UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
CENTRO DE ENGENHARIA ELÉTRICA E INFORMÁTICA
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

CAIO SANCHES BATISTA DE LIRA

UMA INTERFACE WEB PARA UM CONTRATO INTELIGENTE

CAMPINA GRANDE - PB

2022

CAIO SANCHES BATISTA DE LIRA

UMA INTERFACE WEB PARA UM CONTRATO INTELIGENTE

Trabalho de Conclusão Curso apresentado ao Curso Bacharelado em Ciência da Computação do Centro de Engenharia Elétrica e Informática da Universidade Federal de Campina Grande, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

Orientador : Professor Dr. José Antão Beltrão Moura

CAMPINA GRANDE - PB

2022

CAIO SANCHES BATISTA DE LIRA

UMA INTERFACE WEB PARA UM CONTRATO INTELIGENTE

Trabalho de Conclusão Curso apresentado ao Curso Bacharelado em Ciência da Computação do Centro de Engenharia Elétrica e Informática da Universidade Federal de Campina Grande, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

BANCA EXAMINADORA:

Professor Dr. José Antão Beltrão Moura

Orientador – UASC/CEEI/UFCG

Professor Dr. Marcus Salerno de Aquino

Examinador – UASC/CEEI/UFCG

Francisco Vilar Brasileiro

Professor da Disciplina TCC – UASC/CEEI/UFCG

Trabalho aprovado em: 02 de Setembro de 2022.

CAMPINA GRANDE - PB

RESUMO

Contrato inteligente é um conjunto de protocolos de transações, que representam termos reais de negócios. O lançamento da rede Ethereum, possibilitou uma infraestrutura facilitadora para os contratos inteligentes. Apesar da popularização de aplicações descentralizadas, o acesso por parte de usuários comuns ainda é restrito. Um dos limitantes a tal acesso, é a ausência de interfaces facilitadoras que aproximem o usuário com o contexto que já estão familiarizados. Na tentativa de atenuar um dos limitantes ao acesso de aplicações descentralizadas, o presente trabalho implementa uma interface web amigável, personalizada para um contrato inteligente de três-pontas (contratante, validador e beneficiário), disponível na blockchain, que permite a criação e interação com transações validáveis.

Uma interface web para um contrato inteligente

Trabalho de Conclusão de Curso

Caio Sanches Batista de Lira
Universidade Federal de Campina Grande
Campina Grande, Brasil.Paraíba
Graduando
caio.lira@ccc.ufcg.edu.br

Antão B. Moura
Universidade Federal de Campina Grande
Campina Grande, Paraíba
Orientador
antao@computacao.ufcg.edu.br

Tiago L.P. Clementino
Universidade Federal de Campina Grande
Campina Grande, Paraíba
Co-Orientador
tiagolucas@copin.ufcg.edu.br

RESUMO

Contrato inteligente é um conjunto de protocolos de transações, que representam termos reais de negócios. O lançamento da rede Ethereum, possibilitou uma infraestrutura facilitadora para os contratos inteligentes. Apesar da popularização de aplicações descentralizadas, o acesso por parte de usuários comuns ainda é restrito. Um dos limitantes a tal acesso, é a ausência de interfaces facilitadoras que aproximem o usuário com o contexto que já estão familiarizados. Na tentativa de atenuar um dos limitantes ao acesso de aplicações descentralizadas, o presente trabalho implementa uma interface web amigável, personalizada para um contrato inteligente de três-pontas (contratante, validador e beneficiário), disponível na blockchain, que permite a criação e interação com transações validáveis.

Palavras-Chave

Contrato digital, Criptomoeda, Blockchain, Interface Web.

Repositórios

https://github.com/conditional-token/conditional-token-frontend

https://github.com/conditional-token/conditional-token-backend

1. INTRODUÇÃO

Com o advento do Bitcoin, uma moeda digital que permite a realização de transferências entre pessoas de forma eletrônica e direta, sem o intermédio de uma instituição financeira [1], o mercado de criptomoedas ficou aquecido, e novas moedas digitais se tornaram populares, entre elas o Ethereum, Monero e Stellar [2].

A estrutura de banco de dados utilizada pelo Bitcoin, nomeada como Blockchain, se caracteriza por ser uma rede descentralizada e distribuída, além da utilização de criptografia nos seus protocolos de operação [1]. A popularidade do Bitcoin ajudou a alavancar novos projetos com diferentes aplicações inspiradas na Blockchain.

Contratos inteligentes foram descritos inicialmente por Nick Szabo em 1996, como um conjunto de protocolos de transações que representam termos reais de negócios, derivando princípios jurídicos e teoria econômica [4]. Com o lançamento da rede Ethereum em 2014, um projeto com inspirações no Bitcoin [5], os contratos digitais passam a ter uma infraestrutura facilitadora para sua aplicação. A rede Ethereum trouxe uma implementação de Blockchain com novas funcionalidades, de tal modo a permitir que contratos digitais possam ser especificados por qualquer pessoa, e lançados na rede [5].

Assim a ideia original de Nick Szabo [4], passa a usufruir de todos os protocolos de segurança, criptografia, rastreabilidade, e descentralização da Blockchain implementada pela rede Ethereum [5],
permitindo a criação de novas aplicações distribuídas baseadas em contratos inteligentes, que permitem digitalizar processos [6].

Para um usuário interagir com um contrato digital ele precisa se conectar a um nó de blockchain, para tal existem diferentes alternativas. Todas as alternativas de conexão, possuem a necessidade de um cliente que conhece os métodos do contrato, para realizar as devidas interações no contrato [7].

Sendo assim, o usuário tem a opção de utilizar um cliente genérico e definir de forma própria os métodos neste, restringindo o uso do contrato a usuários com familiaridade com tais ferramentas, que apesar de proverem abstração de alto-nível nas operações, são pouco amigáveis para usuários sem conhecimento prévio acerca de aplicações descentralizadas.

Nesse contexto, este trabalho de conclusão de curso, implementa uma interface web de um contrato digital, que permite um usuário interagir e realizar operações de forma amigável com o contrato, sem a necessidade de conhecimento prévio acerca de detalhes de implementação, variáveis de ambiente, e conceitos de operações descentralizadas.

A interface oferece um visual semelhante a aplicações tradicionais bancárias, com o intuito de inserir o usuário em um contexto conhecido. Sendo assim, o trabalho busca atenuar a distância entre usuários e aplicações descentralizadas, que já existe por diversos outros fatores como segurança, transparência e escalabilidade [10].

