



# Sistema de Gestão de Transferências em Rede Blockchain

Josué Patatas 42199

João Vieira 43385

**Orientadores: Eng. Pedro Pereira**

**Relatório do projeto realizado no âmbito de Projeto e Seminário  
Licenciatura em Engenharia Informática e de Computadores  
Semestre de Verão 2021/2022**

**Setembro 2022**



# Instituto Superior de Engenharia de Lisboa

Licenciatura em Engenharia Informática e de Computadores

## Sistema de Gestão de Transferências em Rede Blockchain

Josué Patatas 42199

João Vieira 43385

---

Orientadores: Eng. Pedro Pereira

---

---

Relatório do projeto realizado no âmbito de Projeto e Seminário Licenciatura em  
Engenharia Informática e de Computadores  
Semestre de Verão 2021/2022

Setembro 2022



## Agradecimentos

Gostaríamos de agradecer ao nosso Orientador Pedro Pereira pela paciência que teve connosco como também por ter dado a oportunidade de fazermos este projeto. Temos noção que nem sempre fomos o grupo mais fácil de orientar e que em grande parte do tempo estivemos "perdidos" em relação ao que fazer, mas em varias situações conseguindo-nos apontar em maneiras de enfrentar os problemas que nos aparecia. Gostaríamos de agradecer as nossas famílias pelo apoio oferecido durante a realização do projecto.



## Resumo

Com a melhoria dos dispositivos de utilizadores comuns em termos de velocidade de processamentos e de transmissão de mensagem, a ideia de uma rede descentralizada torna-se possível, conhecida como **blockchain**. Este tipo de rede permite a utilizadores criarem a sua própria comunidade com as suas próprias regras e foi originalmente utilizada para tirar poder a entidades controladoras como bancos e governos. Atualmente este tipo de redes são maioritariamente conhecidas devido à popularidade de cripto-moedas que têm mostrado alguns projetos promissores e outros que apenas utilizam a sua explosão de popularidade para explorar o cidadão comum, com projetos e esquemas de moralidade duvidosa.

A tecnologia **blockchain** tem vários benefícios, mas como qualquer outra tecnologia tem algumas lacunas que, ao invés de serem solucionadas apenas é criado um ambiente alternativo com comportamentos limitados. Um dos principais problemas devido a ser uma rede anónima sem entidade reguladora são as transações entre utilizadores, isto porque se baseiam fortemente na confiança entre as duas entidades. Isto abre caminho ao problema já mencionado anteriormente as burlas por entidades que queiram usufruir desta metodologia.

Para solucionar este problema foi desenvolvido um sistema com base numa rede **blockchain** que possibilita as transações com inovações como possibilidade de reembolso. Este sistema poderá ser utilizado por vários tipos de aplicações principalmente as comerciais. Esta sistema será acessível a todos os utilizadores da **blockchain** mas terá operações mediadoras que apenas podem ser controladas pelo nosso **Web Application Programming Interface** (API) que com o uso de **smart contracts** são implementadas todas as funcionalidades necessárias para que seja possível uma transação entre utilizadores de uma forma mais segura.

Foi construída uma aplicação **Web** para demonstrar todas as funcionalidades e o potencial deste sistema, esta aplicação é constituída por uma API que é responsável pela ligação entre o cliente e a base de dados onde se encontra toda a informação da aplicação.

**Palavras-chave:** blockchain, base de dados, cripto-moedas, **Web Application Programming Interface**, aplicação **Web**, **smart contracts**

## Abstract

As common user devices improve in terms of processing speed and processing power, the idea of a decentralized network became possible, known as **blockchain**. This type of network allows users to create their own community with their own rules and was originally used to take power away from controlling entities like banks and governments. Today these types of networks are mostly known due to the popularity of crypto-currencies that have shown some promising projects, but also some that just use their surge in popularity to exploit the common citizen with projects and schemes of dubious morality.

The **blockchain** technology has several benefits, but like any other technology it has some loopholes that instead of being solved only an alternative environment with limited behaviors is created. One of the main problems due to it being an anonymous network with no regulatory entity are the transactions between users, this is because they rely heavily on trust between the two entities. This opens the way to the previously mentioned problem of fraud by entities that want to take advantage of this methodology.

To solve this problem, a system based on a **blockchain** network was developed, which enables transactions with innovations such as the possibility of reimbursement. This system will be accessible to all users of the **blockchain** but will have mediating operations that can only be controlled by our **Web Application Programming Interface** (API) that with the use of **smart contracts** are implemented all the necessary functionalities for a transaction between users to be possible in a more secure way.

A **Web** application was built to demonstrate all the features and potential of this system, this application consists of an API that is responsible for the connection between the client and the database where all the application information is located

**Keywords:** blockchain, database, cryptocurrencies, Web Application Programming Interface, Web application, smart contracts



# Índice

<b>1</b>	<b>Introdução</b>	<b>1</b>
1.1	Formulação do problema . . . . .	1
1.2	A Solução proposta . . . . .	2
1.3	Organização do Documento . . . . .	2
<b>2</b>	<b>Infraestrutura da Proposta</b>	<b>3</b>
2.1	Utilizadores do Sistema . . . . .	3
2.2	Processo de criação de uma transação . . . . .	3
2.3	Arquitetura do projeto . . . . .	4
<b>3</b>	<b>Tecnologias e Conceitos</b>	<b>5</b>
3.1	<i>Blockchain</i> . . . . .	5
3.2	Rede Kovan Testnet . . . . .	6
3.3	Rede Goerli . . . . .	6
3.4	Infura.io . . . . .	6
3.5	Solidity . . . . .	7
3.6	Smart Contracts . . . . .	7
3.7	Remix . . . . .	8
3.8	Web3j . . . . .	8
3.9	Metamask . . . . .	9
<b>4</b>	<b>API</b>	<b>10</b>
4.1	Base de Dados . . . . .	10
4.2	Contrato . . . . .	10
4.2.1	Estrutura de uma Exchange . . . . .	11
4.2.2	Métodos do contrato . . . . .	13
4.3	Autentificação . . . . .	17
4.3.1	Tokens . . . . .	18
4.4	Rotas do API . . . . .	18
<b>5</b>	<b>APP</b>	<b>19</b>
5.1	Web API . . . . .	19
5.2	Base de Dados . . . . .	20
5.3	Aplicação Web . . . . .	23
5.3.1	Navegação Páginas . . . . .	23
<b>6</b>	<b>Conclusão</b>	<b>34</b>
6.1	Trabalho Futuro . . . . .	34
<b>7</b>	<b>Referencias</b>	<b>35</b>
<b>8</b>	<b>Apêndices</b>	<b>36</b>
8.1	Smart Contract . . . . .	36

# 1 Introdução

Este projeto é realizado no âmbito da disciplina “Projeto e seminário” do curso “Engenharia informática e de computadores” do Instituto Superior de Engenharia de Lisboa (ISEL).

Com o aumento da popularidade de cripto-moedas e avanços tecnológicos na área da *blockchain*, prevê-se que o uso desta tecnologia vá aumentar ainda mais. Visto estas observações, este grupo procura aumentar o seu conhecimento sobre a tecnologia *blockchain* e o seu futuro potencial através do desenvolvimento e implementação de um sistema que resolva alguns dos problemas encontrados atualmente na compra e venda de produtos com cripto-moedas e disponibilizá-lo ao utilizador comum.

Para tal, é necessário um período de investigação para poder primeiro identificar como podemos alcançar dado objetivo, como também, potenciais limitações que possam aparecer mais tarde.

## 1.1 Formulação do problema

Uma das principais funcionalidades da *blockchain* é o facto de ser descentralizada, isto em muitos sentidos é visto como uma vantagem, mas em certas situações há necessidade de ter uma entidade reguladora/neutra, devido a certas situações e lacunas com a utilização desta tecnologia para transações monetárias. Este é o principal objetivo do projeto, oferecer alguma regulação nestas transações para garantir a satisfação e segurança para o utilizador comum através de *smart contracts* autónomos. Dando o exemplo de compras de produtos online, depois de ter sido feito o pagamento de um produto, não há entidade que garanta que o vendedor do mesmo tenha de seguir um determinado nível de qualidade ou até que o mesmo tenha sido entregue. E como todas as transferências na *blockchain* são finais, não há maneira de a reverter qualquer transação impossibilitando o cliente ter direito a reembolso caso algum imasse aconteça no seguimento compra do produto, este tipo de insegurança faz com que o cliente comum não participe neste tipo de transações optando então por converter as cripto-moedas para moedas comuns e depois realizar alguma compra, tendo assim ter de passar por terceiros para a realização de uma simples compra.

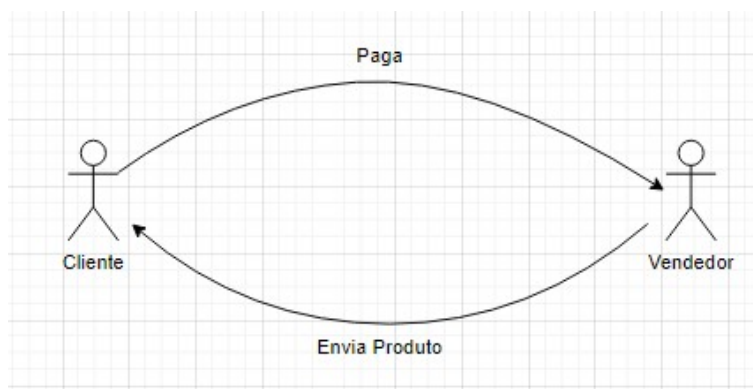


Figura 1: Exemplo de uma transação comum

## 1.2 A Solução proposta

A solução proposta baseia-se na criação de *smart contract* que armazene o montante durante determinado período. Durante este período é possível que o cliente requisiute o cancelamento da transação, podendo desta maneira, receber um reembolso.

O funcionamento do *smart contract* será semelhante aos pagamentos utilizados atualmente, a criação de uma referência onde determinadas operações estarão disponíveis.

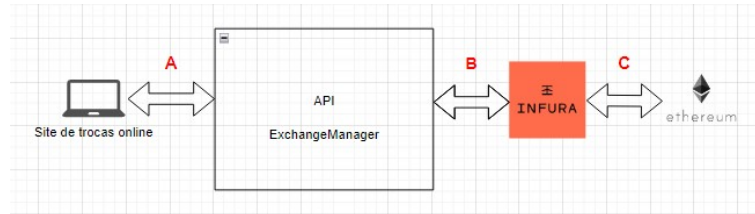


Figura 2: Modulos utilizados na criação de uma nova Exchange

A: Pedido de criação de novas transações, requisito de informação de entidades criadas, emissão de reembolso de transação existente.

B: Ligação entre o API e o *end point* disponibilizado pelo Infura para interagir com a rede *blockchain*.

C: Transmissão das operações para a rede *blockchain*

## 1.3 Organização do Documento

O restante documento encontra-se organizado da seguinte forma. No Capítulo 2 será descrita a infraestrutura da proposta, apresentado os diferentes tipos de utilizadores do sistema e o processo de criação de transações. No Capítulo 3 será apresentada toda a informação essencial sobre as tecnologias que irão ser utilizadas na realização do projeto proposto . No Capítulo 4 será descrita a API assim como todas as suas funcionalidades.No Capítulo 5 será descrita a APP assim como todas as suas funcionalidades . A conclusão s e as propostas de trabalho futuro constituem o tema do Capítulo 6.

## 2 Infraestrutura da Proposta

As cripto-moedas foram consideradas por Sichel e Calixtoi (2018, p. 1922) como “fenómeno tecnológico que está rompendo paradigmas económicos e sociais, maravilhando o mundo com suas possibilidades e alarmando com seus efeitos económicos”. A inserção das cripto-moedas no mercado tratou-se de uma revolução em todos os sentidos, incluindo tecnológicos. Ao utilizar esta tecnologia o processo da compra de um produto até a receção do mesmo pelo cliente com uma satisfação plena é também um desafio informático. Há nomeadamente um vácuo nesse processo que se propõe preencher através desta aplicação trazendo inovação nesta área.

Este capítulo irá descrever o funcionamento e a estrutura da *API* criada para preencher o vácuo mencionado tal como a aplicação criada para demonstrar as suas funcionalidades e potenciais aplicações. São apresentados os diferentes tipos de utilizador que fazem parte tanto da *API* como da *APP* na Secção 2.1, o processo de criação de uma transação entre cliente e vendedor irá ser descrita na Secção 2.2. A arquitetura desta proposta de projeto será descrita na Secção 2.3.

