funcionamento do mecanismo de consenso e propagação de blocos por toda a rede P2P [9]. Assim, no momento que estiver sendo criada uma aplicação em cima de uma blockchain esse delay deve ser levado em conta já que é intrínseco aos mecanismos de consenso utilizados hoje em dia.

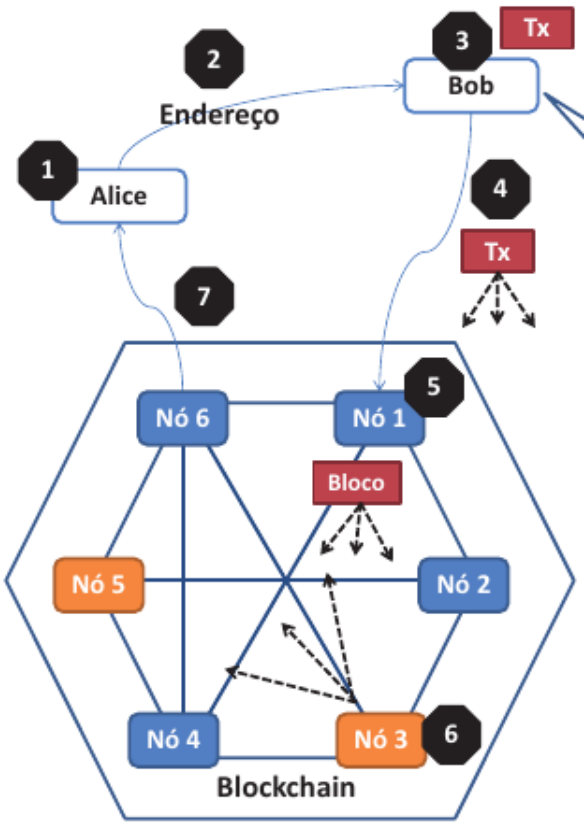


Figura 2.3: Fluxo de transação na blockchain. Fonte: [1]

## 2.7 Contratos Inteligentes

Como já dito, as transações de uma blockchain são texto livre e seu conteúdo depende da semântica da aplicação. A partir dessa liberdade no uso de blockchain foi criado o conceito de contratos inteligentes. Contratos inteligentes são "programas de computador seguros e imparáveis que representam acordos executáveis e exigíveis automaticamente"[1]. Estes programas são implantados como transações na blockchain. Eles rodam nos nós da rede de blockchain e "funcionam automaticamente em benefício dos participantes da rede"[1].

Quando implantado, o programa do contrato para a blockchain, ele não pode mais ser impedido de rodar uma vez que a condições programadas sejam atingidas e portanto o que foi programado é o que será executado na rede P2P. Essa característica de inviolabilidade de código é interessante para execução de acordos em que não existem necessariamente confiança entre duas ou mais parte envolvidas [1]. Com o uso de contratos inteligente, uma vez feito um acordo e implantado na blockchain ele será executado nas condições previstas sem qualquer entidade confiável controlando o processo.

## 2.8 Aplicativos Descentralizados

Uma vez que foi explicado como é o funcionamento de uma blockchain e que as transações podem possuir um aspecto geral em que contratos inteligentes podem ser utilizados de diversas formas, uma blockchain pode ser usada como infraestrutura para a criação de *Decentralized Applications* ou DApps [1]. Os DApps possuem todas as propriedades discutidas pois os dados transacionais dos usuários estarão armazenados na blockchain e para as atividades executadas no aplicativo deverão ser submetidas novas transações para o mecanismo de consenso da blockchain. Exemplos de DApps são produtos de transações financeiras, cartórios, eleições, doações e vários outros onde a confiabilidade, imutabilidade, autenticidade, e integridade podem ser úteis.

Um DApp pode ser construído com uma arquitetura pensando nos seguintes componentes [1]:

1. Front-end: camada que os usuários usarão para interagir com a aplicação. Normalmente é um aplicativo para smartphone e/ou um website que mandarão requisições para um back-end.

2. Aplicação servidora: back-end cujo o front-end trocará mensagens com. É a camada responsável por conter regras de negócios do produto e poderá ter um banco de dados próprio além de ser responsável de criar transações para a blockchain.

3. Integração aplicação com blockchain: interface de programação de aplicações (Application programming Interface - API) responsável por servir de ponte entre o back-end da aplicação servidores com a blockchain.

4. Aplicação Blockchain: Aplicação responsável por manipular o livro razão através de uma API.

5. Contratos inteligente: Contrato que está presente em cada um dos nós da rede P2P e é responsável por implementar transações de semântica complexa na plataforma blockchain.

A figura 2.4 mostra os componentes descritos acima na arquitetura de uma aplicação blockchain.

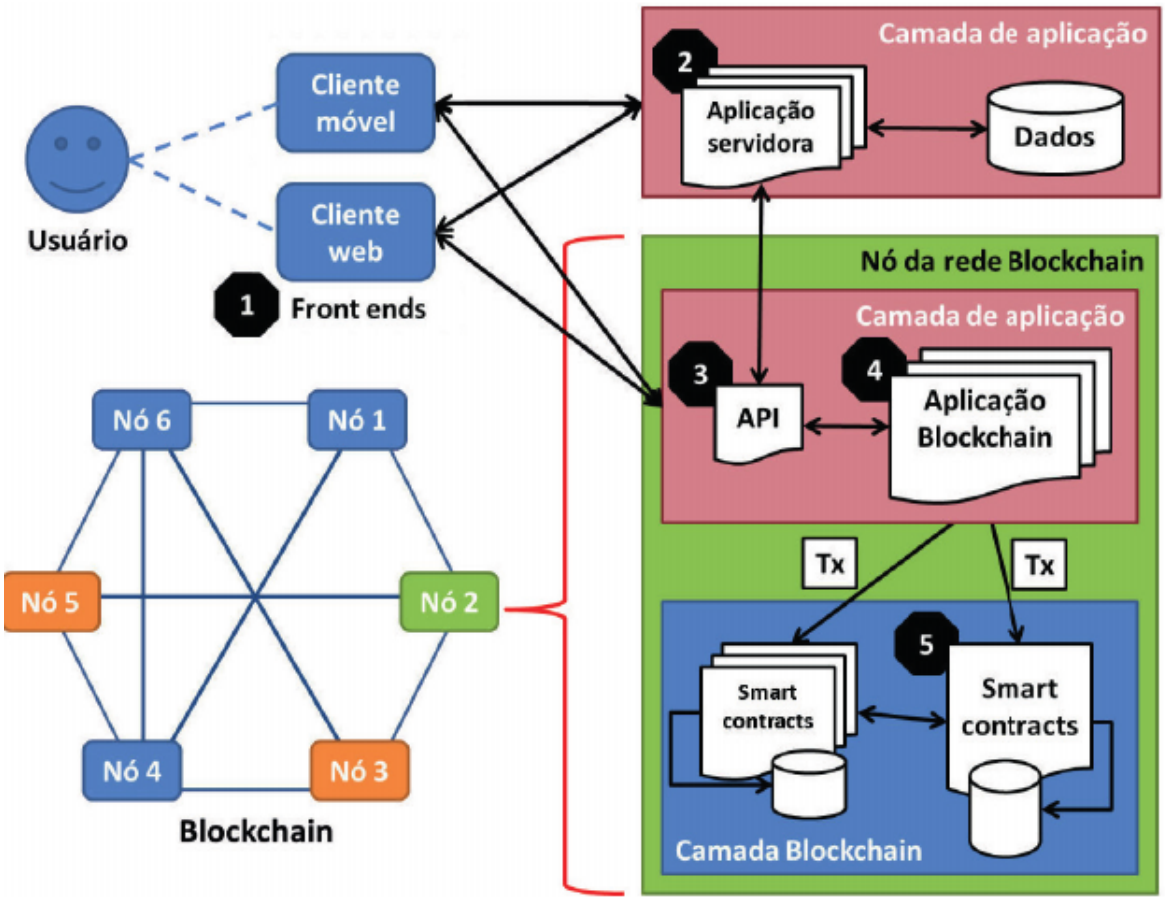


Figura 2.4: Estrutura de uma aplicação blockchain. Fonte: [1]

# CAPÍTULO 3

# Aplicações Financeiras

## 3.1 Moedas Digitais

O conceito de blockchain foi introduzido na apresentação de uma aplicação específica que foi a criação da primeira moeda digital descentralizada em 2008, o Bitcoin [3]. Como imaginado, esta moeda digital não tem a entidade de um banco central controlando ela podendo alterar o seu valor a partir da impressão de mais moedas. Hoje o único jeito de se criar um bitcoin é através do processo de mineração do mecanismo de consenso PoW sendo que nesse processo foi previsto no código do bitcoin a limitação da quantidade de moedas produzidas ao longo do tempo e está previsto o número máximo de bitcoins que poderão ser gerados [9].

Desde a criação do bitcoin muitas outras moedas digitais surgiram e muitas delas possuem valor financeiro real o que permite que as moedas digitais possam ser usadas como meios de pagamento.

## 3.2 Ethereum

"O Ethereum é um Blockchain de propósito geral que suporta a execução de contratos inteligentes por meio de uma máquina virtual chamada Ethereum Virtual Machine (EVM). A EVM é uma camada de abstração, executando no host que conecta o cliente com a rede e oferece o ambiente que executa as instruções dos contratos inteligentes de terceiros em uma rede global de computadores." [9].

A Ethereum é uma blockchain pública e portanto qualquer pessoa pode criar e ler transa-

ções e participar do processo de consenso PoW na versão antiga e PoS na versão nova [7]. Por ser pública também tem a entrada e saída aleatória de nós.

O contrato da Ethereum possui uma linguagem própria de escrita que é o Solidity. Cada operação realizada por um contrato está associado a um custo de execução de *gas* que é o "combustível"consumido na operação[9].

O aspecto de propósito geral da linguagem solidity empregada nos contratos inteligentes faz com que da Ethereum possam nascer aplicações diversas, não necessariamente financeiras, que utilizem a infraestrutura da blockchain já existente sem a necessidade de criar uma blockchain nova. Ethereum é uma das plataformas mais utilizadas em projetos pilotos atualmente e também uma das tecnologias blockchain mais estudadas com diversas ferramentas de suporte ao desenvolvimento disponíveis.

A blockchain Ethereum possui redes de testes que permitem que o desenvolvimento e testes de aplicações em cima dela possam ser feitos sem qualquer custo financeiro real. Exemplo de rede de teste é a Goerli [10] que possui um *faceut* onde usuários podem gerar Ether para serem usados em suas contas sem custo. Diferentemente da mainnet, Goerli usa o consenso *Proof of Authority*.

Além disso, há soluções no mercado, como o infura [11], que disponibilizam um nó remoto conectado na rede Ethereum e que permite que desenvolvedores submetam transações e consultas para a rede Ethereum sem a necessidade de provisionar infraestrutura para um nó em um servidor local da Ethereum para cada uma das redes, mainnet e de testes.

### 3.2.1 Gas

A blockchain Ethereum possui sua própria forma para financiar a validação de blocos e monetizar a mesma. Na blockchain existe uma unidade de medida de esforço computacional chamada gas. O gas busca quantificar o esforço computacional (ciclos de CPU, acesso a disco e acesso de memória) que um nó da Ethereum precisa fazer para processar uma transação em um contrato inteligente na EVM [12].

Naturalmente, aplicações diferentes possuem contratos diferentes que por sua vez tem complexidades diferentes com um gas associado. Uma vez sabido o gas de uma transação, o nó que está enviando a transação para ser inserida na blockchain define um preço de Ether pelo qual deseja pagar por cada gas. Deste modo, a transação terá um custo total para o requisitante de gas da transação multiplicado por preço do gas [12]. O custo de gas será enviado para o nó que executar a transação e inserir ela na blockchain como uma forma de recompensa pelo trabalho executado.

Uma vez que cada nó possui um número finito de poder computacional, eles precisam decidir quais transações executar na EVM. A decisão mais coerente é de pegar as transações que irão devolver uma remuneração maior devido ao seu custo mais alto. Desse modo, uma

estratégia para que nós tenham as suas transações processadas com prioridade é aumentar o preço do gas da transação submetida à rede Ethereum. Essa definição de preço de gas definida considerando o quão prioritária deverá ser o processamento da transação vai oscilar de acordo com a oferta de recurso computacional e a demanda de novas transações para serem processadas configurando um "mercado de processamento de transações". Os que querem ter suas transações processadas mais rápido devem pagar mais no preço do gas [12].