Atualmente no mercado, existem algumas soluções de interface web para interação com contratos inteligentes como o Metamask [9], Coinbase Wallet [17] e a Trust Wallet [18]. No entanto, o contrato em questão no qual a aplicação deste trabalho provém

interface, detém funcionalidades particulares ligadas à descentralização bancária.

Trata-se de um contrato de transações validáveis por terceiros, onde um contratante submete a transação, apontando o beneficiário e um terceiro, o validador. Este validador tem o papel de reconhecer o cumprimento de certos pré-requisitos do mundo real antes de uma data limite, tais como exemplo, a prestação de um serviço. Isto permite conceder a autoridade de validador para qualquer indivíduo, desde que haja confiança das outras duas partes envolvidas: pagador e o beneficiário, sendo este o prestador do serviço, descentralizando a função de intermediário confiável e virtualizando a validação de eventos e fatos do mundo real.

Pelo fato do contrato tratar de um fluxo financeiro com vários papéis e fluxos diferentes, as soluções existentes no mercado, não conseguem atender de forma ideal, a demanda por uma interface que simplifique a complexidade dos casos de uso. A solução aqui proposta, diminui o foco em detalhes técnicos e de implementação, e mira na experiência de uso do usuário.

Este trabalho foi desenvolvido no contexto da tese de doutorado do aluno Tiago Clementino, em complemento ao o TCC do aluno Marcos Barros, que desenvolveu a estrutura do contrato digital onchain (interna à blockchain), e permite a realização de pagamentos com a presença intermediária de validadores entre compradores e vendedores [8].

Assim, a aplicação interage com o referido contrato, permitindo que todos os papéis do contrato exerçam suas atribuições dentro dos pagamentos, sejam de compradores, vendedores ou validadores.

2. SOLUÇÃO

A interface desenvolvida, é uma aplicação web voltada para usuários que desejam realizar operações de interação com o contrato inteligente de 3 pontas, implementado no trabalho de conclusão de curso [8]. O foco do trabalho é possibilitar que usuários comuns, que não possuam conhecimento prévio acerca de conceitos de blockchain e operações com contratos inteligentes, consigam consumir e transacionar no contrato de modo análogo a operações realizadas em aplicações financeiras tradicionais na web, nas quais já estão habituados, sem a necessidade do conhecimento de variáveis de ambiente como o endereço do contrato.

2.1 Descrição da Aplicação

O presente trabalho, propõe uma solução facilitadora para interação entre usuários e um contrato digital na blockchain, de modo que a realização de operações neste contrato seja tão simples quanto operações cotidianas realizadas em aplicativos bancários ou redes sociais.

Sendo assim, a aplicação consiste em um sistema web acessível por qualquer navegador moderno, que se conecta à blockchain por intermédio de um cliente genérico, o Metamask [9], instalado via extensão no navegador, e interage com os métodos do contrato, provendo ao usuário uma interface gráfica, apresentada na Figura 1.

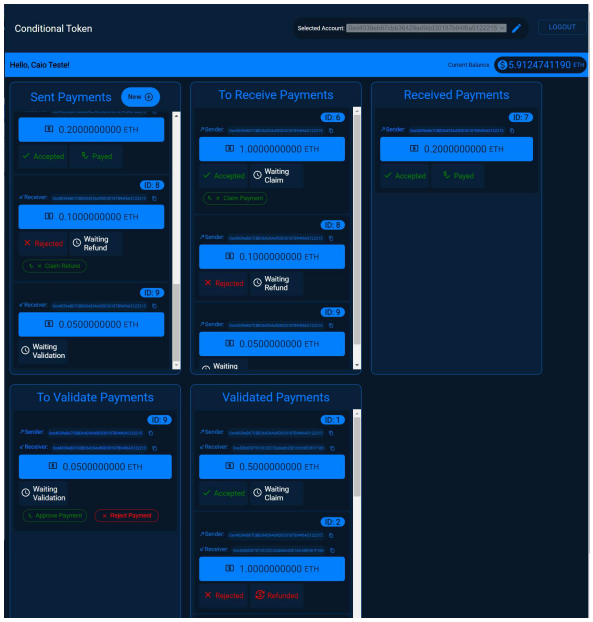

Figura 1. Visão geral da aplicação

O Metamask [9], é uma carteira de moedas digitais, que permite interação com contratos digitais na Blockchain, através de uma interface web. Para interagir com um contrato através do Metamask, o usuário precisa conhecer a assinatura das funções implementadas no contrato, e inseri-las na interface. Por não conhecer as funções dos contratos, o Metamask é uma interface de acesso genérico.

Além da interface web, o Metamask [9] oferece uma API de conexão à Blockchain, que pode ser consumida por aplicações de terceiros. Sendo assim, a interface implementada neste trabalho utiliza a API oferecida pelo Metamask, como meio de acesso à Blockchain, conforme a Figura 2 .

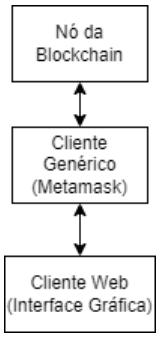

Figura 2. Diagrama de comunicação com a Blockchain

2.2 O problema da necessidade do conhecimento prévio do contrato

A necessidade de um cliente que se conecte a um nó da blockchain para transacionar em um contrato digital, e que tal cliente conheça os métodos deste contrato, acaba tornando a interação entre usuário e contrato não-trivial, assim limitando o acesso ao contrato para usuários com conhecimento prévio sobre as tecnologias relacionadas, ferramentas e afins.

Para realizar as operações no contrato, o cliente deve indicar explicitamente o método da operação que está realizando, além do endereço do contrato. Sendo assim, para que o usuário interaja com o contrato, em uma interface genérica (que não conhece os métodos e endereço do contrato), é necessário inserir tais informações de forma manual.

Na solução implementada neste trabalho, o usuário passa a utilizar um cliente genérico, o Metamask [9], como meio de acesso a um nó da blockchain, contudo sem a necessidade de conhecer e inserir de forma manual os métodos e endereço do contrato, que são conhecidos pela interface web. Dessa forma, o meio de acesso deixa de ser genérico e passa a ser personalizado para o contrato, logo o usuário não precisa saber detalhes de implementação para utilizar o contrato.

A aplicação dispõe de um servidor, banco de dados e API próprios, implementados sob demanda para permitir armazenar e retornar os métodos e endereço do contrato, além dos dados que o controle de sessão dos usuários.

A interação com a Blockchain se dá utilizando os métodos e endereços, obtidos através da API própria da aplicação, tendo o cliente genérico Metamask [9] como API de acesso ao contrato inteligente .