### 2.1 Utilizadores do Sistema

Esta secção tem como objetivo descrever as várias funcionalidades que cada tipo de utilizador consegue exercer tanto na *API* como na *APP*.

*Api:*

- Cliente - poderá utilizar todas as funcionalidades da *API*.
- Moderador - modera todos os clientes da *API* bem como o funcionamento da mesma.

*App:*

- Cliente - poderá criar e editar o seu perfil, pesquisar e comprar produtos tal como cancelar a sua compra, tendo também a possibilidade de pedir reembolso caso tenha ocorrido qualquer problema com a transação.
- Vendedor - tem todas as funcionalidades do cliente mais a possibilidade de criar e editar o seu perfil de vendedor assim como adicionar produtos.
- Moderador - modera todos os clientes da *APP* bem como o funcionamento da mesma, tendo também a função da monitorização dos pedidos de reembolso.

### 2.2 Processo de criação de uma transação

O processo de criação começa com um pedido ao *Api* para a criação da mesma, este pedido deve ter os detalhes necessários da troca (valor da troca, destino e data-limite). Este pedido retorna o *id* da *Exchange* gerada, que irá ser utilizado para transferir o valor referido. Podemos pensar neste processo da mesma maneira que atualmente utilizamos referencias multibanco para pagamentos online, escolhemos os produtos, e pedimos uma referência para onde enviar o valor da compra.

## 2.3 Arquitetura do projeto

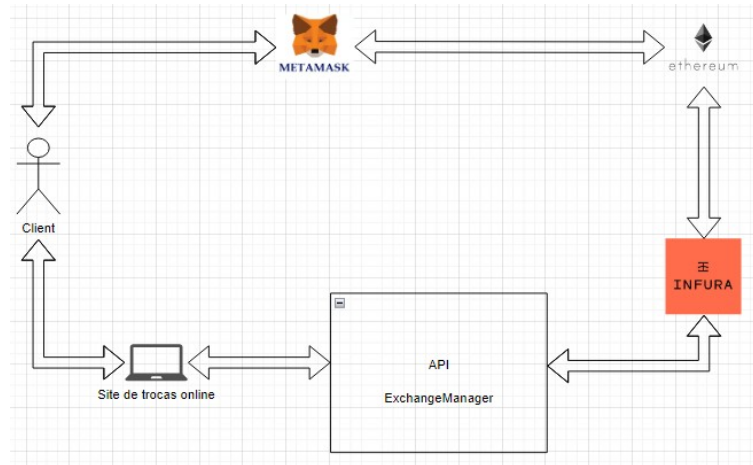


Figura 3: Arquitetura do projeto.

A solução está organizada de maneira a que a plataforma de trocas online nunca tenha de interagir com a rede blockchain, pedindo ao API que faça as operações moderadoras das **exchanges** através de pedidos Http e aos seus utilizadores disponibiliza botões preparados para interagir com a carteira virtual conectada.

### 3 Tecnologias e Conceitos

#### 3.1 *Blockchain*

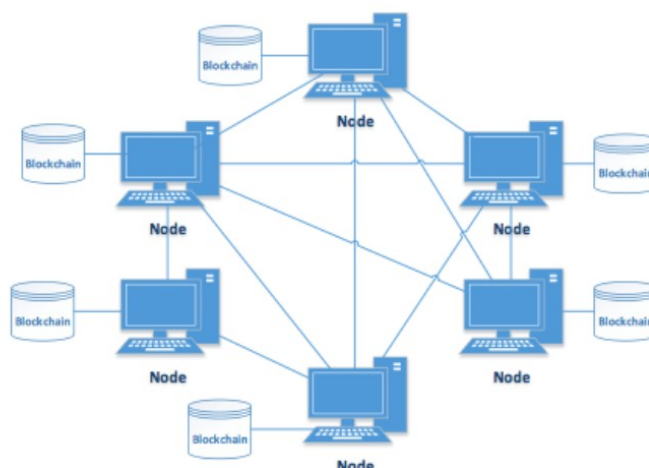


Figura 4: representação de uma rede blockchain

*Blockchain* é uma rede descentralizada que utiliza metodologia ponto a ponto com o objetivo de armazenar todo o histórico de transações feitas nela de um modo seguro e transparente. Isto quer dizer que todas as transações feitas na rede podem ser observadas e verificadas por todos os elementos da mesma.

As máquinas que participam na manutenção da rede são chamados de *Nodes* que têm como função participar nos três principais processos de adicionar uma nova Transação (“Validar transação”, “construir novos Blocos” e “validar novos Blocos”). Quando é iniciado o processo de adicionar novas transações, primeiro verifica-se se a transações que são possíveis (os exemplos mais simples seriam “existe fundos suficientes para esta operação?”, “a taxa que está disposto a pagar é suficiente para a função que pede?”). De seguida ficam à espera que um *Node* aleatoriamente selecionado as introduzida num novo bloco (porção da lista de transações registadas permanentemente na *blockchain*) da *blockchain*. Uma vez que esse bloco é gerado, outros *Nodes* validam o mesmo e partilham com o resto da rede. Apenas um *Node* gera um Bloco no final de certo intervalo de tempo. Após o bloco ter sido gerado e adicionado à *blockchain* é criado um *hash*(chave encriptada de identificação) para a transação que se encontra alocada no bloco, tendo também adicionado o *hash* identificador do bloco anterior da cadeia sendo a *blockchain* uma cadeia de ordem cronológica. Também inclui a data e o momento que foi assinado além da quantidade de transações inclusas no bloco presente.

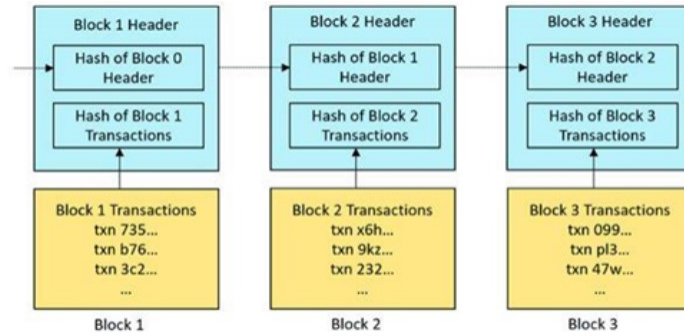


Figura 5: representação da ligação entre blocos da blockchain

### 3.2 Rede Kovan Testnet

Kovan testnet é uma rede *blockchain* de *Ethereum* desenvolvida para simular e facilitar o desenvolvimento de aplicação na *blockchain*, o uso desta rede permite que se possa testar todas as funcionalidades da aplicação e pagar os seus custos com oferecidos pela rede.

### 3.3 Rede Goerli

Tal como a rede Kovan é uma rede de *blockchain* de *Ethereum* com todas as funcionalidades da mesma.

### 3.4 Infura.io

Como a *blockchain* é descentralizada, não existe um *end-point* permanente para aceder às mesmas. Para estabelecer uma ligação com a *blockchain* é necessário contactar um *node* pertencente á rede de *nodes* que constitui a mesma, seja para conectarmos o nosso próprio *node* ou simplesmente para executar operações na *blockchain*.

Infura é uma empresa que disponibiliza diversas ferramentas que facilita o teste e implantação de aplicação na *blockchain*. No âmbito deste projeto vai ser utilizada para fornecer um *end-point* de acesso à rede *blockchain* kovan-testnet.

Uma vez que a rede Kovan terminou operações no dia 1 de Setembro de 2022, foi necessário alterar o projeto para a rede *blockchain* Goerli.

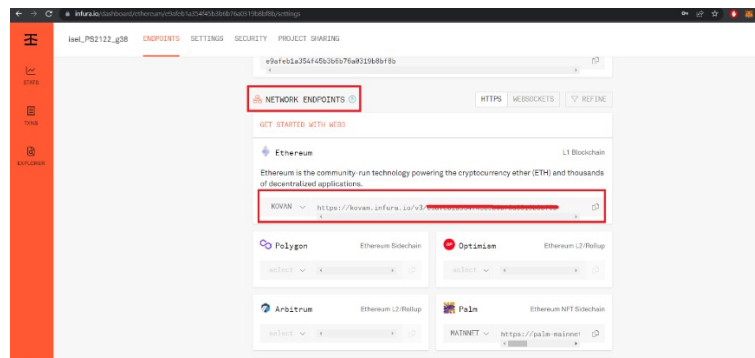


Figura 6: captura de uma pagina de infura a disponibilizar um endpoint

### 3.5 Solidity

Solidity é uma linguagem de programação orientada a objetos, desenvolvida para correr na máquina virtual *EVM* (*Ethereum Virtual Machine*) pela equipa de contribuidores de Ethereum, onde se destacam Christian Reitwiessner e Alex Beregszaszi como principais contribuidores para o desenvolvimento desta linguagem. Desenvolvida e utilizada para a programação de *smart contracts* em redes *blockchain*.

### 3.6 Smart Contracts

```
1 // SPDX-License-Identifier: GPL-3.0
2
3 pragma solidity >=0.7.0 <0.9.0;
4
5 /**
6  * @title Storage
7  * @dev Store & retrieve value in a variable
8  * @custom:dev-run-script ./scripts/deploy_with_ethers.ts
9  */
10 contract Storage {
11
12     uint256 number;
13
14     /**
15      * @dev Store value in variable
16      * @param num value to store
17      */
18     function store(uint256 num) public { 20425 gas
19         number = num;
20     }
21
22     /**
23      * @dev Return value
24      * @return value of 'number'
25      */
26     function retrieve() public view returns (uint256){ 520 gas
27         return number;
28     }
29 }
```

Figura 7: Exemplo de um simples smartcontract

*Smart contracts* são contratos auto-executáveis programados, no âmbito deste projeto o contrato foi programado utilizando linguagem solidity para reagir a transações feitas com eles em *blockchains*. Anteriormente as transações baseavam-se no envio de valor monetário de uma entidade para outra, *smart contracts* reagem a transações mais detalhadas, estas não só podem enviar valor monetários, mas também dados que serão utilizados pelo contrato incluindo os métodos a ser chamados, os parâmetros do método, valor, entre outros.



### 3.7 Remix

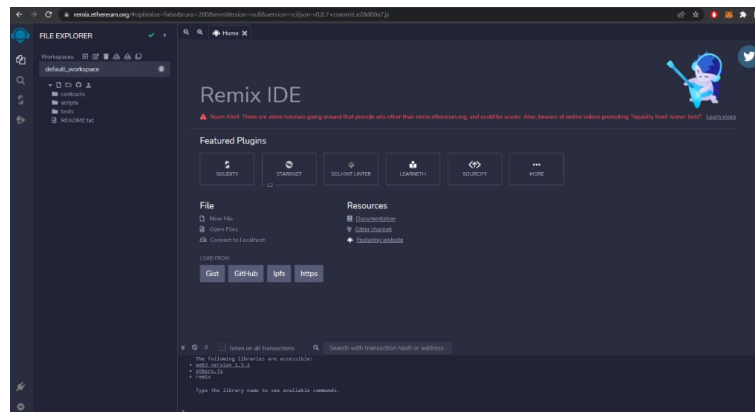


Figura 8: Pagina principal do IDE remix

Remix é um IDE utilizado para desenvolvimento de aplicações em solidity, com um conjunto de ferramentas para a criação de smart contracts e o seu deployment. Oferecendo também instâncias de máquinas virtuais Ethereum locais e ferramentas debug.

### 3.8 Web3j

Web3j é uma biblioteca desenvolvida para interagir com smart contracts de blockchains Ethereum. Disponibiliza a capacidade de não ter de integrar o código do nosso projeto para a plataforma. Apenas necessitamos de inserir o nosso smart-contract e a biblioteca a classe java para conseguirmos executar as suas funcionalidades.