### 3.2.2 Tokens

O *Ether* é a moeda nativa do Ethereum que possui valor monetário. Os contratos inteligentes processados na EVM permitem que desenvolvedores de DApps consigam criar seus próprios tokens, que funcionam como uma moeda para determinadas aplicações. Assim, usuários conseguem comprar tokens de DApps gastando seus *ether*, bem como vendê-los [13]. Os tokens podem ser usados pelos DApps como moeda de troca por serviços dos produtos, como uma espécie de mensuração de direitos que uma pessoa tem para fazer determinada atividade em um DApp [14] ou virtualização de qualquer objeto do mundo físico [7].

Devido a possibilidade de existir diversos tokens na Ethereum é de interesse que seja criado um padrão para todos os tokens. A padronização dos tokens traz vantagens como a garantia que um token criado seguindo o padrão terá determinadas funções que são úteis e importantes mas que num primeiro momento poderiam passar despercebidas. O padrão mais utilizado de token é o ERC-20 [7]. O padrão ERC-20 foi proposto por Fabian Vogelsteller em 2015. Tokens seguindo esse padrão devem implementar alguns métodos em seus contratos que estão definidos na interface do ERC-20. Dentre as funções, há métodos de transferência de tokens entre contas e também métodos para verificar o balanço de um usuário [7].

## 3.3 Micropagamentos

Micropagamentos são transações financeiras cujo valor financeiro é muito pequeno e que geralmente são feitas de maneira online [5]. Um grande empecilho para a adoção de micropagamentos é que muitas vezes as taxas das transações financeiras cobradas por entidades que intermedeiam o pagamento excedem o valor que se quer transferir. Desse modo, por não existirem essas entidades nas transferências de moedas digitais elas são um grande candidato para servir como meio de pagamento para micropagamentos

## 3.4 Limitações em pagamentos

Apesar de moedas digitais em cima de blockchain não terem entidades de confiança intermediadoras entre transferências, pagamentos com moedas digitais ainda possuem certas limitações [5].

1. Escalabilidade - A blockchain do bitcoin consegue lidar com cerca de 7 transações por segundo e a blockchain Ethereum com Ether consegue lidar com cerca de 15 transações por segundo. Números bem abaixo do que a Visa consegue fazer hoje uma média de 1700 transações por segundo [5–7].

2. Taxa de transações - apesar de não ter entidades intermediadoras como bancos, a submissão de uma transação para a Ethereum, por exemplo, tem um custo associado para recompensar os mineradores impactando no custo final do pagamento.

3. Tempo de confirmação - como já discutido, a submissão de uma transação para inserção na blockchain tem o delay do processo consenso e distribuição do novo bloco pela rede. Este tempo de espera faz com que um pagamento seja um processo demorado quando, idealmente, deveria ser quase instantâneo.

Apesar de blockchains terem permitido a existência de um ativo digital com valor financeiro as limitações acima dificultam o uso de moedas digitais como soluções de pagamentos.

## 3.5 Payment Channel Network (PCN)

PCN é uma estratégia de segunda camada para realizar um alto número de transações fora da blockchain (off-chain) até que em algum momento um número menor ou único de transações sejam executadas na blockchain (on-chain) passando pelo processo de consenso[5]. No método PCN contratos inteligentes são utilizados como repositórios confiáveis de tokens por duas contas quaisquer na blockchain, representando o canal aberto entre $A$ e $B$. A transferência off-chain ocorre quando uma conta $A$ envia um comprovante de transferência assinado por sua chave privada para uma conta $B$ onde o comprovante assinado por $A$ serve como um cheque que $B$ pode usar para "sacar"o valor depositado por $A$ no contrato inteligente com uma transação on-chain [8].

O envio do comprovante assinado ocorre por troca de mensagens diretamente entre as duas partes sem nenhum processo de consenso na blockchain, o que faz com que as transferências feitas off-chain não tenham o problema de latência que transferências on-chain possuem. Além disso, $A$ pode fazer quantas transferências desejar para $B$ limitado ao saldo presente no contrato inteligente na blockchain. Esse alto número de transferências possíveis sem a necessidade de submissão on-chain faz com que as contas envolvidas não

fiquem tão expostas à variação das taxas cobradas on-chain. Além de que, em teoria, faz com que haja um número menor de transferência on-chain fazendo com que os recursos computacionais da blockchain não sejam usados em demasia, controlando mais as taxas para submissão de transferências e o problema de escalabilidade.

Transferências usando PCNs podem ser feitos, com o acordo das partes, sem taxas off-chain tendo como as únicas taxas existentes as cobradas pelas transações on-chain na abertura do canal entre as duas contas e no fechamento do canal [8]. Tais características amenizam as desvantagens de submissão de pagamentos on-chain. A abertura de um canal tem uma transação on-chain envolvida e portanto tem um custo envolvido bem como o tempo de validação pelo processo de consenso. Da mesma forma, no "saque"de tokens ocorre a modificação do estado de um contrato e também tem um custo transacional envolvido e tempo de confirmação do processo de consenso [8].

Apesar do processo de abertura de canal entre um remetente $A$ e um destinatário $B$ ter um custo associado, não é necessária a abertura de um canal com toda e qualquer conta que se deseja fazer um pagamento devido a possibilidade do uso de uma rede de canais de pagamento que é criada entre as contas. Se uma conta $A$ quer realizar um pagamento para uma conta $C$ e ambas possuem um canal aberto com $B$ cujo valor depositado nos canais seja menor do que o valor do pagamento então é possível realizar o pagamento entre $A$ e $C$ com intermédio de $B$ off-chain, caracterizando um pagamento mediado. Um pagamento pode ser realizado por meio de quantos canais necessário desde que todos possuam um valor maior do que o valor do pagamento [8].

## 3.6 Raiden

Raiden Network é uma solução off-chain que implementa sobre a blockchain Ethereum o conceito de PCN. A implementação de uma PCN é complexa e não trivial o que dificulta o processo de desenvolvimento de aplicações de pagamento com moedas digitais. Apesar disso, toda a complexidade de implementação de uma PCN é abstraída através de uma API do Raiden de fácil uso, o que permite o desenvolvimento de soluções descentralizadas e escaláveis de maneira mais fácil. Raiden Network é compatível com o pagamento de qualquer token ERC20 [8].

Como dito anteriormente, uma PCN se baseia no conceito de contratos inteligente e na capacidade deles guardarem tokens on-chain, fazendo com que toda operação off-chain seja resguardada por uma operação on-chain. Uma conta $A$ pode realizar pagamentos para uma conta $B$ enviando-o um balance-proof, análogo a um cheque assinado por $A$. De posse do balance-proof assinado, $B$ pode enviá-lo para o contrato inteligente e realizar o "saque"dos tokens on-chain. Como o pagamento off-chain está apoiado no saldo que está depositado no

contrato inteligente on-chain, uma transferência off-chain nunca pode ter o valor maior do que esse saldo.

### 3.6.1 Contratos inteligentes

Para realizar essa dinâmica, o framework Raiden faz o uso de 3 tipos de contratos inteligentes [2]. O primeiro tipo é o *TokenNetwork* que é o canal que armazena tokens entre duas contas ethereum diferentes para um determinado token ERC20 e é o ponto de contato principal para operações on-chain. O segundo tipo de contrato inteligente é o *TokenNetworkRegistry* que registra todos os contratos inteligentes do tipo *TokenNetwork* e pode ser usado para procurar contratos inteligentes de um token específico entre 2 participantes. E por fim, o terceiro tipo de contrato inteligente é o *SecretRegistry* que armazena os segredos revelados e em qual bloco foi revelado no protocolo Raiden permitindo que nós intermediários de um pagamento possam realizar o "saque"de tokens em situação de erro como será visto a seguir.

### 3.6.2 Privacidade

Um importante ponto de atenção, é entender como se dá a privacidade no uso de PCNs por meio do Raiden. O uso de contratos inteligentes na Ethereum faz com que as informações contidas no contrato fiquem pública por todas a blockchain, logo o seu uso implica nas seguintes limitações [2]:

- A quantidade de tokens depositados em um contrato inteligente, bem como os dois endereços de conta, o token e o saldo final de cada participante são informações públicas na blockchain.
- Os contratos inteligentes só sabem do saldo final de cada participantes, não de todas as transações que ocorreram off-chain.

### 3.6.3 Transferências Mediadas

De forma a não se limitar no pagamento entre nós com canal aberto entre si, Raiden foi construído para permitir transferências utilizando canais intermediários entre nós. Neste caso, a transferência denomina-se transferência mediada [2] e o conjunto de todos os canais e nós passam a constituir o Raiden Network. Todos os protocolos foram pensados e construídos pensando na realização de transferências mediadas onde uma transferência entre dois nós que possuem um canal aberto entre si é apenas um caso especial dos protocolos que preveem mediadores entre o iniciador do pagamento e o alvo.

Uma transferência mediada é composta por um inicializador, um alvo, um certo número $N$ de mediadores e, no melhor cenário, $10N + 14$ mensagens. Em uma rede em que há 3 nós

*A*, *B* e *C*, onde *A* tem um canal aberto com *B* e *B* tem um canal aberto com *C*, o nó *A* deseja fazer uma transferência de *n* tokens para *C*. Para isso, o protocolo Raiden com a seguinte troca de mensagens tendo o nó *A* como inicializador, o nó *B* como mediador e o nó *C* como alvo é iniciado:

1. o nó *A* cria uma mensagem *locked transfer* que é composta, principalmente, pela quantidade de tokens *n*, pelo hash de um segredo, por um número de bloco até o qual o pagamento pode ser finalizado que serve como um tempo limite até o qual o pagamento pode ser finalizado, endereço da conta destino, endereço da conta origem, endereço do token e outros. O pagamento só poderá ser dado como realizado pelo nó *A* quando o segredo for enviado para o nó destino e portanto, até então, foi criado um pagamento bloqueado. A mensagem é assinada com uma chave privada para que outros nós possam verificar a autenticidade e integridade da mensagem e então enviada para o nó *B*.

2. O nó *B* verifica que ele não é o alvo da mensagem e encaminha a *locked transfer* para o nó *C*. Antes de encaminhar a mensagem, o nó *B* precisa garantir que o canal que ele tem aberto com o nó *C* tem a quantidade de tokens necessários para fazer a transferência pois, apesar que irá receber *n* tokens do nó *A* no canal *A*-*B*, o canal *B*-*C* precisa ter esses *n* tokens para serem enviados a *C*. Caso não haja a quantidade de tokens disponíveis é retornado um erro a *A* com esta causa.

3. Recebido o *locked transfer* no nó C, ele identifica que é o alvo final do pagamento, verifica que o pagamento ainda não venceu, identifica quem é o remetente do pagamento e sua assinatura e envia uma mensagem do tipo *secret request* para o remetente. A mensagem *secret request* contém principalmente um identificador do pagamento e o hash do secret recebido na *locked transfer* para que o remetente saiba qual segredo deve ser enviado, a assinatura do nó *C* para autenticar que é o alvo do pagamento quem está enviando a mensagem para o nó *A* e outros campos.

4. O nó *A* recebe a mensagem *secret request*, verifica a qual pagamento a mensagem se refere, a identidade de quem enviou a mensagem através da assinatura e então envia uma mensagem do tipo *reveal secret* para o nó *C* diretamente. A mensagem *reveal secret* é composta principalmente pelo segredo da *locked transfer* e pela assinatura do nó *A*. Assim que enviada a mensagem *reveal secret* e confirmado o recebimento pelo nó *C*, o nó *A* pode assumir que o pagamento foi realizado.

5. O nó *C* recebe a mensagem *reveal secret* contendo o segredo do *locked transfer* e com isso o nó *C* já consegue reivindicar os tokens do nó *B* para si. Como os *n* tokens

chegaram via nó *B*, o nó *C* envia o *reveal secret* para o nó *B* reivindicar os tokens do nó *A*.

6. o Nó *B* recebe o *reveal secret* e devolve uma mensagem de *unlock* para *C*. Esta mensagem *unlock* contém o balance proof, estrutura que determina o estado do nó *B*, em termos de pagamentos pendentes no canal *B*-*C*, e busca dizer ao nó *C* que o *locked transfer* que estava com o status pendente em sua árvore de merkle foi removido, o total transferido de *B* para *C* foi atualizado e que foi alterado a locksroot da árvore de merkle.