2.3 Funcionalidades

Conforme descrito na introdução, a aplicação suporta 3 papéis de usuários diferentes, sendo que um mesmo usuário pode exercer múltiplos papéis, assim os papéis suportados são:

- Contratante: responsável por criar uma transação de pagamento, onde a sua execução está condicionada a algum requisito ou evento;
- Validador: responsável por indicar se o evento ou condição determinada para execução de um pagamento ocorreu, é indicado pelo contratante na criação do pagamento que, idealmente, dispõe da confiança do contratante e do beneficiário;
- Beneficiário: é quem recebe o pagamento, após o validador indicar que a transação de pagamento foi aprovada, ele pode solicitar o recebimento do pagamento.

Com base em tais papéis, a aplicação dispõe das seguintes funcionalidades: autenticação, seleção de conta, saldo, realização de pagamento, validação de pagamento, reivindicação de pagamento, e reivindicação de reembolso.

2.3.1 Autenticação

Apesar das operações na blockchain possuírem um nível de autenticação independente, realizado via conta no cliente genérico Metamask [9], a aplicação possui autenticação para controle de sessão do usuário, guardar predefinições e registros dele.

Sendo assim há uma tela de Cadastro (Figura 3) e uma tela de Login (Figura 4). Na tela de cadastro o usuário cria suas credenciais de acesso, introduzindo nome, email e senha. Já na tela de login o usuário inicia a sessão, com as credenciais previamente cadastradas.

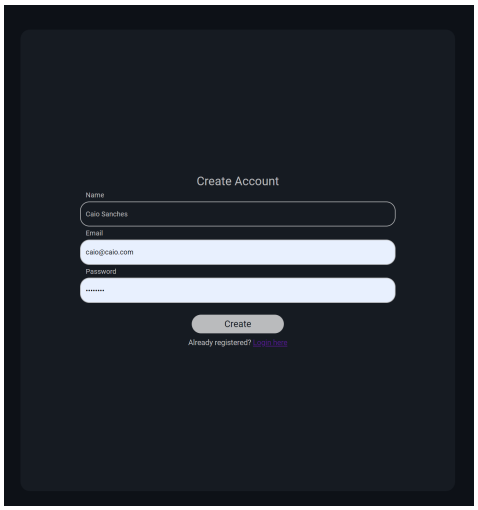

Figura 3. Tela de cadastro de usuário

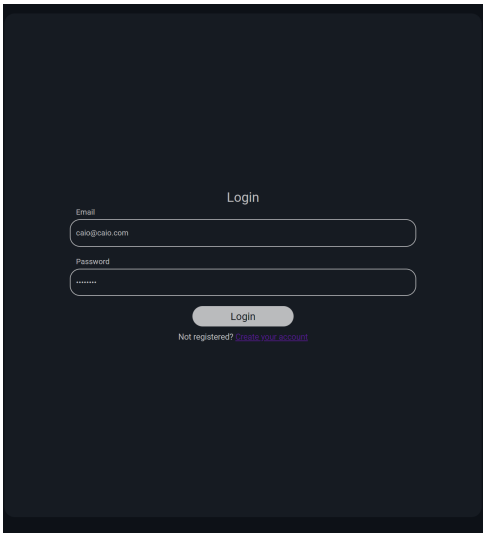

Figura 4. Tela de login de usuário

Para utilizar a aplicação, o usuário precisa ter uma sessão ativa, sendo assim, se ele tentar acessar qualquer URL da aplicação sem sessão, será direcionado para a tela de login.

2.3.2 Seleção de Conta

Permite selecionar uma conta entre as disponíveis na carteira do Metamask [9] (Figura 5). A seleção da conta é salva no banco de dados próprio da aplicação, logo se o usuário logar na aplicação em qualquer dispositivo que tenha uma conta disponível e pré-definida como selecionada anteriormente, terá a predefinição de conta carregada na aplicação.

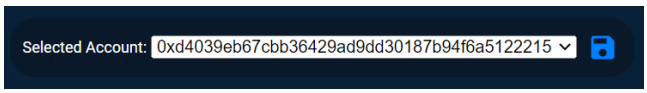

Figura 5. Seletor de conta

2.3.3 *Saldo*

Exibe o saldo atual disponível (Figura 6) na conta selecionada do Metamask [9].

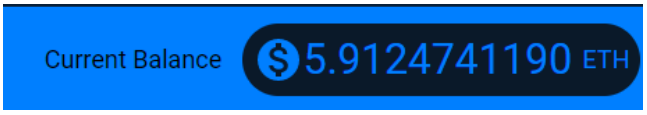

Figura 6. Card com saldo atual da conta do usuário.

2.3.4 *Realizar Pagamento*

Realizada pelo contratante, permite criar uma transação de pagamento, onde o usuário indica o valor a ser pago ao beneficiário, e uma taxa a ser paga a quem validar a transação, a soma do valor do pagamento e da taxa deve ser menor ou igual ao saldo da conta do usuário, além disso, na criação da transação há a indicação dos validadores desta, através do endereço das suas contas, conforme a Figura 7.

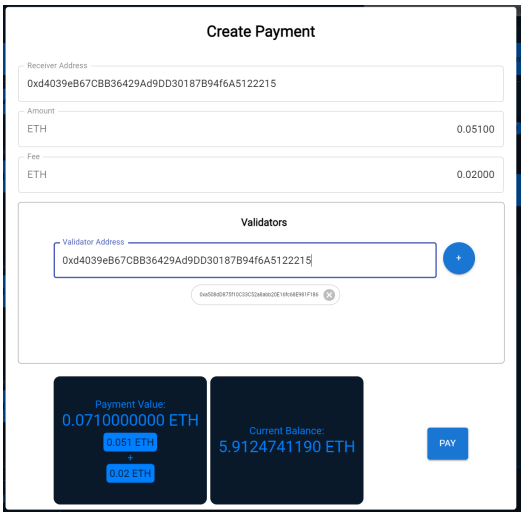

Figura 7. Tela de criação de pagamento.

O Modal de criação de pagamento, é exibido após o usuário clicar no botão "New" da coluna "Sent Payments".

É possível adicionar mais de um validador na transação, contudo, um único realiza a validação, a decisão de qual dos validadores incluídos será responsável pela validação, é feita pelo critério de corrida, ou seja, valida a transação quem primeiro disparar a ação de aceite ou rejeição. Após a primeira validação, o pagamento perde o status de validável, logo os demais validadores não conseguem realizar nenhuma ação.