### 3.9 Metamask

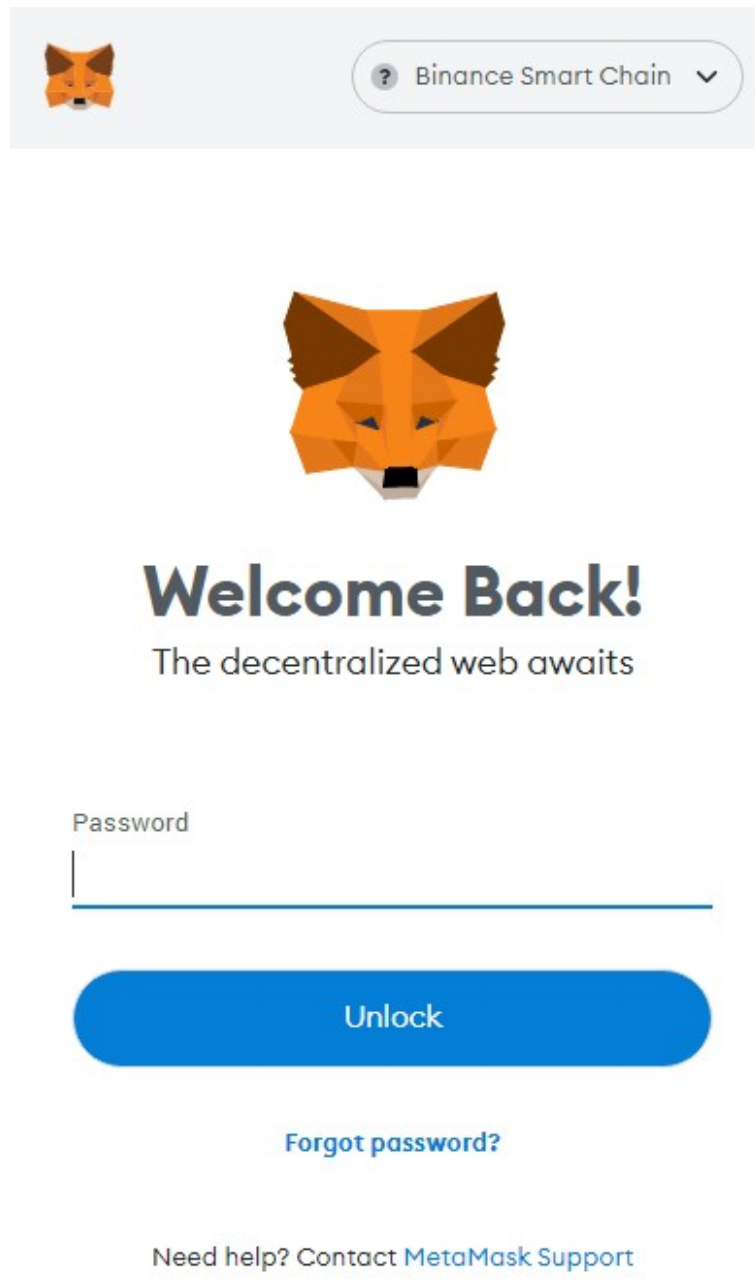


Figura 9: Pagina de login da extensão da carteira virtual Metamask

Metamask é uma carteira de cripto-moedas desenvolvida por ConsenSys Software Inc que permite aos utilizadores interagir com blockchain, para transferir fundos e até mesmo para interagir com smart contracts através de uma extensão para o browser ou aplicação movel.

## 4 API

Nesta secção irá ser descrita a estrutura e funcionalidades API criada para a realização desta proposta de projeto. Na subsecção 3.2.1 será descrita a base de dados do sistema, na subsecção 3.2.2 será descrito o smart contract utilizado pela API, por fim, a subsecção 3.2.3 e 3.2.4 são dedicadas á descrição do processo de autenticação.

### 4.1 Base de Dados

Como os dados da Exchanges geradas pelo Api serão armazenadas na blockchain, a base de dados usada pela Api é apenas usada para a funcionalidade adicional de verificação de clientes e validação dos pedidos. Esta funcionalidade requer que o cliente crie uma conta, e requirite um token para que todos os pedidos à Api possam ser processados. Baseia-se em três tabelas que armazenam os clientes, as credenciais dos mesmos e a lista de tokens de cada cliente.

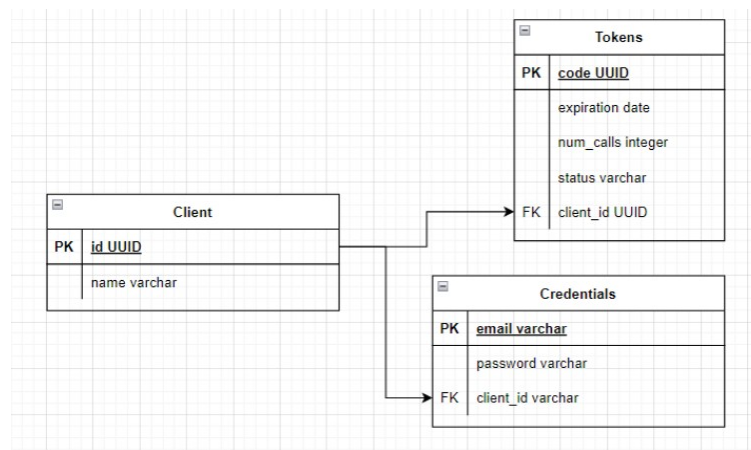


Figura 10: Arquitetura da base de dados da API

- *Cliente* - usada para armazenar a informação mais basica do cliente, neste caso, apenas o nome do mesmo.
- *Credentials* - modera todos os clientes da *API* bem como o funcionamento da mesma.
- *tokens*- usados para validar todos os pedidos que interajam com a blockchain.
  - *Code* -codigo a ser enviado no header do pedido ao Api.
  - *Expiration* - data em que o token vais expirar
  - *Status*-estado do token, usado para sinalizar se o token está num estado ativo ou expirado/eliminado.
  - *Client\_id* - id do cliente dono do token.

### 4.2 Contrato

O smart contract desenvolvido pode ser dividido em duas partes. A parte que só pode ser acedida pelo Api (gerar nova Exchange, alterar a flag de reembolso numa Exchange) e as funcionalidades publicas (pagar, coletar reembolso e coletar da entidade destino).

```

modifier onlyBy(address _account) {
    require(
        msg.sender == _account,
        "Unauthorized user."
    );
    _;
}

```

Figura 11: Modificador utilizado no projecto

Solidity já oferece estruturas de dados e ferramentas que facilitam a implementação de funcionalidades. O mais relevante ou diferente de outras linguagens conhecidas pelo grupo são os modificadores (modifiers), que podem ser descritos como métodos.

Neste caso, o modificador é usado para garantir que o método apenas pode ser chamado pelo “owner”, uma variável local que foi inicializada com a address da conta usada para meter o contrato na blockchain.

```

function refund(uint _id) public onlyBy(owner) {
    // ...
}

```

Figura 12: Exemplo de uma utilização do modificador

#### 4.2.1 Estrutura de uma Exchange

```

struct Exchange {
    uint price;
    address payable origin ;
    address payable destination;
    uint end_date;
    bool isPayed;
    bool refundable;
    bool completed;
}

```

Figura 13: Estrutura Exchange em solidity

A estrutura *Exchange* terá as seguintes propriedades:

- price: valor da exchange.
- origin: address da carteira que efetuou o pagamento.

- destination: address do vendedor para onde o montante será transferido.
- end\_date: data em é permitido emitir reembolso.
- isPayed: flag que indica que o montante já foi pago.
- refundable: flag que identifica se a exchange é apta para reembolso.
- completed: flag que indica se a exchange está finalizada

Cada exchange será armazenada num mapa que terá uma chave do tipo (uint) para a conseguir obter

```
mapping(uint => Exchange) public exchanges;
```

Figura 14: definição de uma estrutura map em solidity

#### 4.2.2 Métodos do contrato

- newExchange: Método que cria uma nova `Exchange`. O método `newExchange` simplesmente verifica se a entidade evocadora é o "owner" e inicializa uma nova exchange com os parâmetros:
  - `_price`: valor da transação.
  - `_destination`: a entidade destino para onde o montante deverá ser enviado.
  - `_end_date`: data a partir do qual a recolha dos fundos será possível.

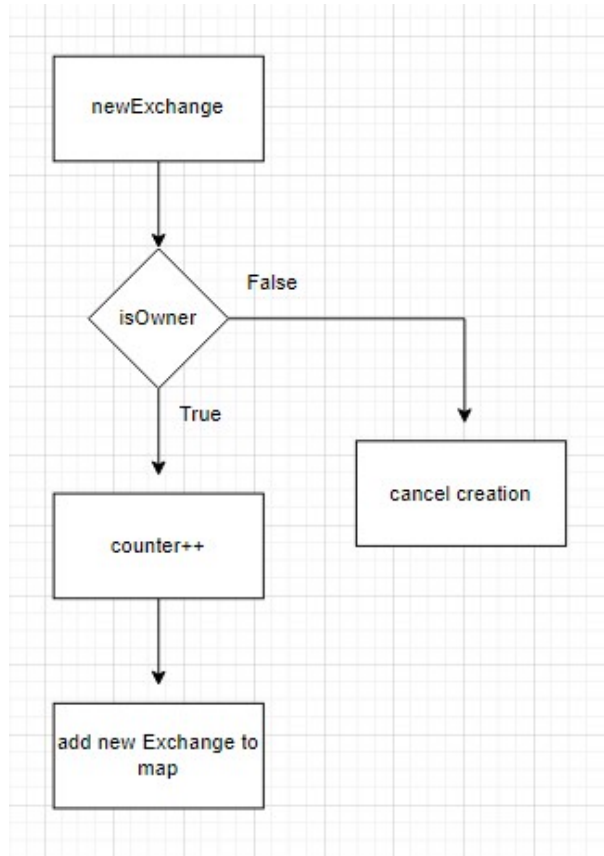


Figura 15: Lógica do método `newExchange`

- pay: Método para pagamento de uma **Exchange**. O método pay apenas recebe o id da exchange como parâmetro sendo que o valor é uma propriedade da Transação em si e não do método chamado. verifica se o valor enviado é o valor correto e que ainda não foi paga.
  - `_id`: identificador da exchange a ser paga.

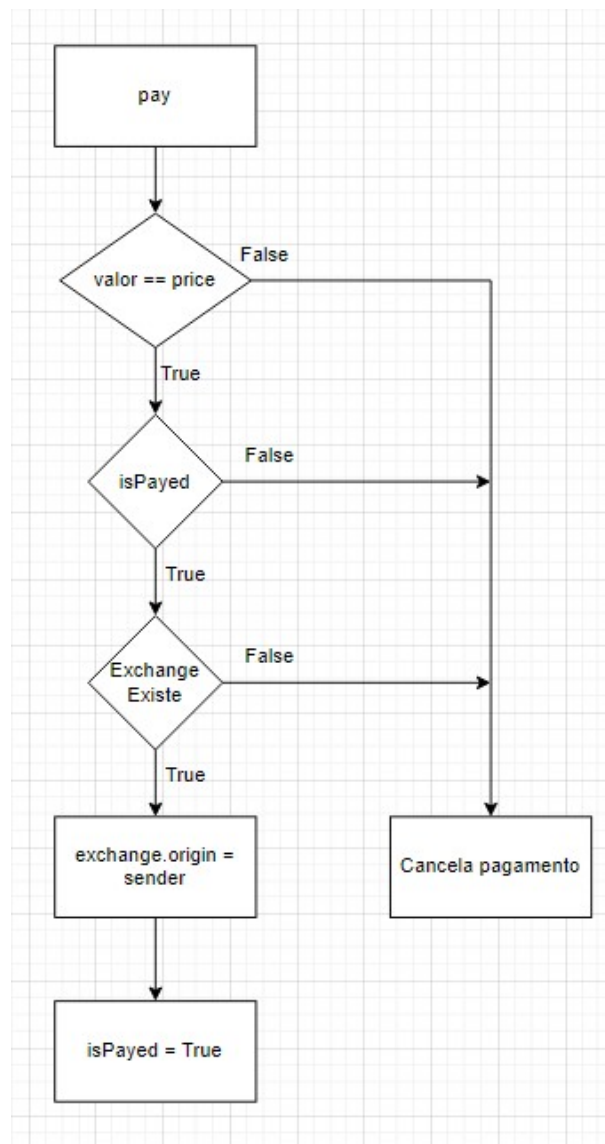


Figura 16: Lógica do método `pay`.

- refund: Metodo para validar o pedido de reembolso de uma **Exchange**. Como no método newExchange, primeiramente é verificado se a entidade evocadora é o "owner" e depois verifica o valor já foi pago e se o montante ainda não foi levantado.
  - \_id: identificador da exchange.