7. em seguida ocorre uma retro propagação da troca de mensagens *reveal secret* e *unlock* entre os mediadores até o nó origem do pagamento, neste caso o nó *A*. Quando o nó *A* envia a mensagem de *unlock* para o nó *B* a transferência off-chain está completa.

A figura 3.1 mostra um diagrama de sequência disponibilizado pela Raiden em [2] com um fluxo de pagamento sem erros ou problemas, como o processo descrito acima. Além das mensagens descritas acima, podem ser vistas no diagrama a presença de mensagens *delivered* e mensagens *processed* que fazem parte do protocolo Raiden. A mensagem *delivered* funciona como uma mensagem de confirmação de recebimento de qualquer outra mensagem e pode-se notar que durante todo o protocolo está presente. A mensagem *processed* tem a função de informar que, após recebida uma mensagem qualquer, a mensagem teve o seu tipo identificado, campos validados e que a próxima ação está definida.

Um cenário que pode ocorrer é o de um nó cair enquanto há uma transferência pendente. Na transferência descrita, caso o nó *A* tivesse caído antes de enviar o segredo via *secret reveal* para o nó *B* o pagamento não teria sido realizado pois ninguém teria conseguido desbloquear a *locked transfer* antes da sua validade terminar.

Outro caso é em uma transferência com pelo menos 2 mediadores. No caso de *A* querer enviar uma transferência através de *B* e *C* para *D* e o nó *C* cair após *A* ter revelado o segredo para *D*. Nessa situação o nó *D* não receberá uma resposta após enviar o *reveal secret* interpretando a falta de resposta como se o nó *C* tivesse caído. Para que o nó *B* e o nó *C* consigam realizar a reivindicação dos pagamentos posteriormente, o nó *D* faz o uso de uma operação on-chain para depositar o segredo no contrato inteligente *SecretRegistry*. Todos os nós conseguem enxergar o *SecretRegistry* e portanto, agora, todos os nós tem acesso ao segredo. De posse do segredo recuperado pelo *Secret Registry*, os nós *B* e *C* conseguem atualizar os seus pagamentos pendentes e gerar a mensagem *unlock*.

### 3.6.4 Múltiplos pagamentos pendentes

Um nó *A* pode querer fazer pagamentos para nós diferentes usando um nó *B* como mediador de todos. Para isso o nó *B* tem que armazenar todos os pagamentos pendentes

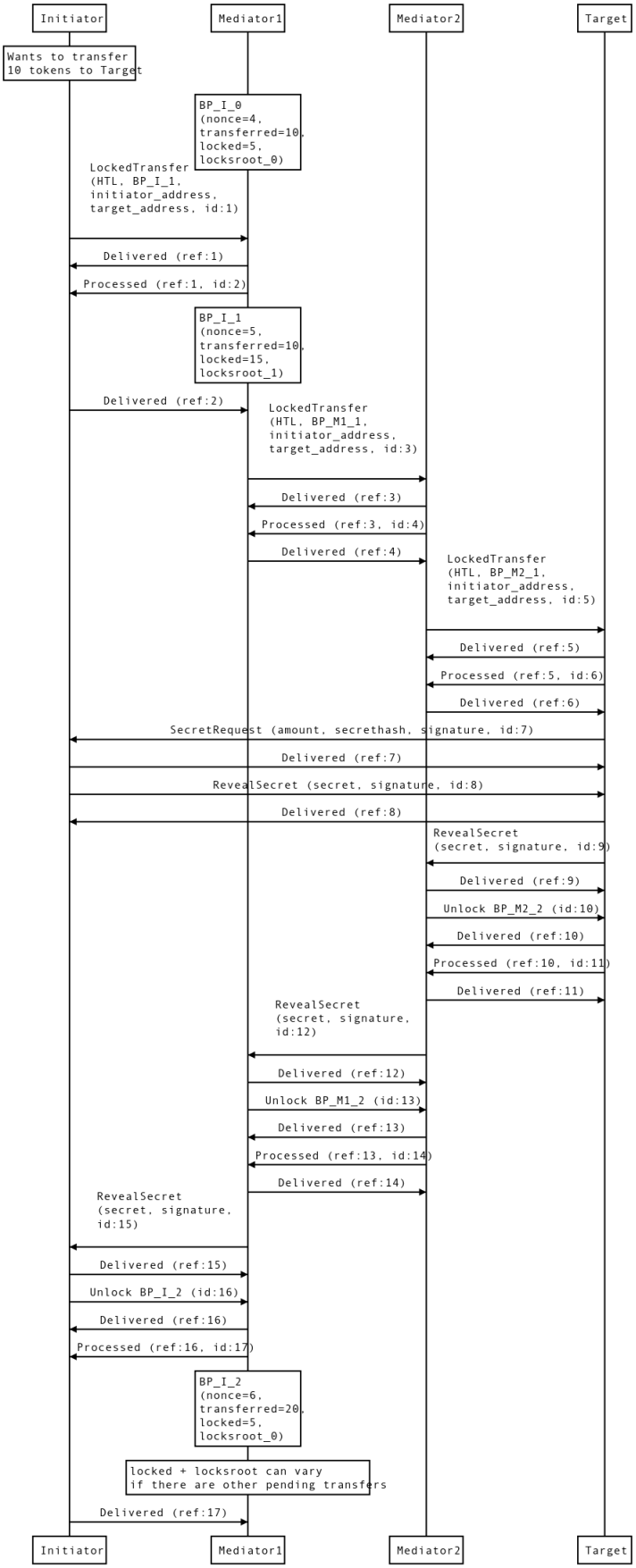


Figura 3.1: Diagrama de sequência de troca de mensagens off-chain em um pagamento com fluxo de sucesso. Fonte: [2]

vindos de *A* até que o protocolo de pagamento seja finalizado. Esse armazenamento de pagamentos pendentes é feito pelo Raiden em uma estrutura chamada Árvore de Merkle, a mesma usada nos blocos da blockhain. Cada nó possui uma Árvore de Merkle por canal por direção.

Um nó que já fez duas transferências por um canal cujos pagamentos estão pendentes, grava em sua Árvore de Merkle esses pagamentos. Outro nó mediador que repassou esses pagamentos faz o mesmo registro na sua Árvore de Merkle. Quando um nó gera um *pending transfer* uma das informações colocadas na mensagem é exatamente a raiz da Árvore de Merkle do canal que será usado para a transferência. Como a cada pagamento pendente colocado na árvore de Merkle o valor da raiz da árvore, denominada *locksroot*, muda, cada novo pagamento terá uma *locksroot* diferente do pagamento anterior, possibilitando que a *locksroot* possa ser usada para simbolizar o estado de um nó com relação aos pagamentos pendentes por um canal. Ao receber um *pending transfer*, o nó compara a *locksroot* vinda na mensagem com a *locksroot* da sua própria Árvore de Merkle. Ao ver que as *locksroot* possuem o mesmo valor então o *pending transfer* é aceito e a Árvore atualizada com um novo pagamento pendente.

### 3.6.5 Transporte de mensagens

Raiden foi construído de forma que os protocolos não dependessem da forma como as mensagens são trafegadas na rede, tendo o único requisito a comunicação com um nó da Ethereum [2]. Raiden usa o protocolo de mensagens peer-to-peer Matrix baseado em federação de servidores que suporta multicast em salas de conversa onde conversas diretas entre nós são modeladas como 2 participantes em uma sala. [2, 15]. Apesar de Matrix ter um sistema de identidade próprio Raiden não faz o uso dele para evitar um possível ponto único de falha. Assim, a identificação e autenticação de usuários são baseadas nos dados e chave privada da conta Ethereum.

### 3.6.6 Trabalhos Relacionados

Raiden surge como uma das possibilidades de pagamentos para uma variedade de aplicações como por exemplo: pagamentos em varejo, infraestrutura de pagamentos em uma economia de máquinas Iot, jogos e etc.

Em [16] é possível verificar exemplos de aplicações construídas usando Raiden como infraestrutura de pagamentos. Dentre os exemplos há a implementação de um sistema de compartilhamento de Wifi cujo consumo de pacotes de 100 Kb é permitidos diante de pagamentos usando Raiden, um jogo de pedra, papel e tesoura e sistema de *live streaming* que cobra pagamento conforme o uso.

# CAPÍTULO 4

# Sistema Proposto

## 4.1 Protótipo

Neste trabalho de graduação, foi construída uma loja virtual usando como meio de pagamento a rede Ethereum. A loja tem o objetivo de mostrar que moedas digitais podem ser utilizadas para fazer transações seguras, de forma descentralizada e rápidas por meio do conceito de PCN implementada pelo Raiden.

A aplicação construída libera o consumo de conteúdos como PDFs, vídeos ou músicas sendo que a disponibilização do conteúdo ocorrerá mediante um pagamento de pequeno valor, configurando um micropagamento.

Como moeda de troca foi escolhido um token da rede de testes Ethereum *Goerli* denominado WIZ cujo código é "0x95B2d84De40a0121061b105E6B54016a49621B44"e que por meio da rede de testes podem ser gerados tokens como massa de testes para a realização de pagamentos sem custo real. O contrato do token pode ser verificado em [17]

## 4.2 Arquitetura

O trabalho foi estruturado utilizando 5 componentes principais:

- Front-end: interface pelo qual o cliente irá interagir com a plataforma e pelo qual haverá o consumo do conteúdo e compra.
- API de Usuários: API que terá como domínio os dados do cliente e contas da Ethereum.

- API de Conteúdos: API que terá como domínio os diversos conteúdos que serão vendidos na loja e que será responsável de verificar se o conteúdo foi comprado ou não para liberação de download;

- API de Compras: API que terá como domínio o vínculo de conteúdos com os usuários que compraram tal conteúdo.

- Robô de monitoração de pagamentos: rotina responsável por verificar transferências que são feitas para a loja e dar baixa em compras feitas por clientes para liberação de conteúdos.

- API Raiden: API disponibilizada pelo framework do Raiden que implementa o PCN e que será utilizada pelos demais componentes para criar transações na Ethereum. API do Raiden irá utilizar um cliente remoto da Ethereum na plataforma Infura para comunicação com a blockchain. Serão rodadas uma instância do Raiden para cada conta, portanto a loja deverá ter a sua instância para recepção dos pagamentos e o cliente deverá ter a instância dele para a realização da transferência.

A figura 4.1 mostra um diagrama com todos os componentes conectadas.

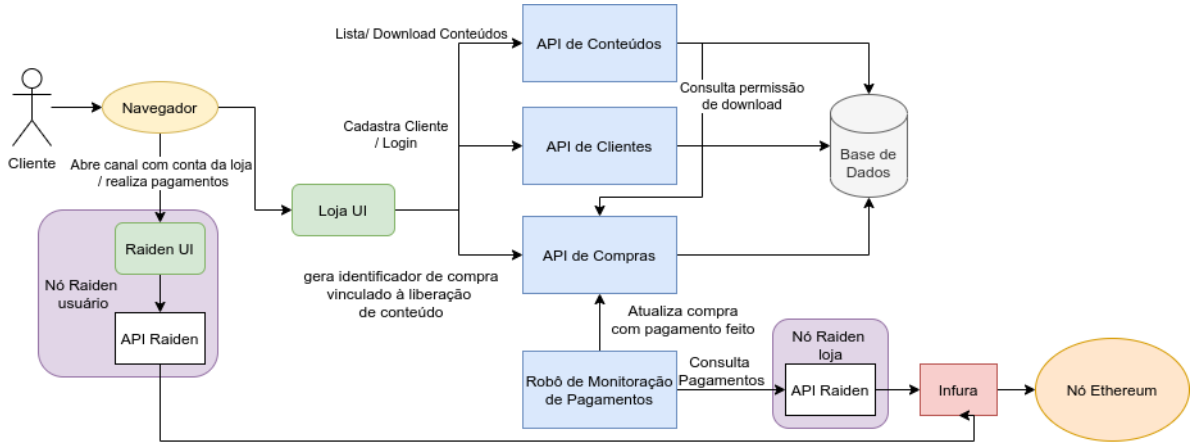


Figura 4.1: Diagrama de componentes.

## 4.3 Tecnologias

As APIs que serão construídas serão escritas utilizando a linguagem Java com suporte do framework Spring para comunicação HTTP e comunicação com o banco de dados.