Ao menos um validador deve ser incluído em um pagamento. Ao criar um pagamento, uma nova transação será adicionada à coluna de pagamentos enviados "Send Payments" do contratante, e a coluna "To Validate Payments" dos validadores.

Após a validação de um usuário, a transação tem seu status alterado, e libera a reivindicação de pagamento do beneficiário caso a operação tenha sido aprovada, ou o reembolso pelo contratante caso a operação tenha sido rejeitada.

2.3.5 *Validar Pagamento*

Realizada pelos validadores, permite aprovar ou rejeitar pagamentos, que atribuíram o usuário como validador.

Os pagamentos com validações pendentes são exibidos na coluna "To Validate Payments", caso o usuário tenha sido atribuído como validador, e possuem botões de ação para aprovar ou rejeitar o pagamento.

Ao clicar em algum dos botões, a validação é enviada e o pagamento passa a ser listado na coluna de "Validated Payments" como na Figura 8. Caso o pagamento tenha sido aprovado, o beneficiário passa a ter um botão de "Claim Payment" na coluna de "To Receive Payments", que possibilita que ele reivindique o pagamento. Caso o pagamento tenha sido rejeitado, o contratante passa ter um botão de "Claim Refund" na coluna de "Sent Payments", que possibilita que ele reivindique o reembolso.

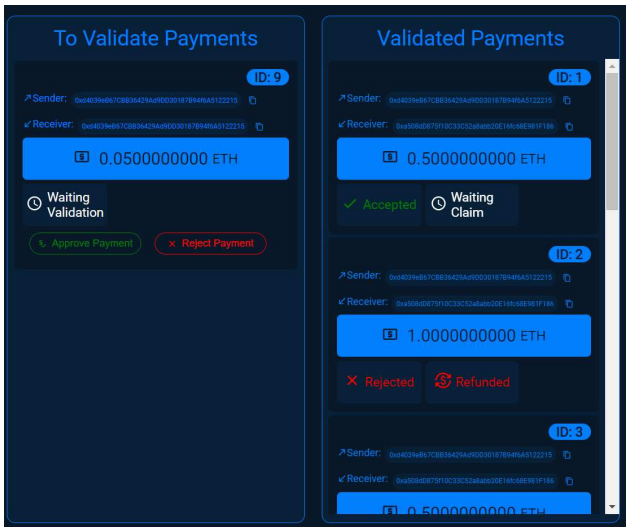

Figura 8. Listagem de pagamentos disponíveis para validação e validados.

2.3.6 *Reivindicar Pagamento*

Realizada pelo beneficiário do pagamento, permite que o usuário indicado para receber um pagamento, reivindique o recebimento de um pagamento que foi validado e aprovado.

Os pagamentos a serem recebidos por um usuário ficam listados no card de "To Receive Payments" (Figura 9), caso um pagamento esteja com o status de validado e aprovado, um indicador de "Accepted" é exibido, junto a um botão de "Claim Payment", ao clicar no botão, o usuário reivindica o pagamento, ao fim da operação o saldo vem para a carteira deste, e o pagamento vai para a coluna de "Received Payments".

Enquanto um pagamento não teve o pagamento reivindicado, ele mostra um indicador de "Waiting Claim".

Adicionalmente são listados os pagamentos que foram endereçados ao usuário, contudo não foram aprovados na validação. Para estes um indicador de "Reject" é exibido, e o usuário não pode realizar nenhuma ação nele.

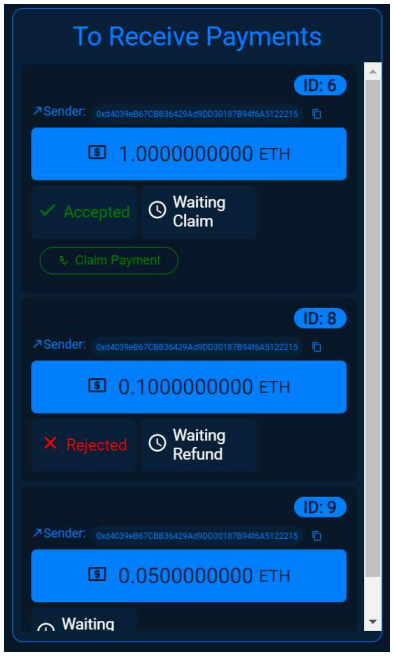

Figura 9. Listagem de pagamentos a receber

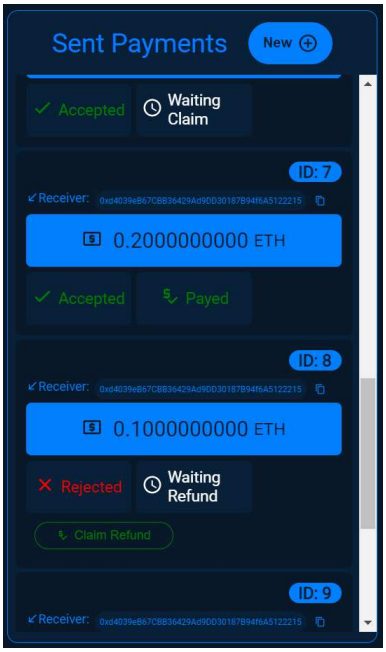

Figura 10. Listagem de pagamentos enviados, com botão para reembolso

2.3.7 Reivindicar Reembolso

Realizada pelo contratante, permite que um usuário reivindique o reembolso de um pagamento feito por ele e rejeitado pelo validador.

Os pagamentos listados na coluna de "Sent Payments" (Figura 10) que tenham o status de validado e rejeitado, exibem um indicador de "Rejected", junto a um botão de "Refund", ao clicar no botão, o usuário solicita o reembolso do pagamento.

Enquanto o usuário não reivindica o reembolso, um indicador de "Waiting Refund" é exibido.

Ao fim da operação de reembolso, o valor do contrato volta para a conta do contratante, e o pagamento passa a exibir o indicador de "Payed".

3. ARQUITETURA E IMPLEMENTAÇÃO

- A aplicação possui um cliente web que o usuário final consome, alimentado por dois serviços:
- Servidor próprio, responsável pelo controle de usuários e suas predefinições, junto a um banco de dados não-relacional, responsável pela persistência das informações;
- Cliente Metamask, instalado via extensão no navegador do usuário, responsável por prover a comunicação com o contrato inteligente na blockchain, injetando uma api Web3 na aplicação que possibilita a realização das transações.