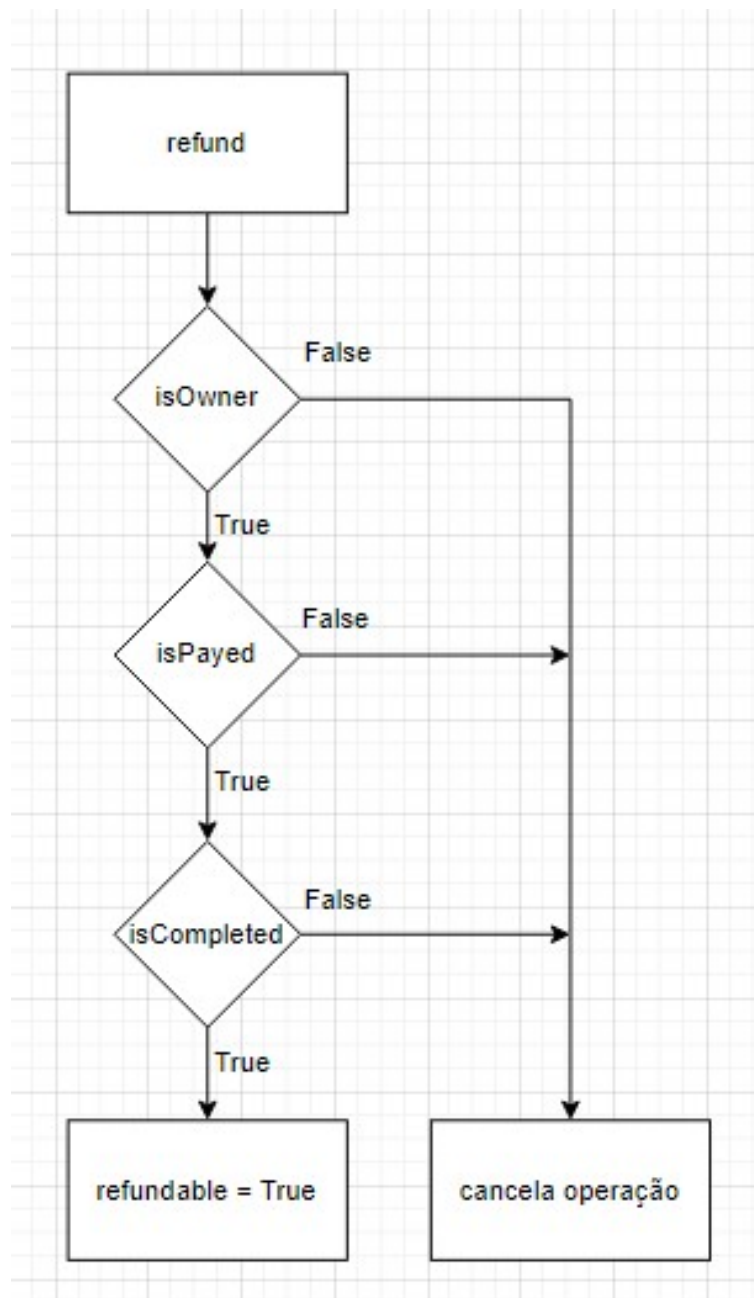


Figura 17: Lógica do método refund.



- collectRefund: Método para levantar o montante de uma **Exhcange** em caso de reembolso. Verifica se a entidade evocadora é a que está registada na exchange ( propriedade origin) e verifica se o montante já foi pago, se foi validado para reembolso e que ainda não foi levantado o montante.
  - \_id: identificador da exchange.

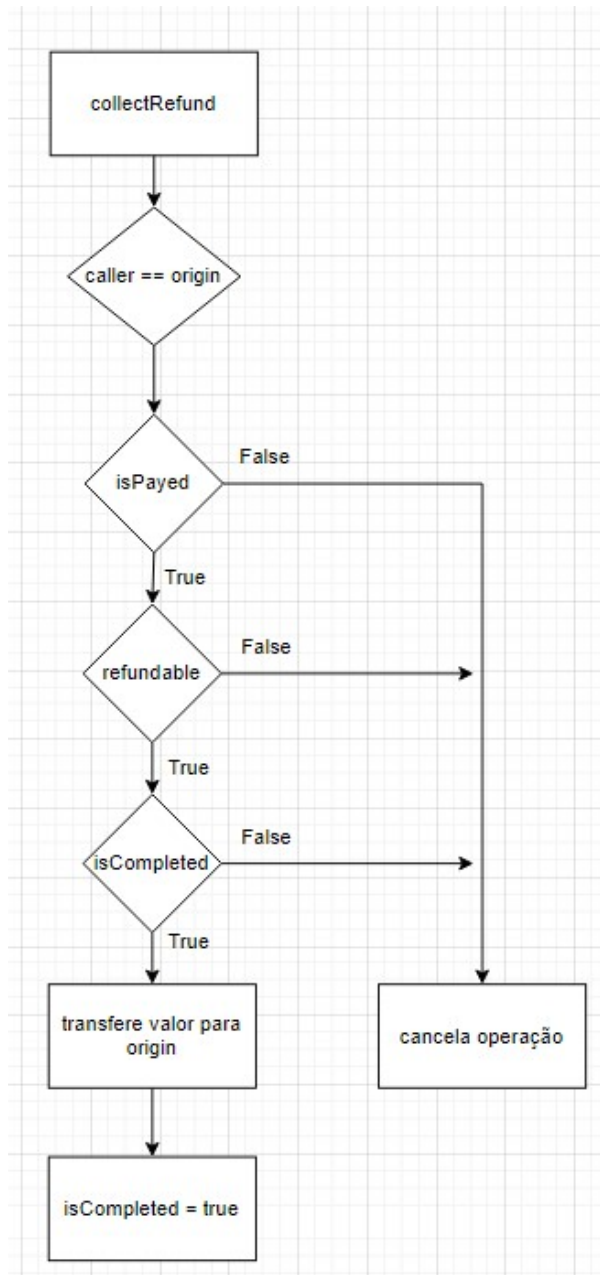


Figura 18: Logica do metodo collectRefund.

- collect: Metodo para levantar o montante de uma **Exhcange** em em casos sem complicações. Verifica se a entidade evocadora é a entidade registada como destino na exchange (propriedade destination) e de seguida verifica se todos os requisitos para levantamento são validos
  - \_id: identificador da exchange a ser paga.

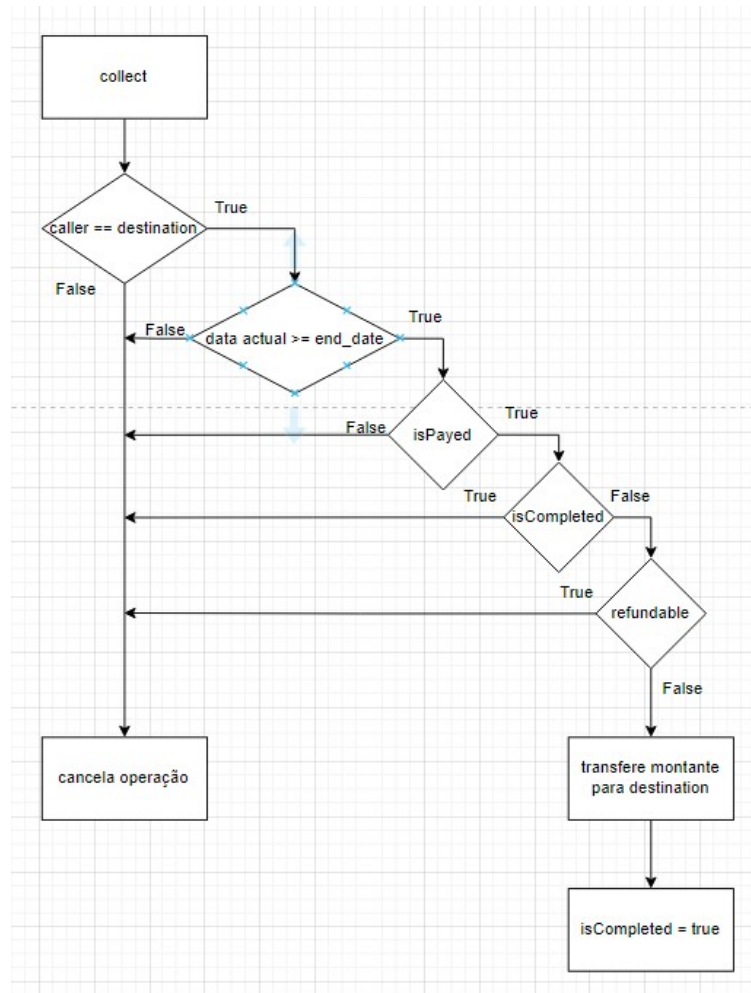


Figura 19: Logica do metodo collect.

### 4.3 Autenticação

Para a autenticação do utilizador utilizamos a tabela “credentials” da base dados para poder verificar que os dados introduzidos na página “login” são validos. Caso sejam validos, é adicionado um header para que este possa ser verificado como uma flag no nosso interceptor (AuthInterceptor) indicando que é para adicionar uma nova conexão ao nosso mapa de conexões ativas e adicionar um cookie com o valor da mesma à resposta do pedido. Este cookie será utilizado para validar todos os pedidos em que autenticação seja necessária, sendo que estes estão identificados com a anotação @AllowAnonymous. Caso esse cookie seja invalido, o pedido será bloqueado.

#### 4.3.1 Tokens

Todos os pedidos que têm a anotação `@ValidateToken` são processados pelo interceptor `@ValidateTokenInterceptor`, que verifica se o token enviado no header “token” existe na base de dados de facto existe. Todos os pedidos que não passem na validação são bloqueados.

#### 4.4 Rotas do API

- Gerar nova Exchange:
  - Metodo: PUT
  - URL: `/ExchangeManager/new`
  - Body: `{value:_____, destination:_____, expriation_date:_____}`
  - Info do parametros:
    - value: (Long) montante da transacção
    - destination: (String) identificação da carteira destinataria
    - expiration\_date: (Long) data em milésimos de segundos em que a transacção termina
    - resposta: informação detalhada da **Exchange** criada.
- Pedido da informação de uma Exchange:
  - Metodo: GET
  - URL: `/ExchangeManager/exchange/{id}/info`
  - Info do parametros:
    - id: (String) identificador da **Exchange**
  - resposta: informação detalhada da **Exchange** criada.
- Pedido para iniciar processo de Reembolso
  - Metodo: PUT
  - URL: `/ExchangeManager/exchange/{id}/refund`
  - Info do parametros:
    - id: (String) identificador da **Exchange**
  - resposta: status do pedido.

## 5 APP

### 5.1 Web API

Nesta secção irá ser descrita a estrutura e funcionalidades da Web API utilizada pela APP.

Esta aplicação tem como base a tecnologia springboot e a escolha de linguagem programática foi o kotlin.

A Web API desta aplicação está subdividida em várias secções, cada uma com o seu objetivo específico:

- *Config* - tem como objetivo agregar todas as configurações para ser utilizadas pela aplicação, mais concretamente sobre as regras de segurança CORS.
- *Controllers* - esta subsecção tem como objetivo o tratamento de todos os pedidos, tratando de toda a informação oferecida e entregando a mesma ao serviço. Para além desta funcionalidade ainda tem o objetivo de criação de todos os caminhos de acesso da APP.
- *Services* - esta subsecção tem como objetivo conter toda a lógica das funcionalidades da APP servindo-se das funcionalidades dos repositórios.
  - *Auth* - contém toda a lógica de autenticação do usuário.
  - *Database* - contém toda a lógica de funcionalidades que comunicação com a base de dados, dentro desta subsecção temos as *Entity* que contêm todas as entidades que existem na aplicação e *Repository* que contêm todos os repositórios do tipo *JpaRepository* que contêm todas as queries com o objetivo de comunicar com a base de dados.

## 5.2 Base de Dados

Nesta secção iremos descrever a base de dados que está a ser utilizada para agregar toda a informação necessário para o funcionamento da APP, esta base de dados está a utilizar como ferramenta a tecnologia PostgreSQL.

Para a identificação de algumas das entidades alojadas nesta base de dados utiliza-se um identificador do tipo UUID para que seja isto possível é necessário adicionar uma extensão "uuid-ossp".