O Banco de dados escolhido é o Postgres por não ter custo de licença de uso.

O Front-end será construído utilizando o framework Angular com uso da linguagem Typescript.

## 4.4 Modelagem Lógica Banco de dados

Na figura 4.2 é possível ver um diagrama de como foi implementada a base de dados. Há uma tabela de domínio para cada API onde há uma relação de muitos para muitos entre usuários e conteúdos que se traduz na tabela de compras. É possível ver que na tabela de compras há um controle de vencimento para que haja um limite para que o pagamento seja realizado. Além disso o controle de se o pagamento foi feito ou não é realizado pelo preenchimento de uma informação de data e hora de pagamento. No caso dessa informação estar vazia indica que o pagamento ainda não foi feito. Conforme dito anteriormente, foi adotado que os valores monetários correspondem a unidade WIZ do token "0x95B2d84De40a0121061b105E6B54016a49621B44".

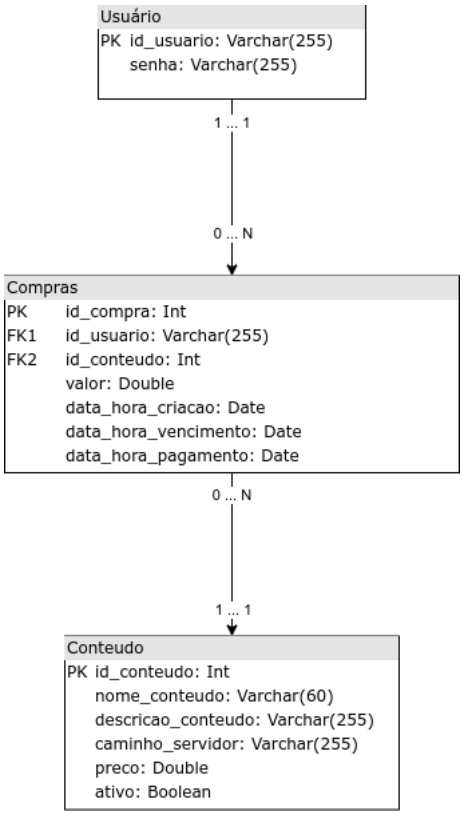


Figura 4.2: Modelagem Lógica Banco de Dados.

## 4.5 Contrato de APIs

Abaixo serão mostrados os principais contratos dos JSONs de entrada e saída de cada uma das rotas implementadas nas APIs com o padrão REST e que são usadas no projeto.

### 4.5.1 API de Usuários

API responsável por guardar e liberar informações de usuários cadastrados na plataforma. Essa API possui 3 endpoints sendo 2 POSTs e 1 GET cujos contratos podem ser verificados no apêndice A. O primeiro endpoint é o POST usuário usado para gravar no banco de dados login e senha novos. O segundo endpoint é o GET usuário que tem como entrada um login e caso esse login seja válido retorna um HTTP 200. Este endpoint é usado pela API de compras para validar que o usuário que está fazendo a requisição de compra é um usuário válido. Por fim, há o endpoint POST login utilizado para validar o login e senha fornecidos permitindo uso da plataforma.

### 4.5.2 API de Conteúdos

API responsável por criar e listar conteúdos do banco de dados e também de realizar o download de binário de conteúdos que estão no repositório de conteúdos cujos contratos podem ser verificados no apêndice B. Esta API possui 3 endpoints. O primeiro endpoint é de download do binário do conteúdo que deve fazer as devidas validações de permissionamento e buscar o binário no repositório de arquivos. O segundo endpoint é o usado para listar os conteúdos disponíveis para compra e download. Por fim, o terceiro endpoint é um usado pelo administrador da loja para gravar novos conteúdos no banco e dados e vinculá-los a um arquivo no repositório e arquivos.

### 4.5.3 API de Compras

API responsável por gerenciar informações de compras de usuários sobre determinados conteúdos. Esta API possui 4 endpoints sendo 2 GETs, 1 POST e 1 PUT.

O primeiro endpoint é o GET Compras de um usuário. Este endpoint é usado pela API de Conteúdos saber quais são e como estão a situação de compras de um usuário e assim colocar a devida visibilidade para cada um dos conteúdos que serão listados na tela.

- GET /store/v1/users/{user_id}/purchase

Lista 4.1: Saída GET lista compras de um usuário

```
1 [
2    {
3       "id_compra": 1,
4       "user_id": "vleite",
5       "content_id": 1,
6       "value": 0.005,
7       "data_hora_compra": "2021-08-01T01:19:30.954296",
```

```
        "data_hora_vencimento_pagamento": "2021-08-03T01:19:30.95
            ↪ 4296",
        "data_hora_pagamento": "2021-08-01T00:21:10.396887",
        "status_pagamento": "PAYMENT_MADE"
    },
    {
        "id_compra": 2,
        "user_id": "vleite",
        "content_id": 2,
        "value": 0.005,
        "data_hora_compra": "2021-08-01T01:19:40.392124",
        "data_hora_vencimento_pagamento": "2021-08-03T01:19:40.39
            ↪ 2124",
        "data_hora_pagamento": null,
        "status_pagamento": "PAYMENT_NOT_MADE"
    }
]
```

O segundo endpoint é o de listar todos as compras filtrando por estado. Este endpoint é utilizado pelo robô de pagamentos identificar quais as compras que tiveram os pagamentos recebidos e assim atualizar o estados delas posteriormente.

- GET /store/v1/purchase?status=PENDING_PAYMENT

Lista 4.2: Saída GET lista compras em aberto

```
[
    {
        "id_compra": 2,
        "user_id": "vleite",
        "content_id": 2,
        "value": 0.005,
        "data_hora_compra": "2021-08-01T01:19:40.392124",
        "data_hora_vencimento_pagamento": "2021-08-03T01:19:40.39
            ↪ 2124",
        "data_hora_pagamento": null,
        "status_pagamento": "PENDING_PAYMENT"
    }
]
```

O terceiro endpoint é o POST conteúdo usado para criar uma compra. Nele devem ser

informados o usuário ao qual a compra deve ser atrelada, o conteúdo que se está comprando e o valor que deverá ser pago. Este endpoint é chamado diretamente pelo front-end quando o usuário inicia o fluxo de compra.

- POST /store/v1/users/{user_id}/purchase

Lista 4.3: Entrada POST compra

```
{
   "content_id" :1,
   "value" :170.1
}
```

Lista 4.4: Saída POST compra

```
{
   "id_compra": 4,
   "user_id": "vleite",
   "content_id": 5,
   "value": 170.1,
   "data_hora_compra": "2021-08-06T02:17:32.902418",
   "data_hora_vencimento_pagamento": "2021-08-08T02:17:32.90241
      ↪ 8",
   "data_hora_pagamento": null,
   "status_pagamento": "PENDING_PAYMENT"
}
```

Por fim, o quarto endpoint é o de atualizar o estado de uma compra. Ele é chamado pelo robô de pagamentos que ao verificar um novo pagamento, identifica a compra relacionada e chama este endpoint passando a data e hora da compra. Com a data e hora de compra preenchida o estado do pagamento fica como pagamento feito.

- PUT /store/v1/users/{user_id}/purchase/{id_compra}

Lista 4.5: Entrada PUT compra para realizar o pagamento

```
{
   "data_hora_pagamento" :"2021-07-24T14:18:10.396887"
}
```

Lista 4.6: Saída PUT compra para realizar o pagamento

```
{
   "purchaseId": 4,
```

```
    "userId": "vleite",
    "contentId": 5,
    "value": 170.1,
    "data_hora_compra": "2021-08-06T02:17:32.902418",
    "data_hora_vencimento_pagamento": "2021-08-08T02:17:32.90241
      ↪ 8",
    "data_hora_pagamento": "2021-08-06T03:00:10.396887"
}
```

### 4.5.4 API Raiden

O Raiden é usado para fazer o pagamento de forma off-chain, abrir canal, depositar tokens e retirar tokens sem precisar fazer qualquer interface com o sistema proposto. Contudo, o sistema da loja necessita fazer interface com o Raiden no que se refere a identificar os pagamentos recebidos de outras contas para a conta da loja. Para isso, com o nó do Raiden em pé, o robô monitor de pagamentos consulta o endpoint GET Pagamentos da API do Raiden passando como parâmetro o endereço do token que se deseja identificar os pagamentos. A resposta dessa requisição é uma lista de pagamentos como pode ser mostrado no JSON de exemplo abaixo.

- GET /api/v1/payments/{token_address}

Lista 4.7: Saída do GET Pagamentos da API do Raiden

```
[
    {
        "amount": "5000000000000000",
        "log_time": "2021-10-27T03:23:36.492000",
        "event": "EventPaymentReceivedSuccess",
        "initiator": "0x34469334c075191e84ba199Aa362Af07A2ad65E5"
          ↪ ,
        "identifier": "94708677092625464 75",
        "token_address": "0x95B2d84De40a0121061b105E6B54016a49621
          ↪ B44"
    },
    {
        "amount": "5000000000000000",
        "log_time": "2021-10-27T03:24:02.356000",
        "event": "EventPaymentReceivedSuccess",
```

```
14        "initiator": "0x34469334c075191e84ba199Aa362Af07A2ad65E5"
             ↪ ,
15        "identifier": "17731617694232268125",
16        "token_address": "0x95B2d84De40a0121061b105E6B54016a49621
             ↪ B44"
17     }
18  ]
```

Em cada pagamento é possível ver a presença do valor pago, data e hora de pagamento, evento confirmando que o pagamento foi recebido com sucesso, conta origem que realizou o pagamento, o endereço do token que foi transferido e o identificador o pagamento. O identificador do pagamento é entrado como texto livre pelo usuário e deve ser usado para colocar o id da compra gerado no momento da compra. A partir desse id o monitor de pagamentos consegue atualizar a compra, que até então estava com o status de pagamento pendente, para pagamento feito.

Importante notar que caso não seja possível identificar a compra com o identificador preenchido ou o valor esteja errado ou o token transferido não é o adotado pela loja, a presença da conta origem do pagamento faz possível o estorno usando o endpoint POST Pagamento que possui JSON abaixo.

- POST /api/v1/payments/{token_address}/{partner_address}

Lista 4.8: Entrada do POST Pagamento da API do Raiden

```
1  {
2     "amount": "5000000000000000"
3  }
```

Lista 4.9: Saída do POST Pagamento da API do Raiden

```
1  {
2     "identifier": "17731617694232268125",
3     "secret_hash": "0x547c7f8a0204c99ca60f248290866d2131884bd52f
         ↪ 4136456a572a00126e7457",
4     "initiator_address": "0x34469334c075191e84ba199Aa362Af07A2ad
         ↪ 65E5",
5     "token_address": "0x95B2d84De40a0121061b105E6B54016a49621B44
         ↪ ",
6     "amount": "5000000000000000",
7     "secret": "0xd7a51b8cbece64f8d9ed05c1c1ddbc1c7e7627a2f8b1046
         ↪ 375b126e00ada1147",
```

```
8     "target_address": "0x9c6960cC8C2034C7295EE0407352366DF45
          ↪ cbffa"
9 }
```

Esses e demais endpoints da API do Raiden não usados diretamente pelo sistema da loja podem ser encontrados em [18].

## 4.6 Diagramas de sequências

### 4.6.1 Listagem de conteúdos

Na figura 4.3 é possível ver o diagrama de sequência executado para realizar a listagem de conteúdos no front-end. Primeiramente o front-end manda a requisição para a api de conteúdos identificando o id do cliente que está fazendo a requisição. Em seguida, a api de conteúdos faz uma requisição para a api de compras para listar as compras realizadas pelo cliente de forma a retornar no JSON se o front deve permitir a realização do download daquele conteúdo pelo cliente ou não.