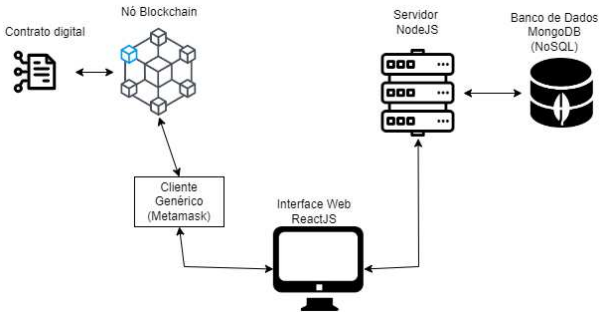

Figura 10. Diagrama com arquitetura da aplicação

No servidor, a arquitetura utilizada foi do tipo MVC (Model View Controller), com intuito de separar as responsabilidades e regras de negócio da modelagem das entidades e seus atributos, além de simplificar o desenvolvimento, dado o escopo restrito do backend na aplicação.

- Sendo assim o sistema foi organizado da seguinte

forma:

- Uma camada de View nomeada como router que expõe as funções dos controladores à rotas da api pública;
- Uma camada de controladores, que realiza as manipulações de regras de negócios, se comunicando com a camada de model.
- Uma camada de model, que realiza a interface com o banco de dados para leitura e escrita.

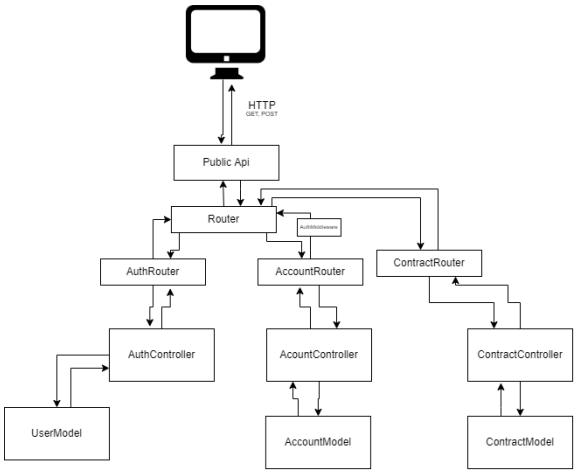

Figura 11. Diagrama com arquitetura do servidor

No banco de dados, são persistidos os dados relacionados ao usuário e suas predefinições, sendo assim existem duas entidades guardadas em coleções:

- Usuário: entidade com dados de usuário, contendo Nome, email e senha, sendo estes do tipo string, e sabendo que a senha salva é um hash do tipo Bcrypt, é utilizada para o controle de sessão do usuário, sendo criada no fluxo de cadastro, e consultada no fluxo de login;
- Account: entidade utilizada para guardar a associação entre usuário e id de conta, sabendo que o id de conta é utilizado como identificador do usuário nas transações com o contrato digital, e que o usuário pode dispor de mais de uma conta na sua carteira do Metamask [9], a persistência da seleção de conta é importante no cenário onde o usuário possui diversas contas e utiliza uma para interagir com o contrato digital, os campos da entidade são id de usuário do tipo ObjectId, e id da conta do tipo string, e há uma constraint de unicidade por id de usuário.

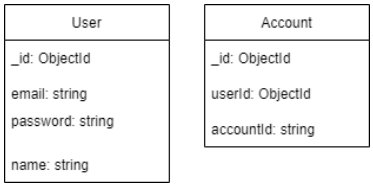

Figura 12. Diagrama com modelo de dados

No frontend, foi utilizada uma arquitetura baseada em componentes, dada o grande reuso de código, proporcionado pela biblioteca utilizada no desenvolvimento, além disso o gerenciamento de estados se deu através do uso da ContextApi do React, que permite criar estado compartilhado entre componentes filhos através de um provider, sem a necessidade do uso de bibliotecas que introduzem complexidade na aplicação, como Mobex e Redux.

Sendo assim, foram utilizados dois providers, para prover contexto do estado de autenticação do app, e do estado de interação com o metamask.

Logo a camada de componentes, que é acessada pelo Router, passou a ser envolvida pelo contexto do metamask, do contrato e o de autenticação, permitindo que os componentes filhos, consigam utilizar funções e acessar o estado compartilhado através do uso de HOCs (Componentes de alta ordem) personalizados, que provém api para acessar os estados e funções compartilhados.

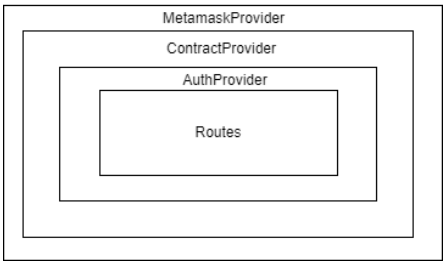

Figura 13. Diagrama com hierarquia de contextos

Para a comunicação com o contrato, foi utilizada a biblioteca ethers.js [11], que utiliza a instância Web3 injetada pelo metamask, e provê uma API para se comunicar com métodos do contrato inteligente.

Sobre os métodos do contrato, a assinatura destes é recebida via API do backend da aplicação em um arquivo JSON, utilizada posteriormente para criar uma instância da ContractApi oferecida pela biblioteca escolhida.

3.1.1 Tecnologias utilizadas

- Metamask: é uma carteira de criptomoedas largamente utilizada na blockchain da rede Ethereum para uso em aplicações e de forma independente, compatível com diversos navegadores modernos e dispositivos móveis, que funciona também como um cliente da blockchain na rede escolhida, e injeta no dom de aplicações web uma instância do tipo Web3, que permite a interação com tokens e contratos digitais na blockchain;
- ReactJS: utilizada no cliente web, ReactJS é uma popular biblioteca de JavaScript, amplamente utilizada na construção de interfaces de usuário, escolhida para o projeto devido a sua ampla comunidade de desenvolvimento, facilidade em escrever componentes reusáveis, e possibilidade de utilizar código JavaScript puro no desenvolvimento das funções;
- NodeJS: é um ambiente de execução que permite utilizar JavaScript no servidor, utilizando o mecanismo V8 do Google Chrome, aliado ao framework Express

permite criar uma aplicação escalável com poucas linhas de código, aliado ao fator da sintaxe ser bastante semelhante a utilizada no frontend, e por tal foi escolhido para este projeto;

- MongoDB: popular banco de dados não relacional, foi escolhido pela sua versatilidade ao dispor dos benefícios do NoSQL como a facilidade de escalar horizontalmente, ausência da necessidade de migrações, com a possibilidade de persistir estrutura de dados flexíveis, propiciando um desenvolvimento mais ágil e sob-demanda para as necessidades do projeto.
- Ether.js: biblioteca JavaScript, que através de uma instância Web3, permite se comunicar com a API de um contrato inteligente.