Esta base de dados tem base em três entidades que representam os três tipos diferentes de utilizadores que podem existir e utilizar esta aplicação

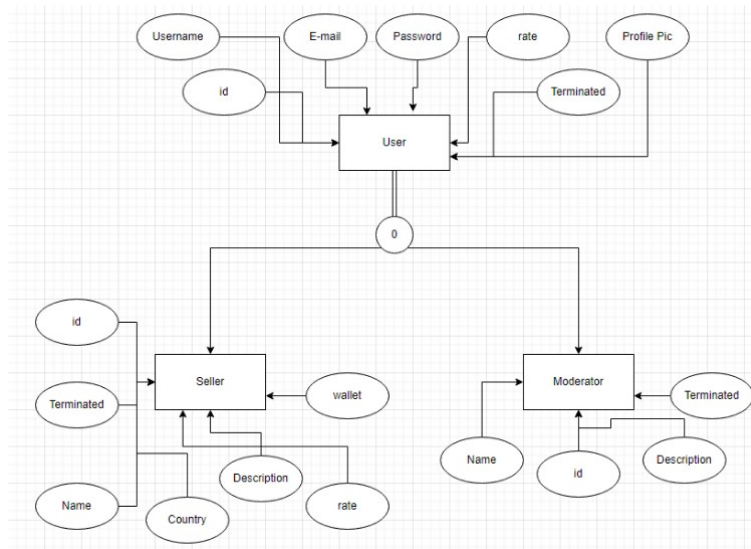


Figura 20: Arquitetura da base de dados da aplicação Web focada a tipos de utilizadores

- *User* - tipo base de utilizador comum.
  - Id: identificador único.
  - Username: nome de utilizador único.
  - Email\_address: email único utilizado para a autenticação e comunicação com o user.
  - Password: password de autenticação para o utilizado.
  - Rate: classificação do utilizador como na APP.
  - Profile\_Picture: url opcional para uma imagem representada no perfil do utilizador.
  - Terminated: boolean que representa se o utilizador foi banido da APP ou não.
- *Seller* - tipo de utilizador com possibilidade de venda de produtos
  - Id: identificador único.
  - Name: nome que será visível nas informações do vendedor.
  - Country: país em que o vendedor se situa.
  - Description: uma pequena descrição opcional que será visível nas informações do vendedor.

- Rate: classificação do vendedor como na APP.
- Wallet: endereço da carteira Metamask em que será enviado as transações monetárias.
- Terminated: boolean que representa se o vendedor foi banido da APP ou não.
- Uid: identificador de relação com a entidade User.
- Moderator: tipo de utilizador com funcionalidades de moderação da APP
  - Id: identificador único.
  - Name: nome que será visível nas informações do moderador.
  - Country: país em que o vendedor se situa.
  - Description: uma pequena descrição opcional que será visível nas informações do moderador.
  - Terminated: boolean que representa se o moderador foi banido da APP ou não.
  - Uid: identificador de relação com a entidade User.

O resto da base de dados é constituída por entidades necessárias para o funcionamento de todas as funcionalidades da APP.

Contendo as seguintes funcionalidades:

- *Product*
  - Id: identificador único.
  - Name: nome do producto.
  - Description: uma pequena descrição opcional do producto .
  - Price: valor pedido para a compra do produto.
  - Sid: identificador de relação com a entidade Seller.
  - Rate: classificação do produto.
- Image
  - Id: identificador único.
  - Path: Url da imagem
  - Pid: identificador de relação com a entidade Product.
- Exchange
  - Id: identificador único.
  - Client\_id: identificador de relação com a entidade User comprador
  - Seller\_id: identificador de relação com a entidade Seller vendedor
  - Pid: identificador de relação com a entidade Product
  - Value: valor monetário da transferência
  - Quantity: quantidade de produto
  - Completed: boolean que representa se a transferência foi completada
  - End\_date: data-limite para que a transferência seja completada
- RefundForm

- Id: identificador único.
  - Client\_id: identificador de relação com a entidade User comprador que requisitou o pedido de reembolso.
  - Exchange\_id: identificador de relação com a entidade Exchange.
  - Description: razão do pedido de reembolso.
- Token
    - Id: identificador único.
    - Uid: identificador de relação com a entidade User.

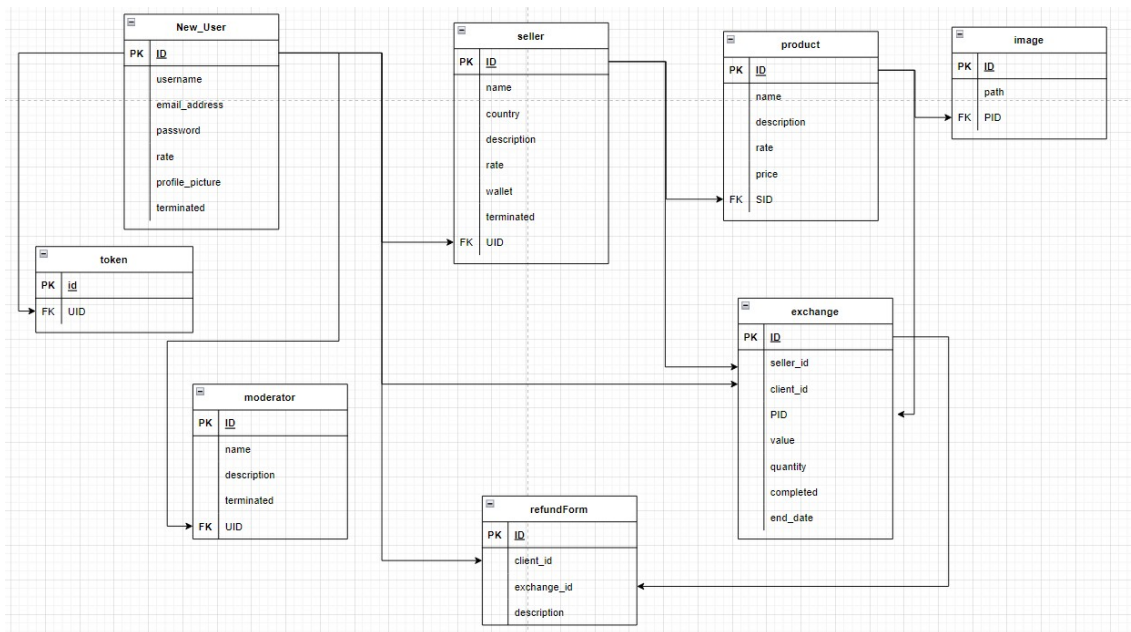


Figura 21: Arquitetura da base de dados da aplicação Web

## 5.3 Aplicação Web

Nesta secção irá ser descrita a estrutura e funcionalidades do Frontend da APP.

Esta aplicação tem como base a tecnologia React e a escolha de linguagem programática foi o Javascript.

### 5.3.1 Navegação Páginas

A aplicação começa com a home page oferece várias funcionalidades uma barra de pesquisa para que o cliente possa pesquisar e encontrar os produtos desejados e uma lista de produtos para que o utilizador tenha uma ideia de produtos que esta aplicação contém. Esta página tem 2 diferentes vistas:

1. Um botão para o cliente conectar a sua carteira Metamask caso não tenha de ter sido conectada e um botão de Login que envia o cliente para a página de login para que possa ser autenticado e utilizar todas as outras funcionalidades da aplicação.

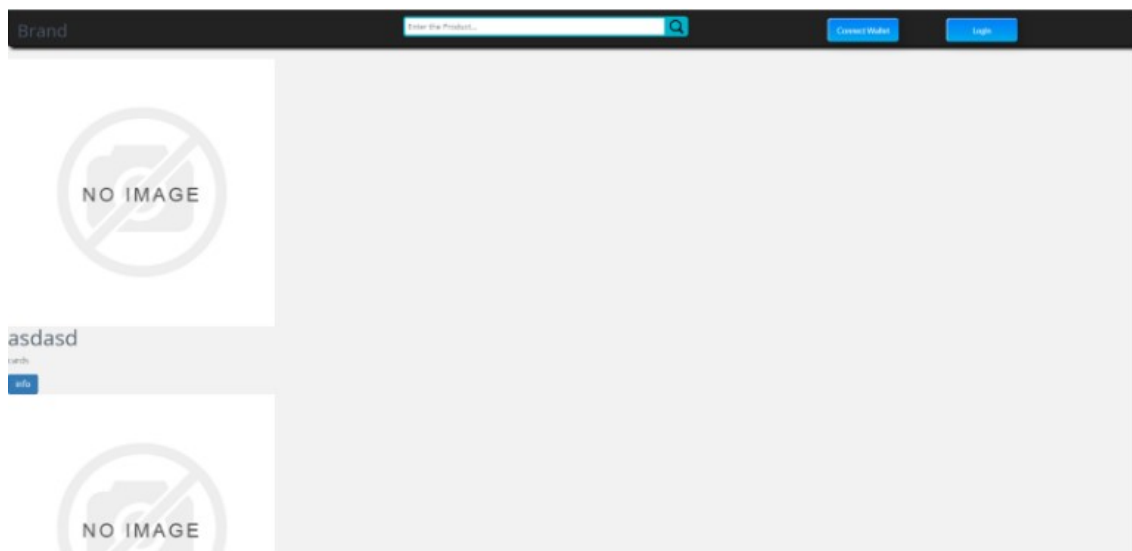


Figura 22: Página Home da aplicação sem Login



2. Assim que o utilizador der Login será apresentado com uma nova home page que substitui o botão de Login por um menu que contém, dependendo do tipo de utilizador, uma hiperligação para o perfil para cada tipo de conta e uma opção de logout.

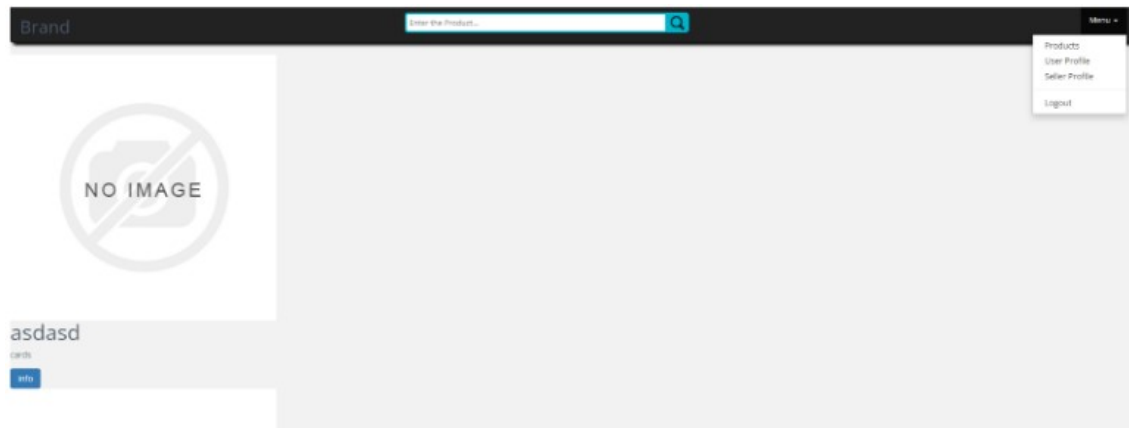


Figura 23: Página Home da aplicação com Login

Tendo o utilizador selecionado a opção login na aplicação irá ser redirecionado para a página do mesmo, em que serão pedidos o email e a password, caso os dados oferecidos pelo utilizador forem aprovados e o mesmo for autenticado irá ser criado um cookie e um token para o utilizador.