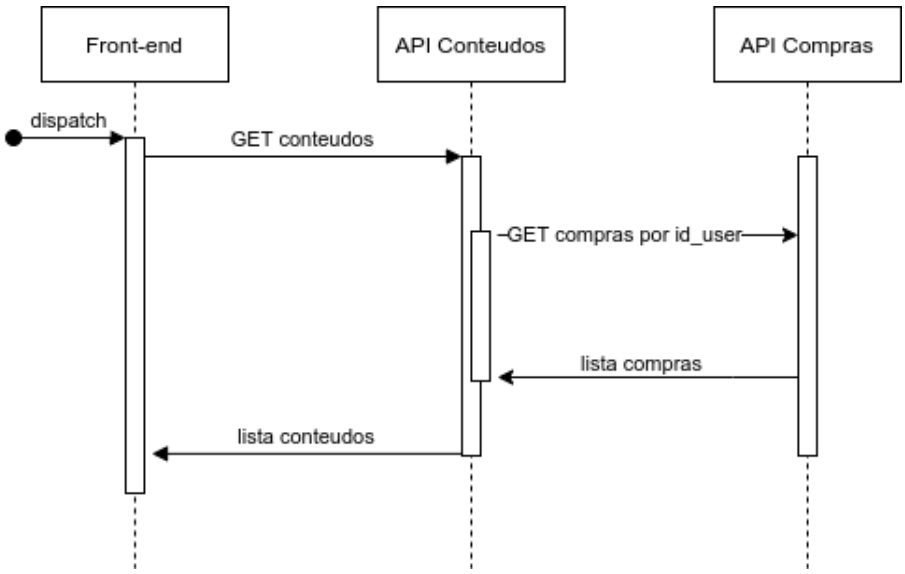


Figura 4.3: Diagrama de sequência da listagem de conteúdos.

### 4.6.2 Compra de Conteúdo

Na figura 4.4 pode-ser ver o diagrama de sequência da compra de conteúdos. O usuário seleciona um conteúdo no front-end e indica que quer realizar a compra. O front então faz um request POST para a api de conteúdos que por sua vez necessita realizar validações de que o usuário informado existe e de que o conteúdo existe e o preço do conteúdo. Ao finalizar as validações é persistido no banco de dados uma compra com status pendente de pagamento.

Neste momento é devolvido para o cliente o id da compra que deve ser informado como identificador do pagamento e também a conta que deve ser destinado o pagamento.

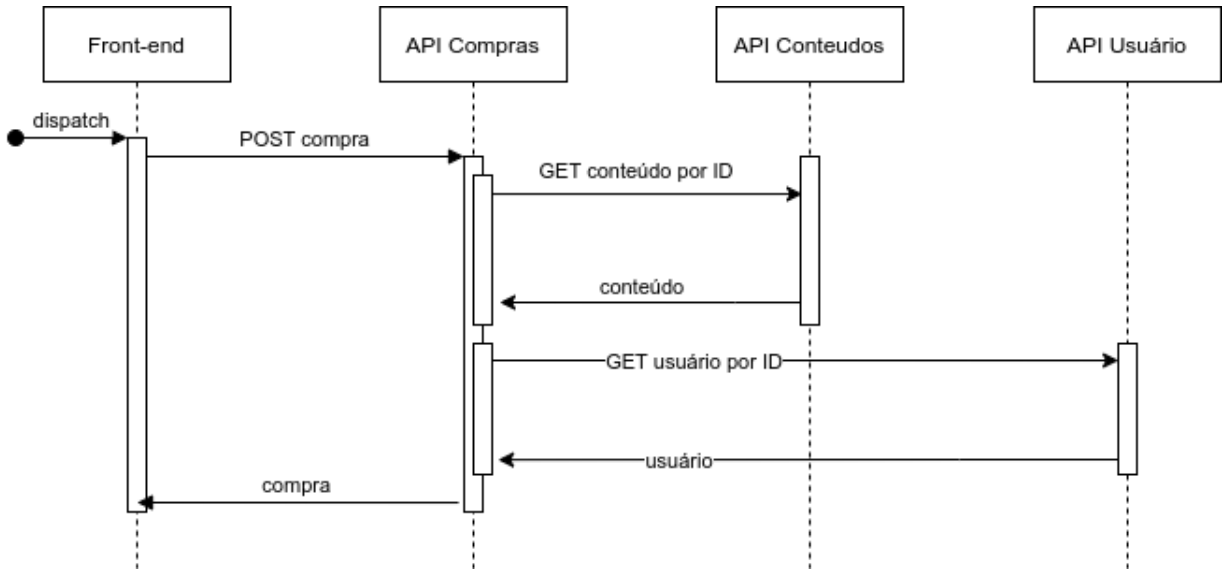


Figura 4.4: Diagrama de sequência da compra de conteúdo.

### 4.6.3 Monitor de Pagamentos

Na figura 4.5 pode-se ver o fluxo seguido pelo robô monitor de pagamentos que é responsável de identificar que um pagamento para uma determinada compra foi feita e liberar o download do conteúdo para o usuário. Primeiramente o robô carrega todas as compras que ainda não tiveram pagamento processado e também não passaram do prazo de vencimento. Em seguida é utilizada a rota GET pagamentos da API do Raiden para carregar as transferências feitas para a conta raiden da loja. Com todos os dados em memória é cruzado o identificador do pagamento com o id das compras em abertos para ver se alguma nova compra pode ser atualizada. Em fim, todas as compras que tiveram o pagamento realizado tem o seu status atualizado.

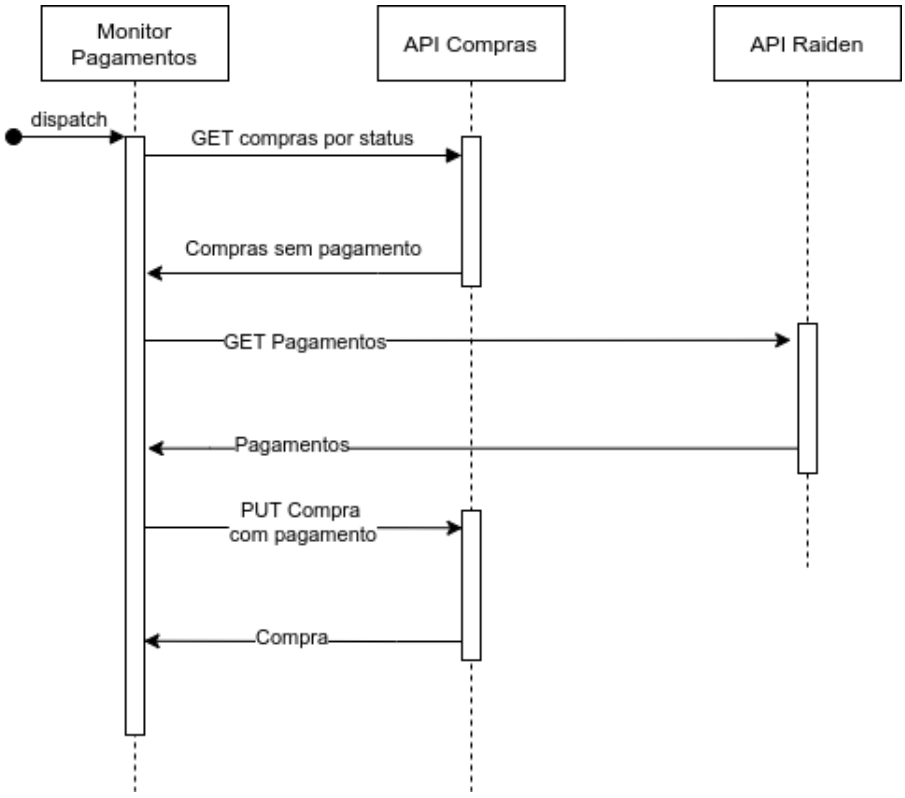


Figura 4.5: Diagrama de sequência do monitor de pagamentos.

### 4.6.4 Download de conteúdo

Na figura 4.6 pode ser visto o fluxo realizado no download de um conteúdo. O Front-end envia o resquest, passando quem é o usuário que está tentando acessar o conteúdo, para a API de conteúdos liberar o binário. A API de conteúdos carrega as compras que o usuário fez para verificar se o usuário comprou esse conteúdo mesmo. Uma vez validado que o usuário tem acesso ao conteúdo, com o caminho que está gravado nso metadados do conteúdo na base de dados, o binário é acessado no repositórios de arquivos e então enviado via HTTP para o Front-end.

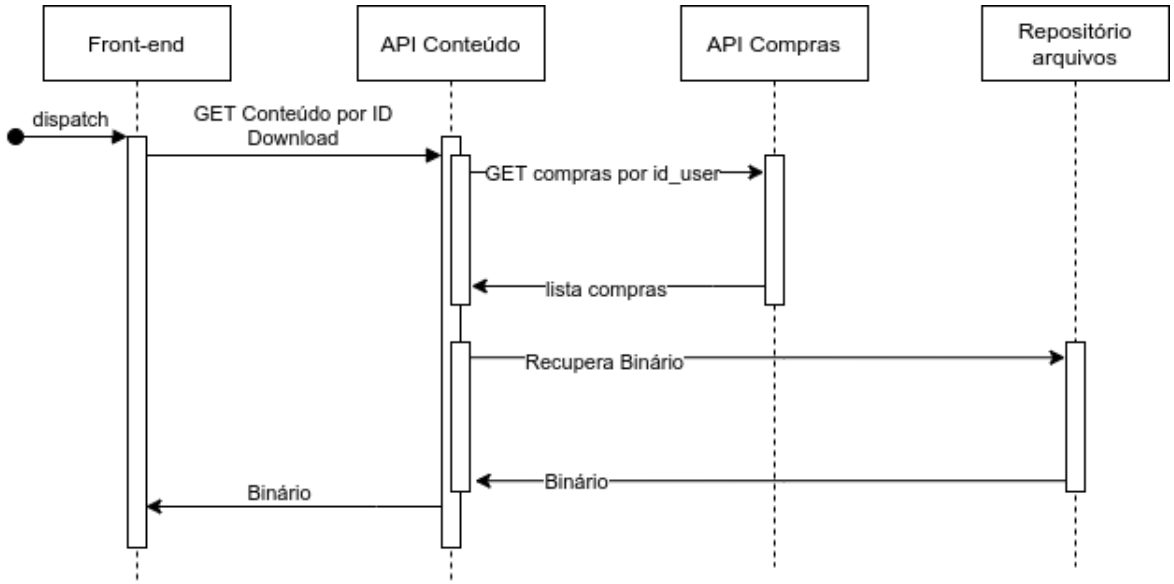


Figura 4.6: Diagrama de sequência do download de binários de conteúdos vendidos.

# CAPÍTULO 5

# Resultados e Discussão

## 5.1 Fluxo de compra

Neste capítulo é apresentado o fluxo da compra de um conteúdo por um usuário no sistema criado através da interface gráfica.

### 5.1.1 Tela inicial

A primeira tela é a tela inicial do sistema que mostra um texto informativo sobre o que é a plataforma com link para o site oficial do Raiden e ainda conta com opção para login e criação de usuário, conforme figura 5.1.

### 5.1.2 Criação de usuário

Caso o usuário ainda não possua cadastro na plataforma é possível criar o cadastro ao clicar em "Novo usuário"na tela da figura 5.1 e ser dado como entrada um id para o novo usuário e a sua senha de acesso.

### 5.1.3 Tela inicial com login feito

Uma vez criado o cadastro, o usuário pode fazer login por meio dos campos de identificação do usuário e senha da tela da figura 5.1. Caso o usuário exista e a senha esteja correta para o usuário informado então o login será aceito. Com o login aceito, os campos para entrar com login e senha são substituídos por uma opção para fazer o logout do sistema, a opção de criar um novo usuário desaparece e também há uma nova aba de "Conteúdos".


Figura 5.1: Tela inicial do sistema com usuário não logado.

### 5.1.4 Tela de listagem de conteúdo

Ao clicar na aba "Conteúdos"entra-se na tela da figura 5.2 onde todos os conteúdos são carregados no formato de uma lista. Na loja estão à venda PDFs de aulas de banco de dados.