3.1.2 Sistema em uso

- Para uso do sistema em produção, foi utilizado um esquema de instâncias distribuídas independentes, com intuito de isolar os nós, evitando que a falha de um nó afete outro, assim a disposição da infraestrutura foi a seguinte:
- Instância do Heroku do tipo dyno Hobby [12], para servir o cliente da aplicação [13].
- Instância do Heroku do tipo dyno Hobby [12], para servir a api da aplicação [14].
- Cluster M0 do MongoDB Cloud [15], servindo o banco de dados da aplicação.
- Contrato inteligente implantado na rede de testes Ropsten [16]

4. METODOLOGIA

A metodologia ágil escolhida para o desenvolvimento foi o SCRUM modificado, devido a sua flexibilidade e possibilidade de realizar iterações acerca do planejamento inicial durante a execução.

Devido a execução ser realizada por um único contribuidor, o SCRUM foi adaptado para que o backlog se tornasse mais flexível. As validações idealmente realizadas a cada entrega, foram realizadas ao fim de cada sprint, além disso não foram realizados rituais do tipo retrospectiva.

Foram definidas sprints de 15 dias, com a realização de reuniões semanais, junto aos integrantes do projeto desenvolvido em conjunto a este, do Contrato Inteligente de três pontas [8].

4.1 Etapas de Desenvolvimento

O desenvolvimento do trabalho foi dividido entre as seguintes etapas (Figura 14):

Figura 14. Etapas de desenvolvimento

4.1.1 Elicitação de Requisitos

Na etapa de elicitação de requisitos, foi utilizada a técnica de definir casos de uso. Tendo em vista que este Trabalho foi desenvolvido no contexto da tese de doutorado do pesquisador Tiago Clementino, e em complemento ao TCC do aluno Marcos Barros, a construção desses casos (Figura 15) se deu em conjunto durante uma reunião virtual.

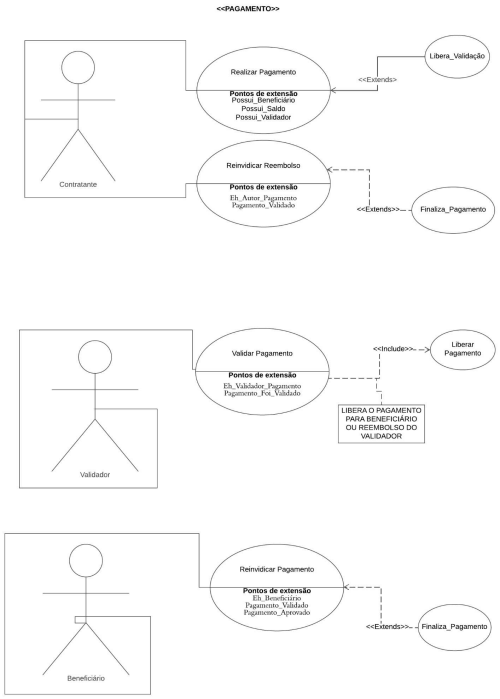

Figura 15. Casos de Uso

4.1.2 Modelagem de entidades e desenho da Arquitetura

Com os requisitos definidos, a etapa de modelagem levou em consideração a estrutura do contrato inteligente [8], para determinar as necessidades arquiteturais da aplicação.

Nesta etapa foram levantadas as necessidades dos seguintes items:

- Cliente Web (Frontend)
- Servidor próprio, para servir detalhes parametrizáveis acerca do contrato, e permitir o controle de sessão de usuários.
- Banco de dados próprio, responsável por guardar dados dos usuários.
- Cliente genérico, que permitisse acessar um nó da blockchain, e oferecesse uma instância Web3 que o frontend pudesse consumir.

4.1.3 Implementação

A etapa de implementação foi realizada durante 4 sprints (60 dias), seguindo a seguinte sequência:

1. Implementação da API e Banco de Dados
2. Implementação do Frontend e integração com a API implementada anteriormente.
3. Integração com o Metamask [9]
4. Setup da infraestrutura e deploy do Cliente e Servidor.
5. Integração com o Contrato Digital [8]
6. Documentação

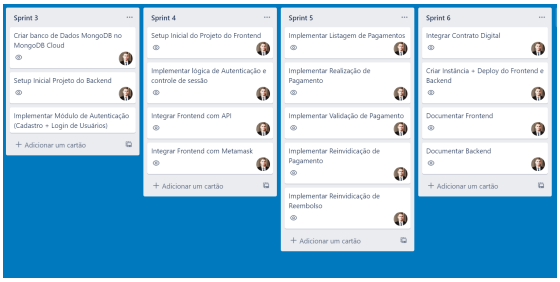

Figura 16. Board com tarefas de implementação realizadas durante sprints.

Para ajudar próximos contribuidores do projeto, foram realizadas as seguintes ações de documentação de código:

- Incluir comentário nas funções do Frontend e Backend
- Documentação dos endpoints da API utilizando a ferramenta SWAGGER [24]
- Inclusão de Readmes nos repositórios, com passos para executar e realizar deploy das aplicações.

Para o fluxo de deploys, foi realizada a integração contínua entre os repositórios do cliente e servidor, e as devidas instâncias e pipeline no Heroku [12].

4.1.4 Validação

Para validação da aplicação foi realizada uma pesquisa de usabilidade, através de um formulário, com as seguintes perguntas:

1. Você já tinha interagido com algum Contrato Inteligente? (Sim ou Não)
2. O Sistema possui fácil usabilidade? (1 a 5)
3. Você conseguiu realizar um pagamento sem dificuldades? (Sim, Não, Não consegui responder)
4. As funcionalidades disponíveis ficaram claras para você? (Sim ou Não)

O Formulário foi enviado para diferentes perfis de usuários, sendo que 11 respostas foram recebidas, e os resultados obtidos foram os seguintes (Figura 17), (Figura 18), (Figura 19) e (Figura 20):

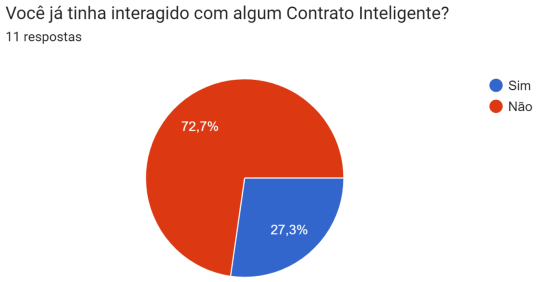

Figura 17. Gráfico com distribuição de respostas da pergunta 1.