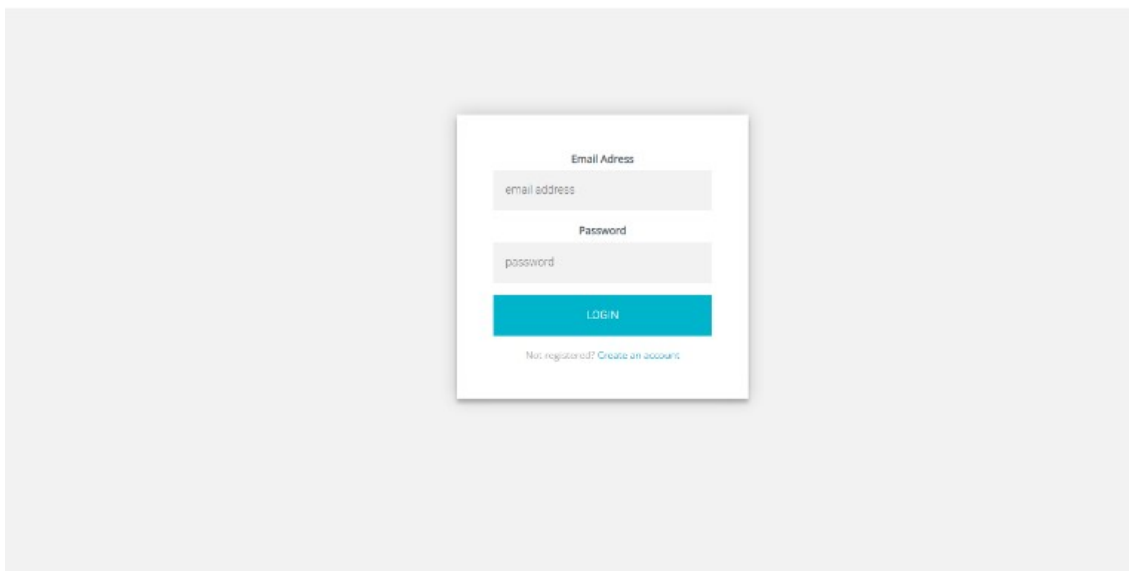


Figura 24: Página de Login da aplicação

Caso o utilizador não tenha conta na aplicação terá a opção de a criar utilizando a opção visível na figura 23 “create an account” que ao ser selecionada ira para a página de criação de conta, nesta página irá ser pedido as credenciais desejadas pelo utilizador.

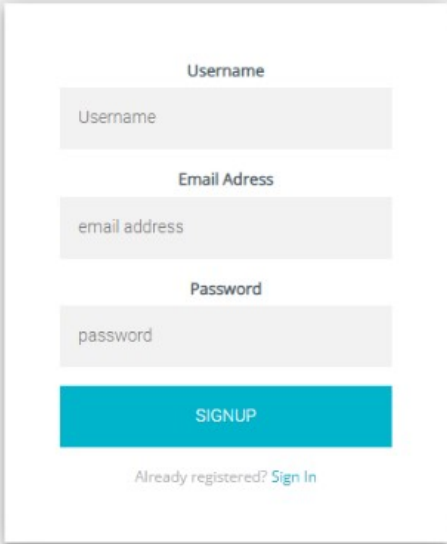
A screenshot of a web application's signup page. The page features a white rectangular form centered on a light gray background. The form contains three input fields, each with a label above it: 'Username' (with placeholder text 'Username'), 'Email Address' (with placeholder text 'email address'), and 'Password' (with placeholder text 'password'). Below these fields is a prominent blue button labeled 'SIGNUP'. At the bottom of the form, there is a link that reads 'Already registered? Sign In'.

Figura 25: Página de Signup da aplicação

Tendo o utilizador ter sido autenticado tem acesso a todas as funcionalidades da aplicação iremos por começar a descrever as páginas de perfil de cada tipo de utilizador

User

A página de perfil de utilizador apresenta o seu historial de transações e uma opção para verificar detalhes de conta.

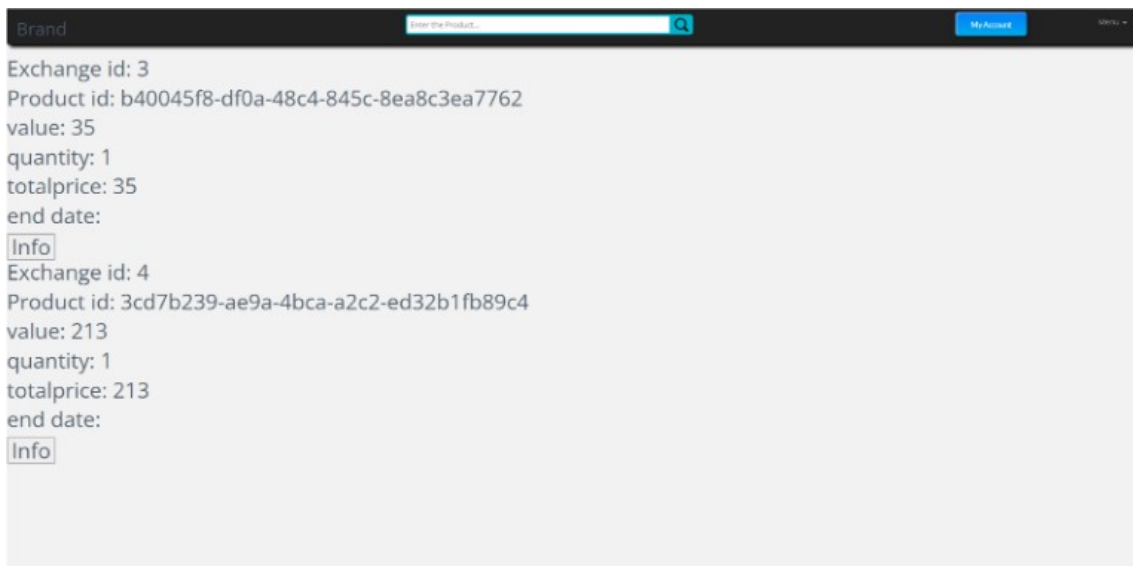


Figura 26: Página de Perfil do utilizador

Caso o utilizador queira interagir ou obter informações sobre cada transação terá essa opção com o botão “info”.

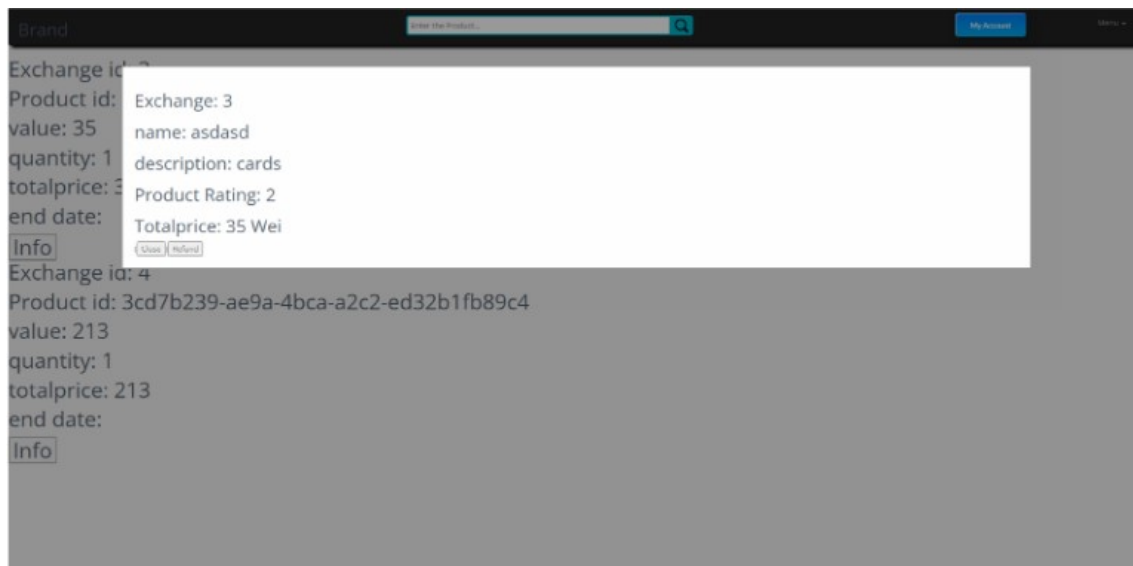


Figura 27: Janela de informação de um produto

Dentro desta janela o utilizador será apresentado com todas as informações da transação e no final irá ser apresentado com três opções caso a transação não tenha sido concluída as três opções serão as apresentadas:

1.Existência de um botão “refund” caso a transação foi paga e o utilizador queira pedir um reembolso esta opção abre outra janela em que o utilizador pode descrever a razão para o reembolso para que a mesma seja adicionada ao pedido de reembolso que será entregue e analisado pelos moderadores.

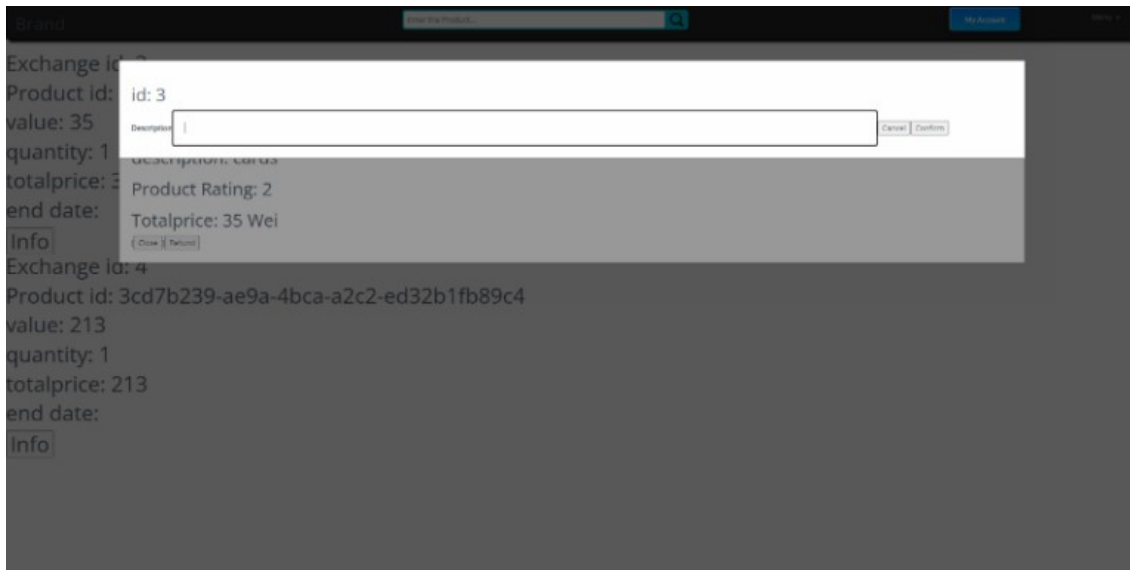


Figura 28: Janela de pedido de reembolso

2. Existência de um botão “pay” ou “cancel” caso a transação ainda não ter sido paga, caso o utilizador escolha a opção pay ira ser enviado para a página de pagamento do produto para que a transação seja validada, caso escolha a opção cancel o utilizador irá cancelar a transação.

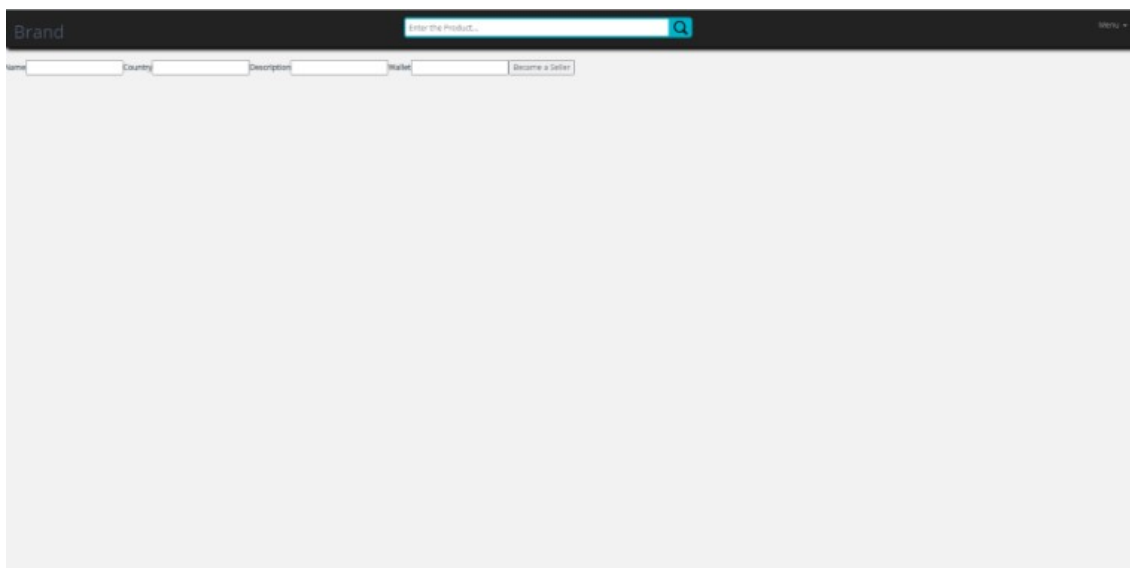
3.Existência de um botão “collect” caso o pedido de reembolso tenha sido aprovado e o utilizador poderá receber o total valor pago na transação.