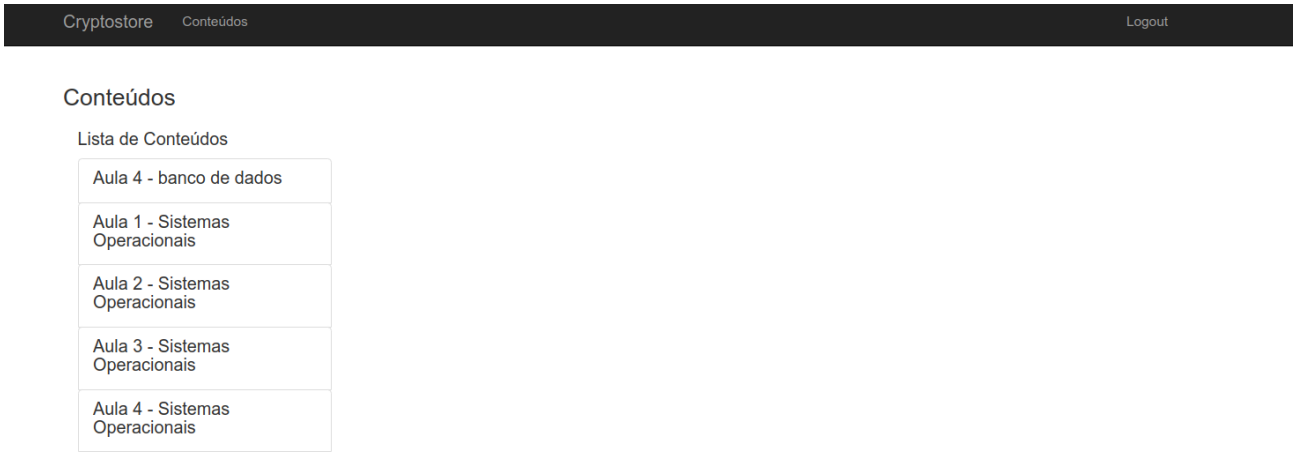


Figura 5.2: Tela mostrando uma lista de conteúdos para serem selecionados.

### 5.1.5 Tela de conteúdo não comprado

Após selecionar um conteúdo, na figura 5.3, vemos a presença de um botão COMPRAR mostrando que o conteúdo ainda não foi adquirido pelo usuário.

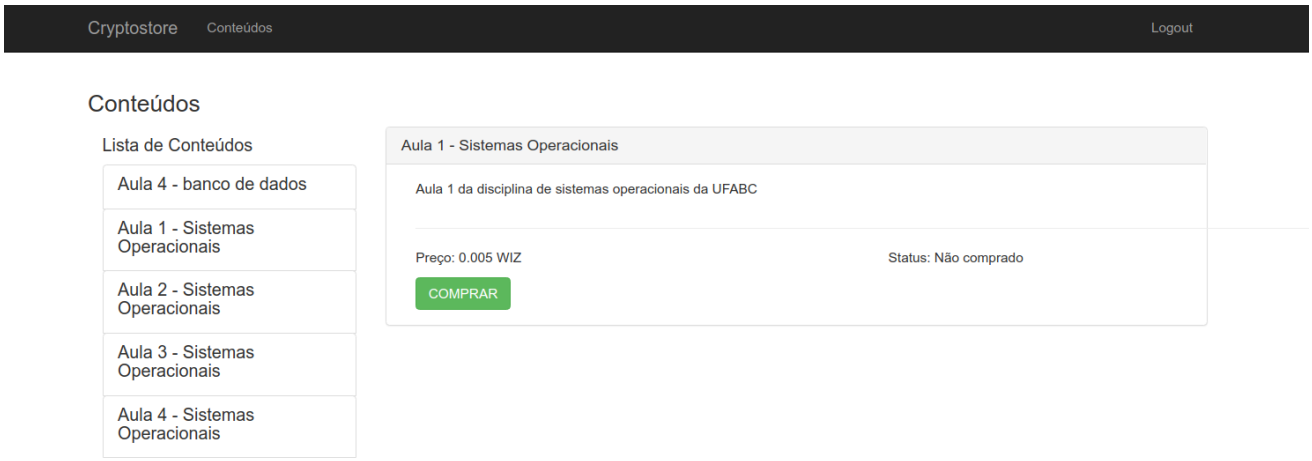


Figura 5.3: Tela mostrando detalhes de um conteúdo ainda não comprado.

### 5.1.6 Tela de confirmação de compra

Caso o usuário deseje comprar um conteúdo específico, ao clicar em "Comprar"aparece uma mensagem perguntado se o usuário realmente deseja realizar a compra conforme a tela da figura 5.4.

### 5.1.7 Tela de pagamento pendente

Na tela da figura 5.5 pode-se ver que o conteúdo teve o seu status alterado para "Pagamento pendente"e não há mais botão para interagir com a tela. Além disso, há um quadro informacional com o texto avisando o valor que deve ser enviado, a identificação do pagamento para que a loja consiga vincular o pagamento à compra de conteúdo pelo cliente e a conta de destino da transferência.

### 5.1.8 Tela Raiden de submissão de pagamentos

Na tela da figura 5.6 pode-se ver a tela do front-end do Raiden por onde podemos submeter pagamentos identificando a conta destino, o valor que queremos transferir, o token ao qual

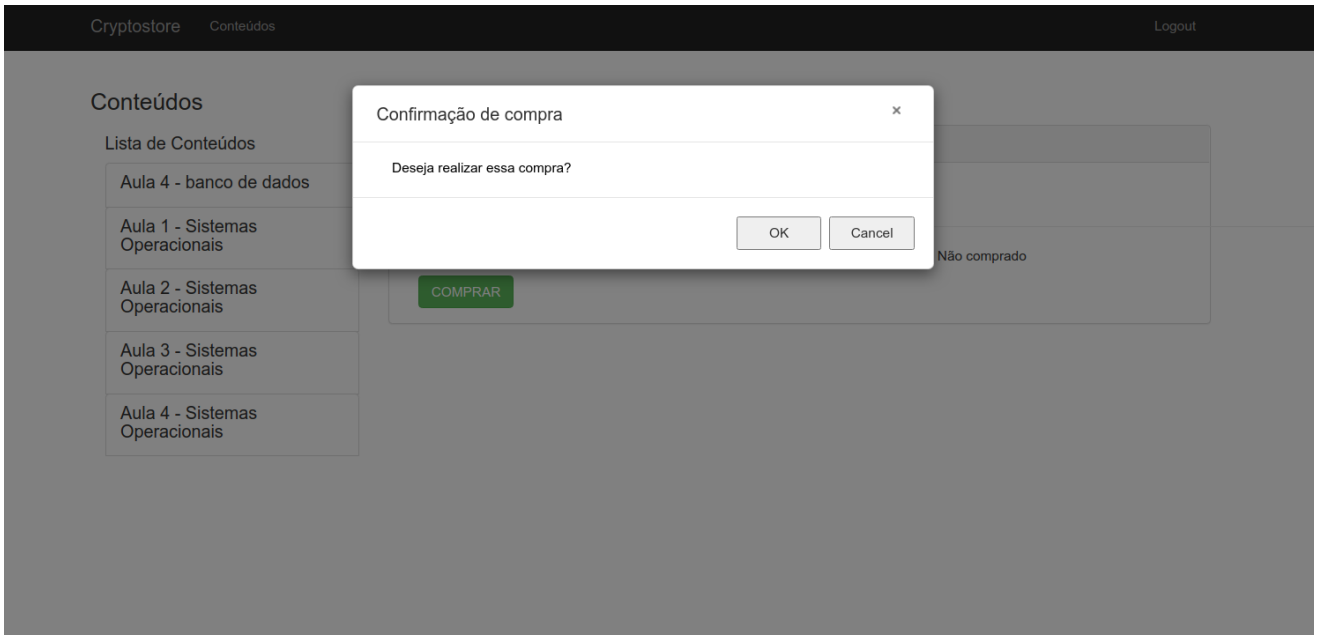


Figura 5.4: Tela de confirmação de compra.

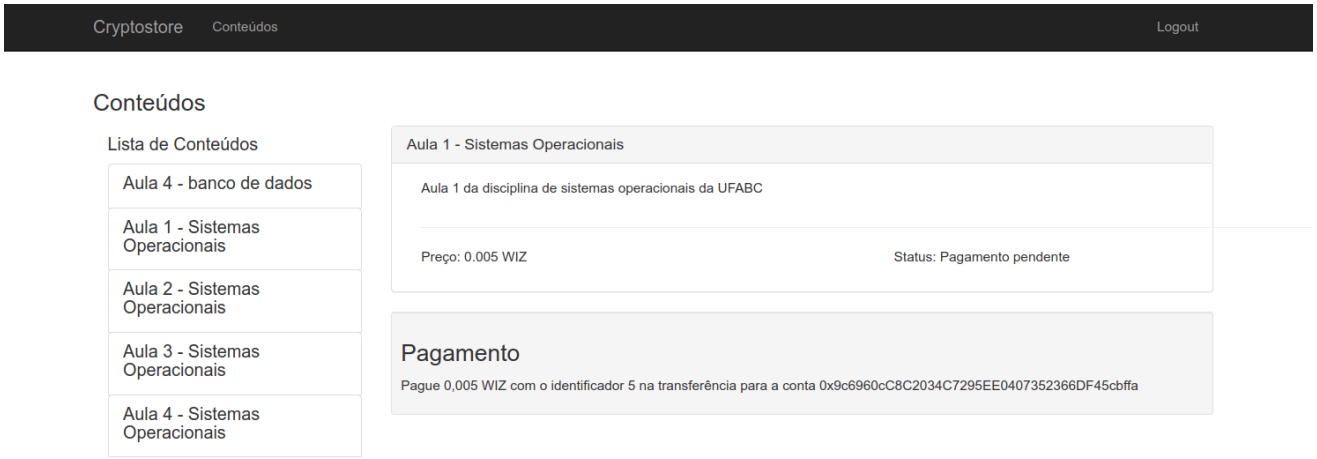


Figura 5.5: Tela mostrando detalhes de conteúdo pendente de pagamento.

estamos transferindo e um identificador, no caso o identificador gerado pela loja.

### 5.1.9 Tela de conteúdo com pagamento feito

Uma vez que o nó do Raiden da loja identifica que o pagamento foi recebido o robô de pagamentos recebe em sua consulta o novo pagamento com o identificador informado, verifica o valor pago e então atualiza o status do conteúdo como pagamento recebido.

Com o status da compra atualizado, ao atualizar a tela do navegador, o cliente pode ver

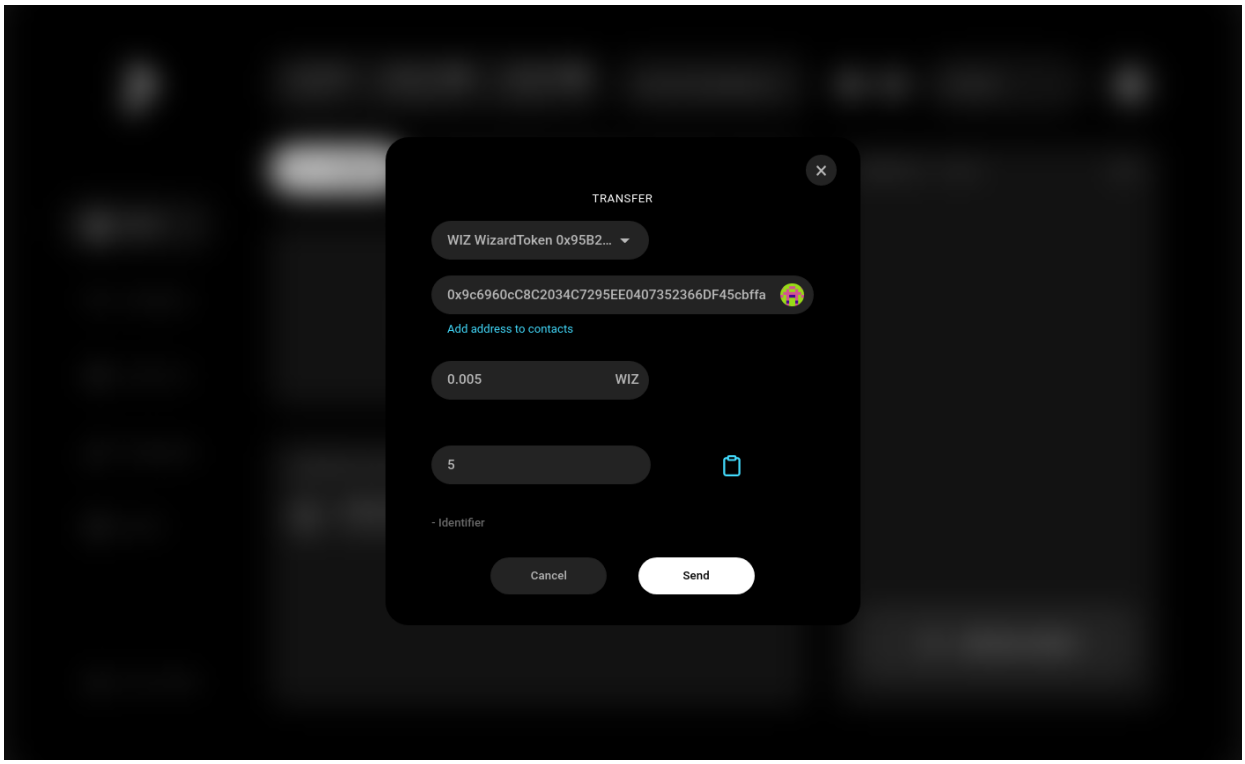


Figura 5.6: Tela do front-end do Raiden por onde pode-se submeter pagamento através do método PCN após um canal ser aberto entre as contas origem e destino.

que o conteúdo foi liberado pela presença botão de "DOWNLOAD"conforme pode ser visto na figura 5.7.

