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

HENRIQUE CASTRO ARRIEL

ACADEMICBLOCK:
APLICAÇÃO DESCENTRALIZADA DE HISTÓRICOS ACADÊMICOS HOSPEDADOS EM UMA BLOCKCHAIN

CAMPINA GRANDE - PB

2022

HENRIQUE CASTRO ARRIEL

ACADEMICBLOCK: APLICAÇÃO DESCENTRALIZADA DE HISTÓRICOS ACADÊMICOS HOSPEDADOS EM UMA BLOCKCHAIN

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 : Kyller Costa Gorgônio

CAMPINA GRANDE - PB

2022

HENRIQUE CASTRO ARRIEL

ACADEMICBLOCK: APLICAÇÃO DESCENTRALIZADA DE HISTÓRICOS ACADÊMICOS HOSPEDADOS EM UMA BLOCKCHAIN

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:

Kyller Costa Gorgônio

Orientador – UASC/CEEI/UFCG

Rohit Gheyi

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

Este documento apresenta o relato de desenvolvimento de uma aplicação descentralizada (*dApp*) utilizando o conceito de *Web3*, que é a *Web* baseada na tecnologia *blockchain*, incorporando principalmente a descentralização. Com isso, é proposto o *AcademicBlock*, uma aplicação descentralizada com objetivo de permitir a visualização e criação de históricos acadêmicos dos alunos como NFTs (*non-fungible token*) hospedados na *blockchain*, introduzindo digitalmente a especificidade, individualidade e propriedade aos históricos dos alunos.

AcademicBlock: Aplicação descentralizada de históricos acadêmicos hospedados em uma blockchain

Henrique Castro Arriel
henrique.arriel@ccc.ufcg.edu.br
Universidade Federal de Campina Grande
Campina Grande, Paraíba

Kyller Costa Gorgônio
kyller@computacao.ufcg.edu.br
Universidade Federal de Campina Grande
Campina Grande, Paraíba

RESUMO

Este documento apresenta o relato de desenvolvimento de uma aplicação descentralizada (dApp) utilizando o conceito de Web3, que é a Web baseada na tecnologia blockchain, incorporando principalmente a descentralização. Com isso, é proposto o AcademicBlock, uma aplicação descentralizada com objetivo de permitir a visualização e criação de históricos acadêmicos dos alunos como NFTs (*non-fungible token*) hospedados na *blockchain*, introduzindo digitalmente a especificidade, individualidade e propriedade aos históricos dos alunos.

ACM Reference Format:
Henrique Castro Arriel and Kyller Costa Gorgônio. 2022. AcademicBlock: Aplicação descentralizada de históricos acadêmicos hospedados em uma blockchain. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

Palavras-chave

Blockchain, descentralização, dApp, aplicação, desenvolvimento, histórico.

Repositório

https://github.com/HenriqueCA/dapp-academic-record

1 INTRODUÇÃO

Recentemente, a tecnologia *blockchain* se popularizou por consequência da alta rentabilidade e esperança em torno das criptomoedas, moedas digitais que possuem a promessa de substituir o dinheiro físico da atualidade. Essa tecnologia é um tipo de base de dados distribuída que guarda permanentemente quaisquer transações que ocorreram na base, possuindo diversas características benéficas para seus usuários, como segurança através da descentralização, transparência e não violabilidade de seus dados.

Junto com a *blockchain*, a web também segue a tendência de descentralização através do novo conceito de *web3*. Os desenvolvedores dessa nova ideia utilizam sistemas descentralizados para prover suas aplicações, ganhando segurança e transparência. Tais aplicações, seguindo o protocolo *web3*, são chamadas de "aplicações descentralizadas", ou *dApps*. Um exemplo atualmente popular de uma aplicação descentralizada é o *OpenSea*[7], um site que permite a visualização, criação e vendas de NFTs (*non-fungible tokens*) como artes, músicas, vídeos e imagens.

Para auxiliar na implementação dessas aplicações, é feito o uso de contratos inteligentes (*smart contracts*), contratos digitais autoexecutáveis que não podem ser perdidos ou adulterados por estarem hospedados na própria blockchain. As aplicações se comunicam com esses contratos, realizando rotinas, permitindo validações e confiabilidade entre transações, realizando o acompanhamento das regras de transação sem a necessidade de um mediador, sem riscos de fraudes e alterações.

No contexto de NFTs, as *blockchains* não são ideais para armazenar dados muito grandes, como arquivos de imagens. Para isso, é comumente armazenado um *hash* de um localizador (*URI*) na *blockchain*. Esse *URI* aponta para o dado armazenado em algum banco de dados. Esse armazenamento é chamado de *off-chain*, pois está fora da rede da *blockchain*. Apesar disso, como a referência está armazenada na rede (*on-chain*), há todas as garantias providas pela tecnologia. Esse armazenamento de dados é geralmente feito com o protocolo de sistema de arquivos interplanetário (*IPFS*), o qual é um protocolo projetado para armazenamento associativo P2P endereçável ao conteúdo de armazenamento e compartilhamento num sistema de arquivos distribuído.