Por fim esta página ainda dá a opção de caso o utilizador não ser ainda um vendedor e queira se tornar um basta selecionar o botão “became a seller”.



Figura 29: Página com botão para que o utilizador se torne um vendedor

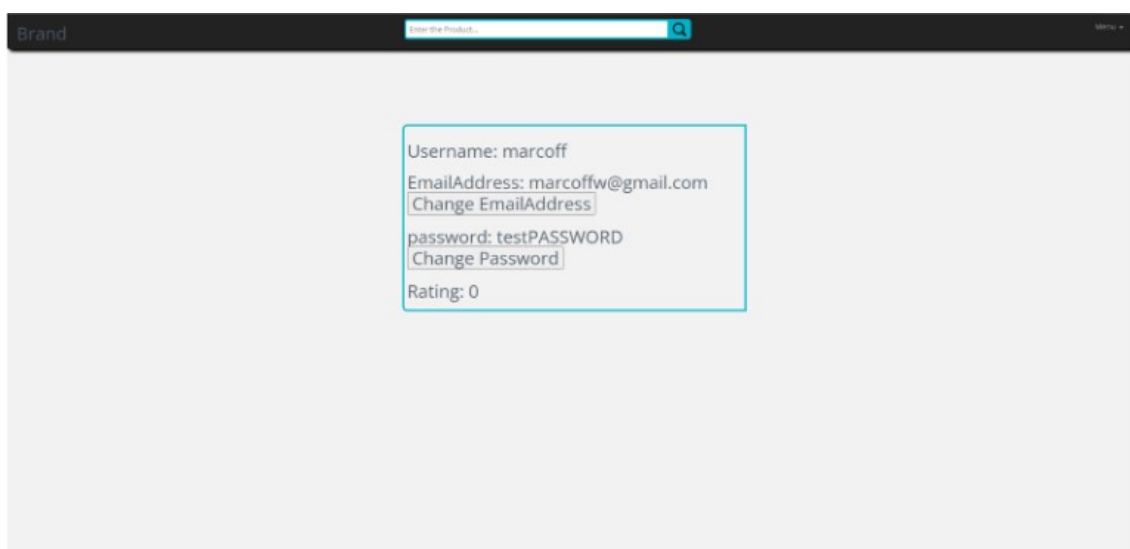
Assim que o botão for selecionado a página para a criação de um novo vendedor, irá ser pedido mais informação sobre o utilizador, mas a principal informação que o utilizador terá de oferecer será a wallet do utilizador esta será o address da carteira Metamask que o vendedor irá receber o valor monetário de todas as suas transações.



The screenshot shows a web interface titled 'Brand' in the top left corner. A search bar with the placeholder text 'Enter the Product...' and a magnifying glass icon is located in the top right. Below the search bar, there is a horizontal row of input fields labeled 'Name', 'Country', 'Description', and 'Wallet', followed by a button labeled 'Become a Seller'.

Figura 30: Página de criação de um vendedor

Caso o utilizador queira ver e editar as suas credenciais de utilizador basta selecionar o botão “my account” .



The screenshot shows a web interface titled 'Brand' in the top left corner. A search bar with the placeholder text 'Enter the Product...' and a magnifying glass icon is located in the top right. In the center of the page, there is a box containing the following information: 'Username: marcoff', 'EmailAddress: marcoffw@gmail.com' with a 'Change EmailAddress' button, 'password: testPASSWORD' with a 'Change Password' button, and 'Rating: 0'.

Figura 31: Página de alteração de credenciais

## Seller

A página de perfil do vendedor apresenta todas as vendas do utilizador e um botão de “collect” que caso a transação foi concluída com sucesso dará a possibilidade de levantar o montante monetário para a carteira correspondente à conta do utilizador.

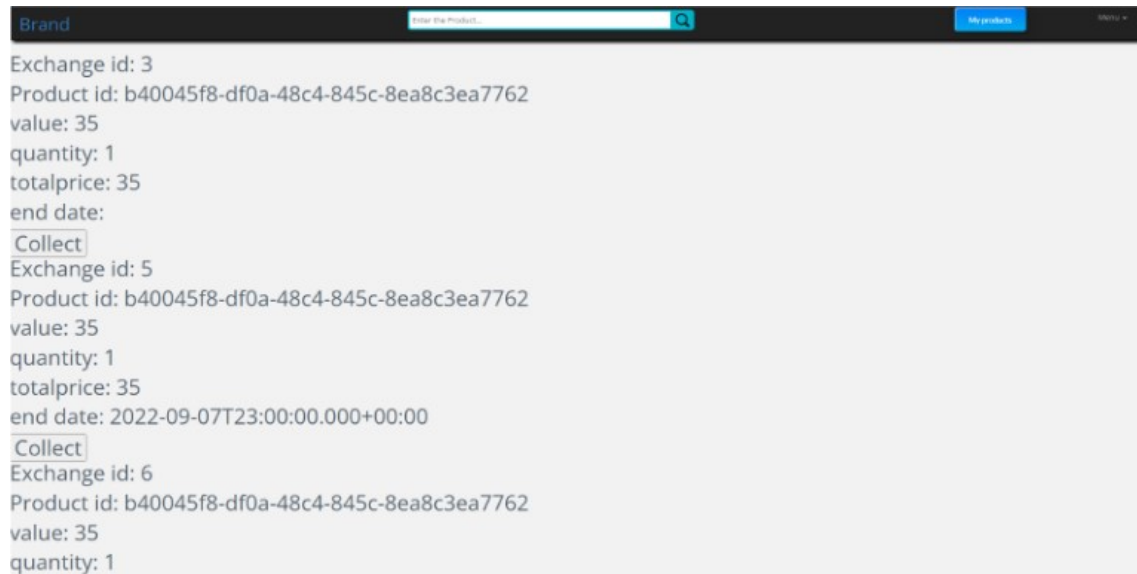


Figura 32: Página de perfil do vendedor

Podendo também ter a possibilidade de adicionar e remover produtos selecionando o botão “my products”, assim que for selecionado o utilizador é enviado para a página de produtos.

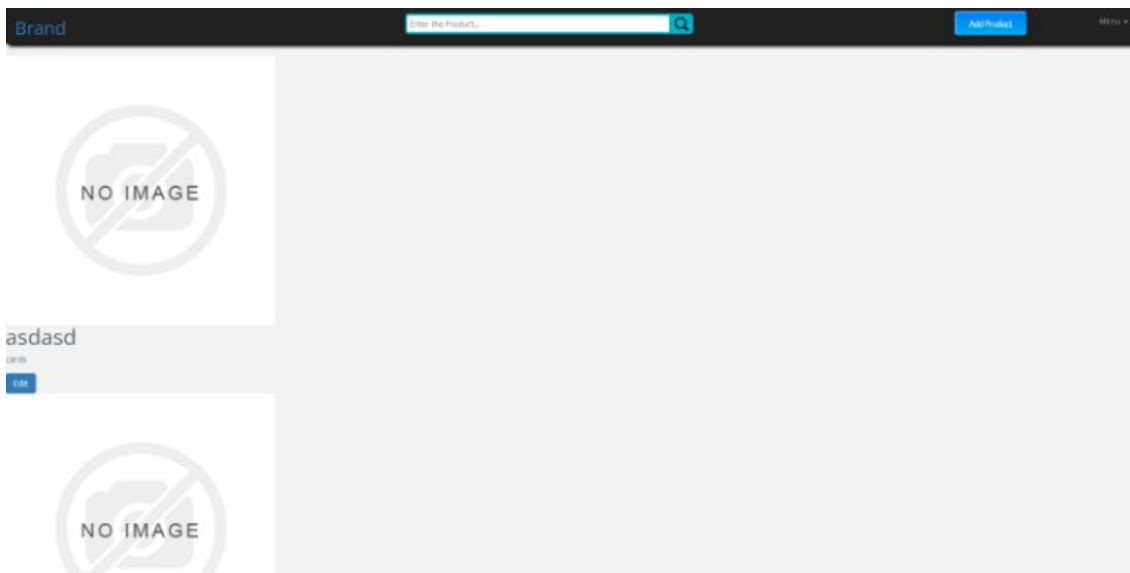


Figura 33: Página de produtos do vendedor

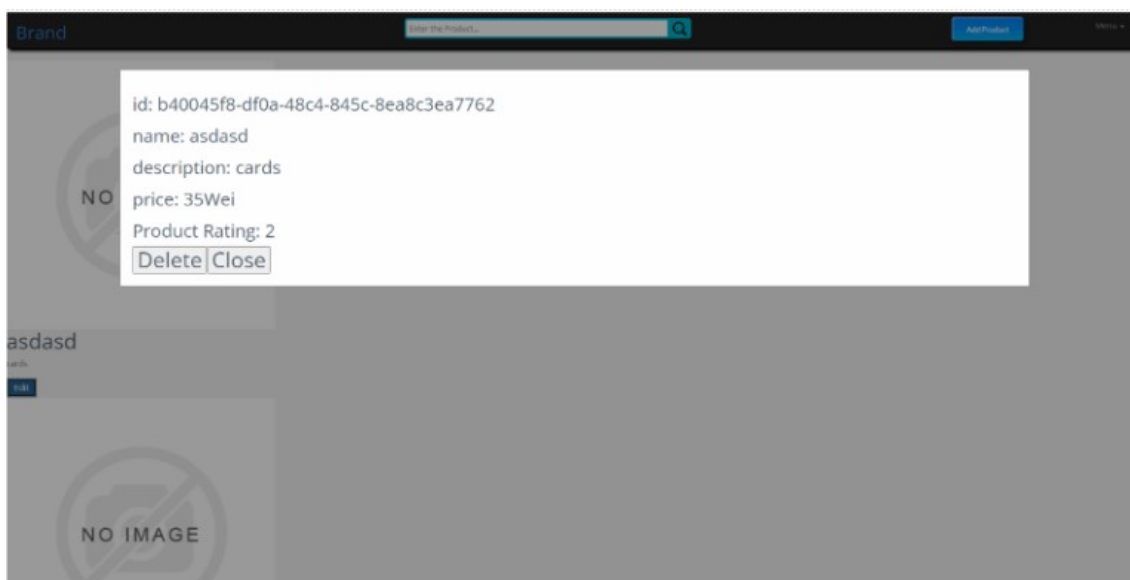


Figura 34: Janela de informação de um produto do vendedor



Figura 35: Página de criação de um produto

### Moderador

A página de perfil de moderador apenas serve para dar acesso à página de tratamento de pedidos de reembolso em que o moderador irá analisar o pedido e caso seja uma justificativa aceitável aos olhos do moderador será aprovada e o utilizador terá direito ao seu reembolso caso o contrário o pedido é removido da lista de pedidos. O moderador pode também caso necessário pesquisar por um pedido específico.



Figura 36: Página de validação de pedidos de reembolso



Posto as páginas de utilizador irá ser descrito todo o percurso que irá ter de percorrer para a compra de um produto, depois de ter sido autenticado.

Em primeiro lugar o utilizador seleciona um produto depois de o selecionar irá ser apresentada a página do produto e todas as informações do mesmo. Nesta página o utilizador terá acesso as informações do vendedor, como também pode escolher a quantidade do produto que deseja comprar.