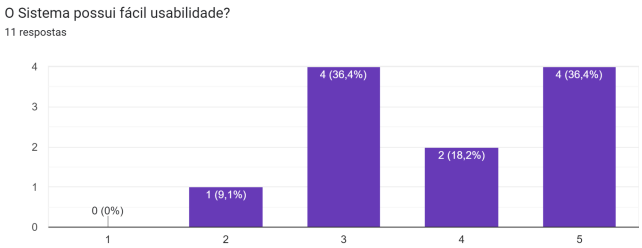

Figura 18. Gráfico com distribuição de respostas da pergunta 2.

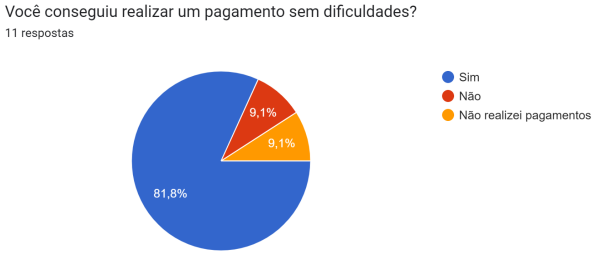

Figura 19. Gráfico com distribuição de respostas da pergunta 3.

Figura 20. Gráfico com distribuição de respostas da pergunta 4.

Com base nas respostas, é possível conjecturar que a maioria dos usuários respondentes não havia interagido com Contratos Inteligentes anteriormente, e que estes conseguiram realizar pagamentos na interface sem dificuldades.

Contudo apesar da grande maioria ter achado o sistema de fácil usabilidade, uma parcela expressiva indicou que as funcionalidades não ficaram claras.

Tal resultado indica que apesar da interface simplificar a interação entre usuário e contrato inteligente, ainda há espaço de melhoria na usabilidade da aplicação.

Remover os passos intermediários para uso da interface, aliado a melhorias de interface que deixem claro todas as funcionalidades, deve melhorar este cenário.

5. CONCLUSÕES E TRABALHOS FUTUROS

5.1 Conclusões

O presente trabalho descreve o relato de desenvolvimento de uma interface web que se conecta à Blockchain para interagir com um contrato inteligente. A solução proposta e entregue, alcançou o objetivo de viabilizar o uso de um contrato inteligente de três-pontas (contratante, validador e beneficiário), por usuários comuns.

Além disso, dado o contexto de aplicação descentralizada com diferentes stakeholders, e sistemas distribuídos, o leitor tem a oportunidade de dimensionar, e entender possíveis desafios e oportunidades no desenvolvimento de uma aplicação descentralizada que tenha características semelhantes.

5.2 Contribuições

O público-alvo deste trabalho é desenvolvedores de software, que busquem referências acerca de aplicações descentralizadas.

A referência das tecnologias utilizadas, junto ao esquema de infraestrutura, permite ao leitor prever possíveis requisitos no desenvolvimento de alguma aplicação com características semelhantes.

Para usuários, a contribuição do trabalho está relacionada à acessibilidade de uma aplicação descentralizada. Possibilitando a interação com um contrato inteligente de forma amigável, sem a necessidade de conhecimento prévio, acerca de detalhes de implementação do contrato [8], assim ampliando o potencial público que pode utilizar tal contrato.

5.3 Experiência

A experiência de desenvolver uma aplicação com camadas descentralizadas, foi desafiadora e enriquecedora.

Trabalhar com temas recentes na comunidade como Web3, requer um maior nível de pesquisa e esforço na tomada de decisões e escolha de ferramentas.

A falta de consenso sobre parte dos temas pelo pouco tempo de existência e tamanho potencial a ser explorado, permite a experimentação e novas descobertas.

Aliado a isto, a possibilidade de trabalhar em equipe, em um projeto de pesquisa que faz parte de uma tese de doutorado, e possibilitou o desenvolvimento em paralelo de dois trabalhos de conclusão de curso, foi uma oportunidade motivadora e gratificante.

O conhecimento adquirido ao longo da graduação de Ciência da computação, foi de suma importância na execução deste trabalho.

Além de habilidades técnicas de desenvolvimento e planejamento, obtidas em disciplinas como Projeto de Software e Engenharia de Software, habilidades obtidas em disciplinas como Economia de Tecnologia da Informação e Análise de Sistemas, permitiram aproximar o trabalho desenvolvido de um produto real e viável, com a mitigação de riscos e empecilhos.

Sobre o processo de desenvolvimento, o uso da metodologia ágil Scrum ajudou no planejamento e execução do projeto. A definição prévia dos épicos, com data limite de entrega, e histórias contempladas nas sprints, permitiu uma visão de longo prazo. Fundamental na tomada de decisões relacionadas à delimitação de escopo, com intuito de cumprir o planejamento estabelecido.

Como o projeto inicial previa a integração com um gateway de pagamentos, para possibilitar que o saldo na conta do usuário fosse inserido pela interface, haviam complexidades burocráticas que não foram previstas inicialmente. Uma análise de risco teria mitigado tais complexidades, e possibilitado que o planejamento inicial fosse cumprido de forma integral.

Acerca das tecnologias utilizadas, a escolha do ReactJS [19], se mostrou uma decisão acertada, a possibilidade de utilizar bibliotecas nativas de JavaScript, facilitou a implementação, haja vista a possível limitação de opções se a tecnologia utilizada fosse do tipo Framework, e demandar bibliotecas específicas para ela.

5.4 Limitações

Por se tratar de uma aplicação de cunho financeiro, sujeita a regulamentação de órgãos públicos como o Banco Central e o Conselho Monetário Nacional, houveram algumas limitações no escopo deste trabalho.

Inicialmente, foram previstas as funcionalidades de depósito e saque através da aplicação, haja vista a necessidade de criar uma empresa financeira para obter CNPJ, e posteriormente realizar os devidos registros previstos na normativa do Banco Central do Brasil [20]. Sendo assim, tais funcionalidades foram retiradas do escopo deste trabalho, de modo a restringir a aplicação à uma interface para realização de operações no contrato digital.

Portanto, todo o contexto de transações financeiras e transferência de valores monetários, ocorre no Metamask, que funciona como cliente genérico de acesso a um nó da blockchain, e carteira de ativos descentralizados.

5.5 Resultados

O trabalho final está disponível em uma URL pública [13], e pode ser acessado por qualquer navegador moderno que suporte HTML 5 e tenha a extensão Metamask instalada [23].

5.6 Trabalhos Futuros

A aplicação desenvolvida neste trabalho, faz parte de um projeto que tem o intuito de possibilitar a virtualização de eventos do mundo físico com origem na tese de doutorado do pesquisador Tiago Clementino.