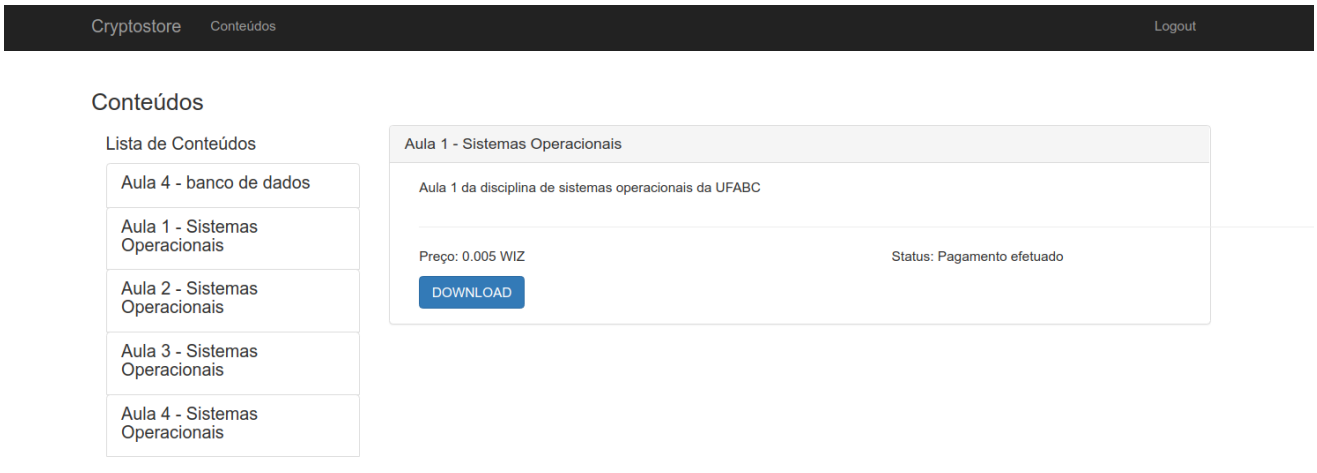


Figura 5.7: Tela de detalhes de conteúdo cujo pagamento foi realizado e conteúdo liberado para consumo.

## 5.2 Código Fonte

Os códigos fontes para as APIs desenvolvidas e front-end podem ser encontrados em [19–23].

## 5.3 Tempos de pagamentos

Em meios de pagamento um importante ponto de análise é tempo de duração do pagamento. O pagamento feito para a compra de um conteúdo na loja é de responsabilidade total do Raiden e portanto o sistema construído não influencia no tempo de pagamento. Considerando o pagamento como a chamada HTTP do endpoint POST de Pagamentos da API do Raiden, que é chamada para fazer o pagamento off-chain, pode-se coletar facilmente o tempo do pagamento como o tempo de duração da requisição HTTP até o retorno de uma resposta usando o software Postman. Assim foram coletados os tempos de 17 transferências off-chain que tiveram uma média de 3,66 s e desvio padrão de 0,46 s.

Considerando o pagamento off-chain como o necessário para transferir a posse de valor de uma pessoa para outra então pode-se admitir que o tempo médio de transferência atende a situações do dia a dia das pessoas. Contudo, após a transferência off-chain o valor monetário presente nos tokens ainda estará preso no canal da PCN e não poderá ser usado para mais nada até que seja resgatado para a conta Ethereum on-chain.

Complementando o tempo de pagamento off-chain, foram coletados tempos de 3 saques de tokens de canais para as contas on-chain. Os tempos coletados são da chamada HTTP do endpoint PACTH Channel em que há o envolvimento de transações on-chain. Foi obtido uma média de 1 min 39,12 s com desvio padrão de 6,66 s.

Pensando no uso cotidiano da PCN como um meio de pagamento a criação de um canal e o depósito de tokens no canal também são necessários para que posteriormente sejam feitas as transferências off-chain. Para análise, coletou-se o tempo de 3 aberturas de canal que tiveram uma média de 4 min 50,64 s com desvio padrão de 10,43 s e também o tempo de 6 depósitos de tokens em canais já abertos com média de 3 min 18,01s e desvio padrão de 9,64s. Importante reforçar que tanto a abertura do canal como o depósito de tokens fazer o uso de transações on-chain pois alteram o estado de um contrato inteligente.

Na tabela 5.1 pode-se ver o resumo dos tempos coletados. As medidas estão presentes no apêndice C.

Apesar das transferências off-chain terem os seus tempos pequenos o depósito e saque de tokens de canais ainda se apoiam em transações on-chain que precisam passar pelo processo de consenso e que são demoradas como pode ser constatado pelos resultados obtidos. Importante lembrar que os experimentos foram feitos na rede de testes Goerli e

| Ação | Média | Desvio Padrão |
|---|---|---|
| Abertura Canal | 4 min 50,64 s | 10,43 s |
| Depósito Tokens | 3 min 18,01 | 9,64 s |
| Pagamento | 3,66 s | 0,46 s |
| Saque Tokens | 1 min 39,12 | 6,66 s |

Tabela 5.1: Resultados de tempos coletados para as operações com a API do Raiden.

que portanto podem haver divergências para os resultados encontrados aqui comparados se realizados na mainnet da Ethereum que possui um número de transações momentâneas e mecanismo de consenso diferente.

## 5.4 Estimativa de custos

Como visto, os pagamentos off-chain podem ser feitos sem custo algum com a concordância das partes sendo, de certa forma, melhor do que em comparação a pagamentos on-chain. Por outro lado, as operações de depósito e saque de tokens no canal, por usarem uma transação on-chain, tem incidência de custo no processo de consenso mediante o custo do gas.

Foram feitos 3 depósitos e 3 saques para medir quantos tokens seriam descontados da conta Ethereum. Os resultados podem ser verificados na tabela 5.2. As medidas estão presentes no apêndice D.

Apesar dos experimentos estarem sendo feitos na rede de testes Goerli, cujo Ether não tem valor monetário real, para estimar o custo em real das transações foi usado o valor da cotação do Ether da mainnet no momento do experimento de R$23.553,40.

| Ação | Medida | Custo em Ether | Custo em Real |
|---|---|---|---|
| Depósito | Média | -0,000147202501776 | 3,47 |
| | Desvio padrão | 0,000000000000017 | 0,00 |
| Saque | Média | 0,000138 | 3,25 |
| | Desvio padrão | 0,000016 | 0,39 |

Tabela 5.2: Resultados de custos coletados para as operações com a API do Raiden.

Similarmente aos resultados de tempos de transação, os pagamentos off-chain resolvem o problema do custo de transação on-chain mas o ato de depositar e retirar tokens do canal ainda tem incidência de custo vinculado às transações on-chain.

## 5.5 Escalabilidade

Neste trabalho de graduação não foi estudado o quão escalável é o uso do Raiden como meio de pagamento. Esta análise deve ficar para trabalhos futuros, mas devido a sua arquitetura em que um nó pagador se comunica diretamente com um nó recebedor há a tese de que este limite esta relacionado apenas ao limite de transações que um nó Raiden consegue executar paralelamente e ao limite da camada de transporte de mensagens utilizado pelo Raiden.

# CAPÍTULO 6

# Considerações Finais

A existência de criptomoedas revolucionou o dinheiro como conhecemos pois fez surgir na sociedade um ativo que possui valor financeiro totalmente descentralizado fora do controle de organizações e governos. Apesar disso, devido funcionarem com base na tecnologia blockchain que faz o uso de um registro distribuído e processos de consenso para garantir alta disponibilidade, integridade, imutabilidade e auditabilidade, criptomoedas possuem certas limitações quando usadas como meio de pagamento. As principais limitações são referentes à escalabilidade, pois a capacidade de processamento de transações ainda é baixa comparada aos meios tradicionais de pagamento, incidência de taxas de processamento de transações e um tempo de confirmação de pagamento alto até que o processo de consenso termine.

Visando contornar essas limitações, foi criado o mecanismo de PCN com a ideia de que os pagamentos pudessem ser feitos off-chain de forma a evitar o processo de consenso até que seja desejado resgatar o saldo onde todos os pagamentos serão liquidados em uma única transação on-chain.

Na blockchain Ethereum, Raiden é um framework que implementa o conceito de PCN e foi usado neste trabalho de conclusão de curso para produzir uma prova de conceito. A prova de conceito é uma loja virtual construída com o frontend feito em Angular com Typescript e o backend em Spring com Java e Kotlin. A loja, como visto ao longo deste trabalho, permitiu a comercialização de conteúdos digitais aceitando como pagamento tokens pagos de forma off-chain usando o Raiden como camada de pagamento. Pelos pagamentos serem feitos em uma rede de testes da Ethereum os tokens usados não possuem valor monetário em si mas se o experimento tivesse ocorrido na rede oficial (mainnet) da Ethereum poderia ser dito que foi feita uma negociação que não precisou de bancos ou entidades financeiras para ocorrer.

Com o sistema construído foi possível analisar os tempos e custos envolvidos nos pagamentos. Como prometido, os pagamentos off-chain se demonstraram rápidos e baratos já que são feitos diretamente entre dois nós Raiden e podem ser feitos sem taxas. Contudo, para a realização de um pagamento off-chain há a necessidade de depositar tokens em um contrato inteligente anteriormente sujeito a taxas e ao tempo do processo do consenso que podem variar conforme atividade da rede. Para posterior uso dos tokens que não para pagar pelo Raiden também é necessário fazer transações on-chain para realizar o saque deles que também está sujeito a taxas e tempo de espera.

Mediante a tais resultados, fica evidente que os pagamentos off-chain tem situações específicas em que as vantagens de escalabilidade, sem taxas e transações rápidas podem ser usufruídas. Para evitar com que se faça transações on-chain com frequência, para pagamentos diretos a uma conta por meio de um canal aberto com ela deve-se depositar tokens no canal cuja quantidade vá ser transferida durante um horizonte longo de tempo. Essa estratégia permite prolongar o máximo possível um novo depósito de tokens no canal. A estratégia análoga pode ser feita para o saque onde o custo de saque pode ser mais otimizado conforme a frequência de saques diminuir.

PCN também permite que transferências sejam feitas entre contas sem um canal aberto entre elas utilizando os canais que as contas têm abertas com nós intermediários. O uso dessas transferências mediadas também evitaria a necessidade de depósitos de tokens em canais específicos abertos diretamente com o destino alvo do pagamento. O uso delas é uma estratégia que permite diminuir o número de transações on-chain mas que a eficácia seria proporcional ao tamanho e acoplamento da rede. Além disso, o uso de nós mediadores está sujeito à cobrança de novas taxas por eles para estimular a atividade de mediação [2].

Assim, o caso de uso para pagamentos usando criptomoedas com o método de PCN que melhor aproveita as suas vantagens são para pagamentos que são feitos com frequência entre duas contas onde um único deposito de tokens no canal permite uma quantidade razoável de consumos de serviços ou produtos da conta de destino do pagamento.

Durante o desenvolvimento deste trabalho foram encontradas dificuldades para o entendimento do funcionamento do Raiden e exploração de suas funcionalidades. Primeiramente, foi encontrado um problema de compatibilidade entre o Raiden com a versão recém lançada do Infura. Sem logs ou manual que pudesse ajudar a encontrar a causa, foi necessário entrar em contato com o time de desenvolvimento do Raiden para obter o suporte necessário através de sua página no Gitter. Descoberta a causa raiz do problema e feita a promessa de que a próxima versão do Raiden traria de volta a compatibilidade com o nó remoto da Ethereum suportado pelo Infura o sistema foi proposto e teve o seu desenvolvimento iniciado com base apenas nas especificações de APIs presentes na documentação técnica. Após o lançamento da nova versão do Raiden ainda foi necessário esperar a atualização dos serviços do

Raiden(PathFinding Service e Monitoring Service), não usados neste trabalho, no ambiente de testes para compatibilidade com a nova versão que mesmo não sendo usados o nó do Raiden necessitava desses serviços para subir.