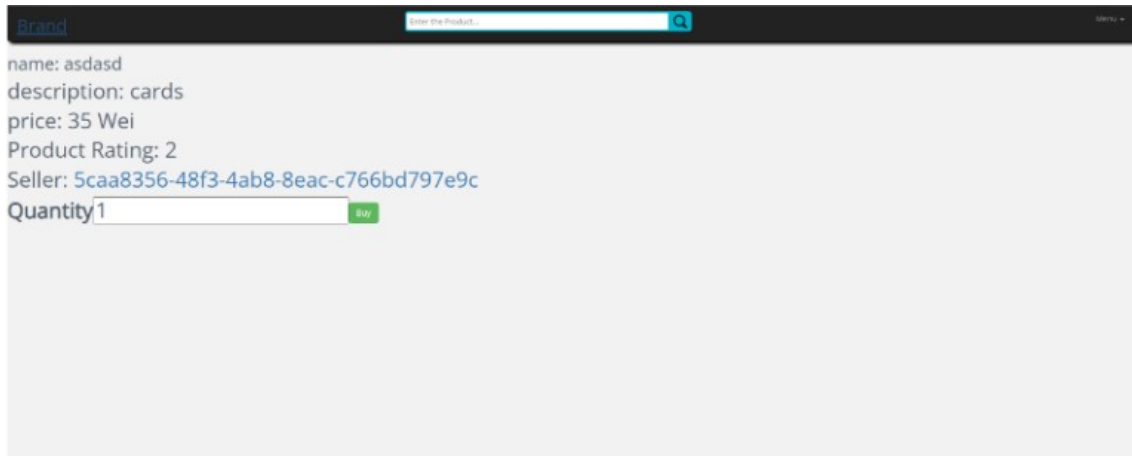


Figura 37: Página de compra de um produto

Depois de o botão “buy” o utilizador será apresentado com uma janela de confirmação do pedido de compra.

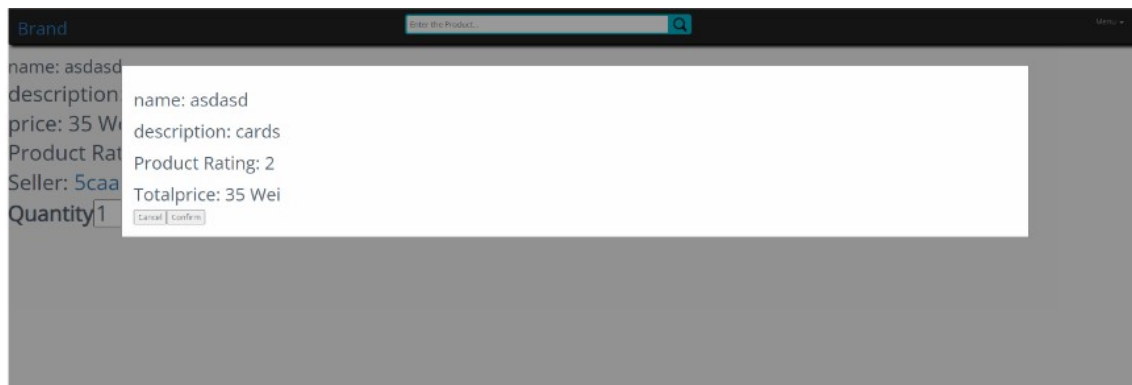


Figura 38: Janela de confirmação do pedido de compra

Caso o utilizador confirme o pedido ira ser apresentada a janela de conclusão de compra apresentado também os dados da blockchain caso o utilizador queira acompanhar e pesquisar a sua transação.

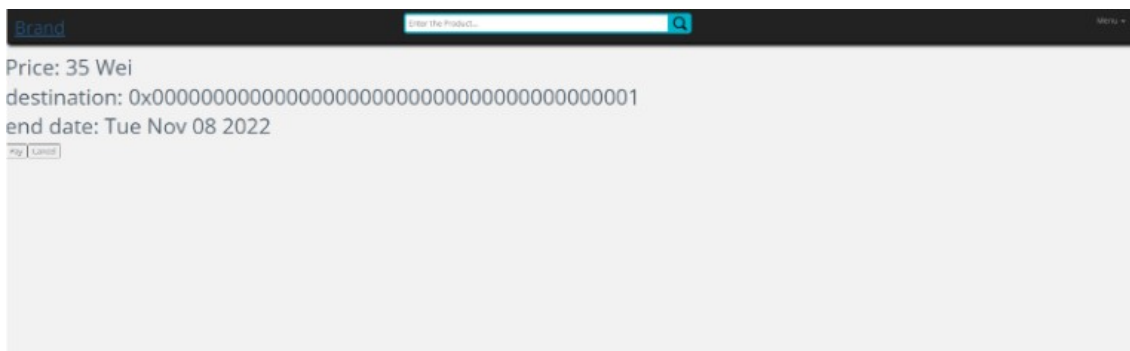


Figura 39: Página de pagamento

Por fim se o utilizador escolher pagar e concluir o pedido de compra será apresentado uma janela Metamask com a confirmação do pagamento.

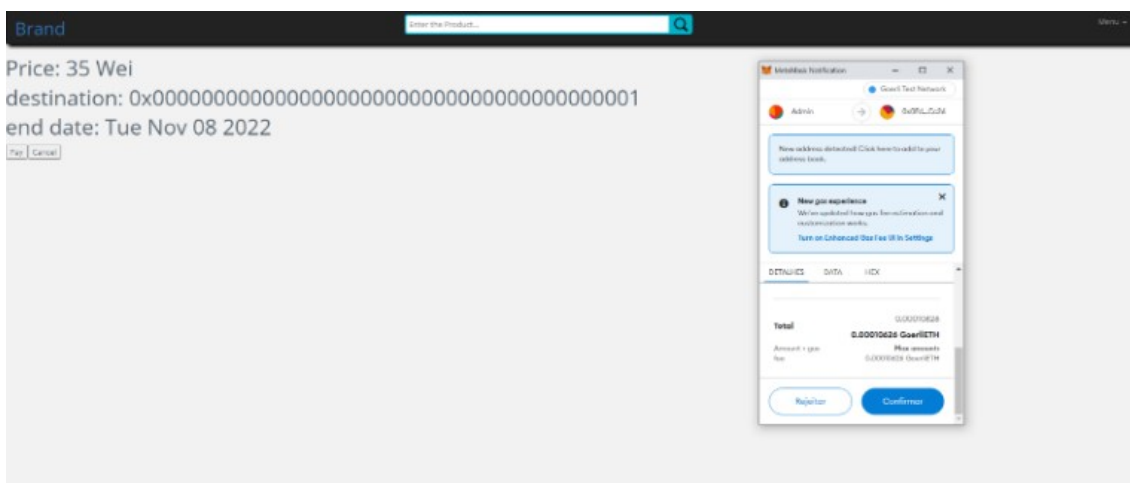


Figura 40: Janela de Metamask para pagamento

## 6 Conclusão

Através da realização deste projeto podemos dizer que o objetivo principal foi cumprido, sendo este, criar um sistema de transações numa rede blockchain e ao mesmo tempo aprender mais sobre esta tecnologia e os seus desafios. Ao longo do projeto tivemos diversas fases de pesquisa e aprendizagem para a realização do nosso projeto. Como bibliotecas existentes, projetos open source que possamos utilizar, frameworks disponíveis. E sempre que descobríamos algo novo, tanto como uma biblioteca mais apropriada para nós ou que as bibliotecas que estávamos a usar foi descontinuada, tínhamos de parar o desenvolvimento e procurar algo novo para a substituir ou reestruturar a nossa aplicação. Tivemos de estar constantemente a pesquisar e aprender novas maneiras de enfrentar um problema, onde muitas vezes, identifica-lo já era um desafio em si. Foi necessário aprender varias linguagens e como lançar a nossa aplicação para uma rede blockchain de testes, como também tivemos de aprender a pedir fundos para poder trabalhar nessa rede. Resumidamente, o nosso grupo não tinha noção do quão difícil este projeto seria e devido à constante reestruturação do projeto, havia sempre pequenas falhas na comunicação em relação às alterações feitas, mas temos orgulho do que produzimos e sentimos que aprendemos bastante ao longo deste projeto. Surgindo mesmo no fim do projeto um problema adicional, a rede de testes que estávamos a utilizar terminou o seu funcionamento no dia 1 de Setembro e no mesmo dia conseguimos lançar a nossa aplicação numa nova rede ( algo que levou 2 semanas a fazer da primeira vez), e como todas as fases finais de cada projeto, conseguimos identificar certas funcionalidades que podem ser melhoradas.

### 6.1 Trabalho Futuro

Melhorar o frontend da API para gerar novos tokens. Optimizar o contrato para reduzir custos e melhorar algumas das suas funcionalidades. Em termos da APP de exemplo vários melhoramentos podem ser aplicados como a autenticação da aplicação, assim como toda a interface de utilizador.

## 7 Referencias

- [1] Sichel, R. L., Calixto, S. R. (2018). Criptomoedas: impactos na economia global. Perspectivas. Revista de Direito da Cidade, 10(3), 1622-1641.
- [2] Solidity: <https://en.wikipedia.org/wiki/Solidity> consultado a 03/03/2022
- [3] Remix: <https://remix-project.org/> consultado a 03/03/2022
- [4] Web3j: <https://www.web3labs.com/web3j-sdk> consultado a 05/03/2022
- [5] MetaMask: <https://metamask.io/> consultado a 05/03/2022
- [6] React. <https://reactjs.org/>. consultado a 15/03/2022.
- [7] Kovan etherscan: <https://kovan.etherscan.io/> consultado a 05/03/2022
- [8] PostgreSQL. <https://www.postgresql.org/>. consultado a 08/04/2022.
- [9] kovan website: <https://kovan-testnet.github.io/website/> consultado a 10/04/2022.
- [10] Faucets: <https://faucets.chain.link/>
- [11] Kotlin. <https://kotlinlang.org/>. consultado a 11/04/2022.
- [12] Spring. <https://www.tutorialspoint.com/spring>. consultado a 11/04/2022.
- [13] Infura: <https://infura.io/> consultado a 05/05/2022.
- [14] goerli: <https://goerli.net/> consultado a 03/09/2022.

## 8 Apêndices

### 8.1 Smart Contract

```
1 // SPDX-License-Identifier: MIT
2
3 pragma solidity >=0.7.0 <0.9.0;
4
5 contract ExchangeManager{
6
7     modifier onlyBy(address _account) {
8         require(
9             msg.sender == _account,
10             "Unauthorized user."
11         );
12         _;
13     }
14
15     event exchangeCreation(uint exchange_id, uint price, address destination, uint end_date);
16
17     struct Exchange {
18         uint price;
19         address payable origin ;
20         address payable destination;
21         uint end_date;
22         bool isPaid;
23         bool refundable;
24         bool completed;
25     }
26
27     address payable owner;
28
29     uint256 counter = 0;
30
31     mapping(uint => Exchange) public exchanges;
32 }
```

```

32
33 constructor(){ 956240 gas 930000 gas
34     owner = payable(msg.sender);
35 }
36
37 function newExchange(uint _price, address payable _destination, uint _end_date) public onlyBy(owner){ undefined gas
38     counter = counter + 1;
39     exchanges[counter] = Exchange( _price, payable( address(0x0) ), _destination, _end_date, false, false,false);
40     emit exchangeCreation(counter, _price, _destination, _end_date);
41 }
42
43 function pay(uint _id) public payable { 42144 gas
44     require(
45         msg.value == exchanges[_id].price && !exchanges[_id].isPaid,
46         "Incorrect transaction amount."
47     );
48     require(
49         exchanges[_id].destination != address(0x0),
50         "Invalid Exchange id"
51     );
52     exchanges[_id].origin = payable(msg.sender);
53     exchanges[_id].isPaid = true;
54 }
55
56 function refund(uint _id) public onlyBy(owner){ 21797 gas
57     require(
58         exchanges[_id].isPaid && !exchanges[_id].completed,
59         "No funds to return! the exchange might already have been completed or not paid at all."
60     );
61     exchanges[_id].refundable = true;
62 }

```

```

63
64 function collectRefund(uint _id) public onlyBy(exchanges[_id].origin){ infinite gas
65     require(
66         exchanges[_id].isPaid && exchanges[_id].refundable && !exchanges[_id].completed,
67         "Invalid request! (No funds to refund / exchange refund not validated )"
68     );
69     exchanges[_id].origin.transfer(exchanges[_id].price);
70     exchanges[_id].completed = true;
71 }
72
73 function collect(uint _id) public onlyBy(exchanges[_id].destination){ infinite gas
74     require(
75         block.timestamp >= exchanges[_id].end_date && exchanges[_id].isPaid && !exchanges[_id].completed && !exchanges[_id].refundable,
76         "Too soon to collect or funds already collected"
77     );
78     payable(msg.sender).transfer(exchanges[_id].price);
79     exchanges[_id].completed = true;
80 }
81 }

```