Sendo assim, as seguintes etapas posteriores são esperadas como trabalhos futuros:

- Evoluir a infraestrutura da aplicação, a partir de melhorias pontuais como o uso de instâncias com auto-scaling, e escala horizontal, e de um cluster dedicado para servir o banco de dados;
- Instrumentar a aplicação com ferramentas de monitoramento de erros e desempenho como New Relic [21] e Sentry [22], que possibilitam identificar possíveis bugs, além de gargalos na infraestrutura;
- Aprimorar a interface web para simplificar o uso por usuários comuns, inserindo a funcionalidade de agenda de contatos, com os endereços de contas de outros usuários conhecidos, mapeados e nomeados, permitindo assim que o usuário não tenha que lidar com endereços hash da conta de terceiros durante operações cotidianas;
- Permitir a inserção de saldo na conta do usuário, através de um gateway de pagamento na aplicação;
- Aplicar a solução em um produto financeiro real, como uma seguradora descentralizada, ou um crowdfunding descentralizado.

6. AGRADECIMENTOS

Agradeço inicialmente a todos que compõem o curso de Ciência da Computação na UFCG, e se esforçam para que este seja um dos melhores cursos do país, não há dúvidas que o ponto mais importante deste curso são as pessoas.

Agradeço também aos meus pais Perla Batista, e Vagner Lira, por me concederem todo apoio necessário para que este momento se tornasse realidade, a minha avó Diana Maria (in memorian), a meus irmãos Cauã Sanches e Paloma Sanches, e a minha namorada Sabrina Morais que tive o prazer de conhecer na Universidade e foi um grande ponto de apoio por todo este tempo.

Ao professor Antão e ao pesquisador Tiago Clementino, por aceitarem me orientar e participar de um trabalho enriquecedor e desafiador, e ao amigo e colega de curso Marcos Barros, que trabalhou em cooperação comigo neste trabalho.

Por fim, agradeço a todos os amigos que fiz neste curso, que foram fundamentais para o sucesso da minha graduação, e me permitiram crescer como profissional, mas também como pessoa.

7. REFERÊNCIAS

[1] Nakamoto, S.: Bitcoin whitepaper. 2008.

Disponível em: https://bitcoin.org/bitcoin.pdf.

Acesso em 05/08/2022

[2] Dastgir S, Demir E, Downing G et al. 2019

Disponível em:
https://link.springer.com/article/10.1007/s43546-022-00206-5

Acesso em 05/08/2022

[3] A Survey of Blockchain From the Perspectives of Applications, Challenges, and Opportunities. 2019

Disponível em: https://ieeexplore.ieee.org/document/8805074

Acesso em 07/08/2022

[4] Szabo, N. (1997). Formalizing and Securing Relationships on Public Networks. First Monday, 2(9). Disponível em: https://doi.org/10.5210/fm.v2i9.548

Acesso em 07/08/2022

[5] BUTTERIN, Vitalik. Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform. 2014.

Disponível em:
https://ethereum.org/669c9e2e2027310b6b3cdce6e1c52962/Ethereum_Whitepaper_-_Buterin_2014.pdf.

Acesso em 07/08/2022

[6] Bocek, T., & Stiller, B. (2017). Smart Contracts – Blockchains in the Wings.

Disponível em:
https://link.springer.com/chapter/10.1007/978-3-662-49275-8_19

Acesso em 07/08/2022

[7] Matevž Pustišek, Andrej Kos,

Approaches to Front-End IoT Application Development for the Ethereum Blockchain 2018

Disponível em:

https://www.sciencedirect.com/science/article/pii/S1877050918302308

Acesso em 10/08/2022

[8] FILHO, Marcos Barros Medeiros (2022),

Contrato inteligente de três pontas:

Pagamentos Seguros

https://github.com/conditional-token/conditional-token

Acesso em 01/08/2022

[9] Why Metamask

Disponível em: https://docs.metamask.io/guide/#why-metamask

Acesso em 13/08/2022

[10] Tiago L.P. C, Joaquim J.C.M , J. Antão B.M et al (2022) Contratos Inteligentes e o Usuário Comum: Revisão Sistemática e Agenda de Pesquisa

Disponível em: https://drive.google.com/file/d/1fkRSEGIRiyW1LUr7LHceYlLJ8Qav5myK/view?usp=sharing

Acesso em 17/08/2022

[11] Ethers.js

Disponível em:

https://docs.ethers.io/

Acesso em 15/08/2022

[12] Dynos: the heart of the Heroku platform

Disponível em:

https://www.heroku.com/dynos

Acesso em 17/08/2022

[13] Interface Web Pagamento Seguro.

Disponível em: https://conditional-token-frontend.herokuapp.com/ Acesso em: 16 ago. 2022.

[14] API Web Pagamento Seguro.

Disponível em: https://conditional-token-backend.herokuapp.com/ Acesso em: 16 ago. 2022.

[15] Atlas M0 (Free Cluster), M2, and M5 Limitations

Disponível em:

https://www.mongodb.com/docs/atlas/reference/free-shared-limitations/

Acesso em: 16 ago. 2022.

[16] Endereço do pagamento seguro.

Disponível em:

https://ropsten.etherscan.io/address/0x6e2a4bda853cba7a7198a63b2be9a9ab654b503d .

Acesso em: 16 ago. 2022.

[17] Coinbase Wallet

Disponível em:

https://www.coinbase.com/pt/wallet

Acesso em 18/08/2022

[18] TrustWallet

Disponível em:

https://trustwallet.com/

Acesso em 18/08/2022

[19] ReactJS, Uma biblioteca JavaScript para criar interfaces de usuário. Disponível em: https://pt-br.reactjs.org/ Acesso em 18/08/2022

[20] RESOLUÇÃO Nº 4.656, Banco Central do Brasil

Disponível em:

https://www.bcb.gov.br/pre/normativos/busca/downloadNormativo.asp?arquivo=/Lists/Normativos/Attachments/50579/Res_4656_v1_O.pdf

Acesso em 18/08/2022

[21] Why New Relic?

Disponível em:

https://newrelic.com/

Acesso em 18/08/2022

[22] Sentry: Application Monitoring and Error Tracking Software

Disponível em:

https://sentry.io/

Acesso em 18/08/2022

[23] Metamask: Supported Browsers

Disponível em:

https://metamask.io/download/

Acesso em 18/08/2022

[24] SWAGGER: API Development for everyone

Disponível em: https://swagger.io/ Acesso em 18/08/2022