Fazer uma análise de escalabilidade do Raiden, uso de nós mediadores para realizar pagamentos entre nós que não tenham canais abertos entre si diretamente com a coleta de métricas de tempos e custos, uso dos serviços do Raiden para encontrar o melhor caminho de canais abertos entre dois nós e comparação com outras implementações de PCNs podem ser temas de trabalhos futuros.

# Bibliografia


[**1**] A. Braga, F. Marino, e R. dos Santos, "Segurança de aplicaçoes blockchain além das criptomoedas," *Livro-texto dos minicursos SBSeg*, pags. 99–148, 2017.

[**2**] Raiden. Raiden network specification. [Online]. Available: https://raiden-network-specification.readthedocs.io/en/latest/index.html

[**3**] S. Nakamoto e A. Bitcoin, "A peer-to-peer electronic cash system," *Bitcoin.–URL: https://bitcoin. org/bitcoin. pdf*, vol. 4, 2008.

[**4**] F. G. Greve, L. S. Sampaio, J. A. Abijaude, A. C. Coutinho, Í. V. Valcy, e S. Q. Queiroz, "Blockchain e a revolução do consenso sob demanda," *Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC)-Minicursos*, 2018.

[**5**] S. Mercan, A. Kurt, E. Erdin, e K. Akkaya, "Cryptocurrency solutions to enable micro-payments in consumer iot," *IEEE Consumer Electronics Magazine*, 2021.

[**6**] K. L. The blockchain scalability problem the race for visa-like transaction speed. [Online]. Available: https://towardsdatascience.com/the-blockchain-scalability-problem-the-race-for-visa-like-transaction-speed-5cce48f9d44

[**7**] Ethereum. [Online]. Available: https://ethereum.org/en/

[**8**] R. Network. Raiden protocol explained. [Online]. Available: https://medium.com/raiden-network/raiden-protocol-explained-489b429fcd90

[**9**] C. C. Miers, G. P. Koslovski, M. A. Pillon, M. A. Simplício Jr, B. B. R. UZH, e J. H. Battisti, "Análise de mecanismos para consenso distribuído aplicados a blockchain," 2019.

[**10**] Goerli. Goerli. [Online]. Available: https://goerli.net/

[**11**] Infura. Infura. [Online]. Available: https://infura.io/

[**12**] E. Conner. Understanding ethereum gas, blocks and the fee market. [Online]. Available: https://medium.com/@eric.conner/understanding-ethereum-gas-blocks-and-the-fee-market-d5e268bf0a0e

[**13**] V. Coutts. Ethereum tokens explained - a beginner's guide to token standards: Everything you need to know. [Online]. Available: https://medium.com/linum-labs/ethereum-tokens-explained-ffe9df918008

[**14**] A. Rajeevan. Tokens,gas and gas limit in ethereum standards: Everything you need to know. [Online]. Available: https://arunrajeevan.medium.com/tokens-gas-and-gas-limit-in-ethereum-f07790f56d8f

[**15**] M. Hodgson. Introducing p2p matrix. [Online]. Available: https://matrix.org/blog/2020/06/02/introducing-p-2-p-matrix

[**16**] R. Network. Build on top of the api. [Online]. Available: https://developer.raiden.network/#Examples

[**17**] Etherscan. Contract 0x95b2d84de40a0121061b105e6b54016a49621b44. [Online]. Available: https://goerli.etherscan.io/address/0x95B2d84De40a0121061b105E6B54016a49621B44

[**18**] Raiden. Raiden's api documentation. [Online]. Available: https://raiden-network.readthedocs.io/en/stable/rest_api.html

[**19**] V. H. da Costa Leite. Cryptostore-api-contents. [Online]. Available: https://github.com/vhcleite/CryptoStore-API-Contents

[**20**] ——. Cryptostore-api-purchases. [Online]. Available: https://github.com/vhcleite/CryptoStore-API-Purchases

[**21**] ——. Cryptostore-api-users. [Online]. Available: https://github.com/vhcleite/CryptoStore-API-Users

[**22**] ——. Cryptostore-payment-monitor. [Online]. Available: https://github.com/vhcleite/CryptoStore-Payment-Monitor

[**23**] ——. Cryptostore-spa. [Online]. Available: https://github.com/vhcleite/CryptoStore-SPA

[**24**] Raiden. The raiden wizard on-boarding tool. [Online]. Available: https://github.com/raiden-network/raiden-wizard

[**25**] ——. Raiden releases. [Online]. Available: https://github.com/raiden-network/raiden/releases

# APÊNDICE A

## Contrato API de Usuários

- GET /store/v1/users/{id_user}

Lista A.1: Saída GET usuário por ID

```
{
   "id": "vleite"
}
```

- POST /store/v1/users

Lista A.2: Entrada POST usuário

```
{
   "id" :"vleite",
   "password" :"teste"
}
```

Lista A.3: Saída POST usuário

```
{
   "id" :"vleite",
   "password" :"teste"
}
```

- POST /store/v1/users/login

Lista A.4: Entrada POST usuário

```
{
   "id" :"vleite",
   "password" :"teste"
}
```

Em caso de login e senha válidos então será retornado HTTP 200 - OK com corpo de saída vazio

# APÊNDICE B

## Contrato API de Conteúdos

- GET /store/v1/content/{id_content}/download?userId={user_id}

  saída: binário arquivo

- GET /store/v1/content?userId={user_id}

Lista B.1: Saída GET lista conteúdos

```
{
   "content": [
      {
         "id": 1,
         "name": "Aula 1- banco de dados",
         "description": "Aula 1da disciplina de banco de
            ↪ dados da UFABC",
         "price": 0.005,
         "status": "ACTIVE",
         "status_pagamento": "PAYMENT_MADE"
      },
      {
         "id": 2,
         "name": "Aula 2- banco de dados",
         "description": "Aula 2da disciplina de banco de
            ↪ dados da UFABC",
         "price": 0.005,
```

```
16          "status": "ACTIVE",
17          "status_pagamento": "PAYMENT_NOT_MADE"
18       },
19    ]
20 }
```

- POST /store/v1/content

Lista B.2: Entrada POST conteúdo

```
1 {
2    "name" :"Aula 10- banco de dados",
3    "description" :"Aula 10da disciplina de banco de dados
       ↪ da UFABC",
4    "price" :0.005,
5    "status" :"ACTIVE",
6    "path" :"/home/user/conteudos/07_08_09
       ↪ _Engenharia_bd_projeto_conceitual.pdf"
7 }
```

Lista B.3: Saída POST conteúdo

```
1 {
2    "id": 5,
3    "name": "Aula 10- banco de dados",
4    "description": "Aula 10da disciplina de banco de dados
       ↪ da UFABC",
5    "path": "/home/user/conteudos/07_08_09
       ↪ _Engenharia_bd_projeto_conceitual.pdf",
6    "price": 0.005,
7    "status": "ACTIVE"
8 }
```

# APÊNDICE C

## Medidas de tempos

Abaixo encontram-se as medidas coletadas a respeitos de tempos de operações utilizando o framework Raiden.

| Tempos de abertura de Canal | |
|---|---|
| Medida | Tempos |
| 1 | 4m 59,28s |
| 2 | 4m 53,58s |
| 3 | 4m 39,06s |
| Média | 4m 50,64s |
| Desvio Padrão | 10,43s |

Tabela C.1: Medidas de tempos de abertura de canal.

| Tempos de depósitos de tokens no canal | |
|---|---|
| Medida | Tempos |
| 1 | 3m 20,94s |
| 2 | 3m 2,16s |
| 3 | 3m 24,18s |
| 4 | 3m 24,94s |
| 5 | 3m 10,21s |
| 6 | 3m 25,62s |
| Média | 3m 18,01s |
| Desvio Padrão | 9,64s |

Tabela C.2: Medidas de tempos para depósito de tokens.

| Tempos de pagamentos off-chain | |
|---|---|
| Medida | Tempos |
| 1 | 4,53s |
| 2 | 4,76s |
| 3 | 3,56 |
| 4 | 3,4s |
| 5 | 3,62s |
| 6 | 3,33s |
| 7 | 4,16s |
| 8 | 3,47s |
| 9 | 3,45s |
| 10 | 3,37s |
| 11 | 3,43s |
| 12 | 3,33s |
| 13 | 4,23s |
| 14 | 3,59s |
| 15 | 3,53s |
| 16 | 3,38s |
| 17 | 3,15s |
| Média | 3,66s |
| Desvio Padrão | 0,46s |

Tabela C.3: Medidas de tempos pagamentos off-chain.

| Tempos de saques | |
|---|---|
| Medida | Tempos |
| 1 | 1m 35,64s |
| 2 | 1m 34,92s |
| 3 | 1m 46,80s |
| Média | 1m 39,12s |
| Desvio Padrão | 6,66s |

Tabela C.4: Medidas de tempos saques de tokens.

# APÊNDICE D

## Estimativas de custo

Abaixo foram coletados os descontos de Ether ocorridos após saques de 0,005 WIZ de um canal para a conta principal usando a cotação de 1 Ether valendo R$23.553,40.

| Custo de saques | | | |
|---|---|---|---|
| Medida | Ether Conta | Delta Ether | Custo em Reais |
| Estado inicial | 0,320954058242688247 | 0 | R$0,00 |
| Saque 1 | 0,320797135097947192 | -0,000156923144741084 | -R$3,70 |
| Saque 2 | 0,320668176206375437 | -0,000128958891571773 | -R$3,04 |
| Saque 3 | 0,320540001332791317 | -0,000128174873584130 | -R$3,02 |
| Média | | -0,000138 | -R$3,25 |
| Desvio Padrão | | 0,000016 | -R$0,39 |

Tabela D.1: Medidas de custo de Ether após saque de canal.

| Custo de depósitos | | | |
|---|---|---|---|
| Medida | Ether Conta | Delta Ether | Custo em Reais |
| Estado inicial | 0,0983615536931085 | 0 | R$0,00 |
| Depósito 1 | 0,09821435119134207 | -0,000147202501766405 | -R$ 3,47 |
| Depósito 2 | 0,098067148689546577 | -0,000147202501795521 | -R$ 3,47 |
| Depósito 3 | 0,097919946187780147 | -0,000147202501766419 | -R$ 3,47 |
| Média | | -0,000147202501776 | -R$ 3,47 |
| Desvio Padrão | | 0,000000000000017 | R$ 0,00 |

Tabela D.2: Medidas de custo de Ether após depósito em canal.

# APÊNDICE E

## Setup Raiden

Para subir o nó do Raiden, primeiro é necessário criar uma conta Ethereum. O Raiden oferece um assistente denominado Raiden Wizard que se encarrega de criar a conta na rede Goerli e alimentá-la com alguns tokens para testes [24]. Neste trabalho foi utilizado a versão *Raiden Wizard Alderaan - Görli Wizard*. Infelizmente, Raiden passou por uma atualização que retirou a retrocompatibilidade do Raiden Wizard com o consumo do Infura e por este motivo, no momento que este trabalho foi feito, somente a funcionalidade de criar a conta da Ethereum utilizada pelo nó do Raiden estava funcionando tendo que utilizar outro software para inicializar o nó.

Para inicializar o nó do Raiden localmente foi utilizado a versão 2.0.0 para Linux que pode ser encontrada em [25]. O comando para a inicialização de um nó na rede de testes pode ser encontrado abaixo:

```
$ ./raiden-v2.0.0-linux-x86_64 --matrix-server=<matrix server
  ↪  address> --eth-rpc-endpoint <infura address> --api-
  ↪ address "http://localhost:<port>" --keystore-path ~/.
  ↪ ethereum/keystore --network-id goerli --environment-type
  ↪  development --accept-disclaimer --password-file $<path
  ↪ to password file> --address <ethereum account address>
```

Para os testes, o comando acima é executado duas vezes. Uma para subir o nó local da loja e outra para subir o nó do cliente. Para cada, somente era alterado a porta que se usava e o endereço da conta Ethereum. O parâmetro matrix server era o disponibilizado pelo próprio Raiden "https://transport.demo002.env.raiden.network"que foi utilizado por conveniência mas é totalmente possível subir um servidor de matrix próprio.