Além dessas tecnologias, há também o mecanismo utilizado para armazenar as transferências e *tokens* de um usuário, conhecido como uma carteira digital. Essas carteiras facilitam os processos de comunicação com as tecnologias descritas anteriormente. Para o protocolo *web3*, o software mais utilizado é o *Metamask*[4], uma extensão de navegador ou aplicativo mobile que se comunica com a *blockchain Ethereum*[1].

No contexto acadêmico, a introdução dessas tecnologias pode facilitar a identificação de fraudes de históricos, transferência entre instituições e armazenamento permanente e transparente de informações durante todo a carreira acadêmica de um indivíduo. Diversas instituições utilizam diferentes sistemas centralizados para armazenar e disponibilizar os históricos acadêmicos dos alunos, suscetíveis a falhas que podem afetar alunos e professores, além de dificultarem validações necessárias para qualquer objetivo que se faça necessidade do histórico.

No cenário atual, os trabalhos feitos utilizando tais tecnologias são voltadas para o processo interno da aplicação, e em grande maioria considerando apenas a finalização do currículo de um aluno (conclusão). Portanto, este presente trabalho pretende propor uma aplicação de alto nível para consulta e criação de históricos acadêmicos com o protocolo *web3*, acompanhando toda a trajetória acadêmica de um estudante, que pode ser utilizada como base para

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

criação de uma única aplicação descentralizada para diversas universidades, pois é feita em cima de uma *blockchain* que pode ser usada por todos, padronizando a forma em que informações sobre estudantes universitários são armazenados digitalmente e facilitando ações de transferência, seja entre diferentes cursos ou diferentes universidades. Esta solução também pode se estender para outras áreas, como, por exemplo, na saúde, permitindo o armazenamento de histórico médico de um paciente e facilitando as consultas e transferências de um paciente entre hospitais.

2 ARQUITETURA E PROJETO DA SOLUÇÃO

A arquitetura da aplicação descentralizada proposta pode ser dividida comumente nos dois processos: *frontend*, o qual é a interface da aplicação em que o usuário realiza ações através do seu dispositivo; e o *backend*, o processo interno da aplicação, contendo as lógicas de operação repassadas para o usuário através da interface.

2.1 *Frontend*

A interface do usuário foi implementada utilizando *Next.js*[6], um *framework* de código aberto para a biblioteca *React*[9], que usa a linguagem de programação *Javascript*. Esse *framework* facilita a produção de aplicações *web* baseada em componentes, adicionando funcionalidades como renderização estática e do lado do servidor. Para o *design* das páginas, foi utilizado o framework de *CSS Tailwind*[11], que permite o design através de classes diretamente no código *HTML*, agilizando o processo de desenvolvimento. Para facilitar a comunicação entre a interface e o *backend*, é utilizado o software de carteira *Metamask*[4] por uma extensão do navegador.

Para iniciar o desenvolvimento de uma interface, é comum a utilização de protótipos de design interface, ou *wireframes*. Nesse trabalho, foi utilizada uma ferramenta para edição gráfica de vetor, a aplicação web *Figma*[2], para produzir protótipos das telas finais da aplicação.

2.2 *Backend*

A lógica da aplicação é feita através das tecnologias de *blockchain* e *smart contract* mencionadas anteriormente.

A *blockchain* utilizada para o desenvolvimento foi a *Ethereum Blockchain*[1], uma plataforma descentralizada com a capacidade da utilização de contratos inteligentes, fundada em 2014, financiada como um projeto de *crowdfunding*. Apesar disso, a aplicação não é presa a essa determinada *blockchain*, podendo ser utilizada qualquer outra que consiga fazer o uso de contratos inteligentes. Para facilitar o ambiente de desenvolvimento da *blockchain*, é utilizado a ferramenta *hardhat*[3], que faz a criação de um ambiente simulado da *blockchain* localmente, permitindo execuções e testes de maneira mais controlada.

O *smart contract* foi programado utilizando a linguagem de programação de alto nível orientada a objetos *Solidity*[10], que tem foco na implementação de contratos inteligentes para várias *blockchains*. Para contribuir na construção do contrato, é usada a biblioteca *OpenZeppelin*[8], que possui modelos básicos para funcionalidades comuns em contratos.

Além dessas tecnologias mencionadas, há o uso do framework *Moralis*[5], que provê ferramentas para facilitar a integração *web3* com o navegador, além de oferecer serviços de armazenamento utilizando o protocolo *IPFS* (*InterPlanetary File System*).

2.3 Visão geral da arquitetura

A comunicação da interface com o processo interno pode ser visualizada através da Figura 1. O usuário interage com a aplicação *web* (cliente). Ao realizar ações, a interface utiliza das ferramentas do *framework Moralis*[5] para integrar a carteira *Metamask*[4] no navegador. A partir disso, é feita a comunicação com a rede da *blockchain*, consultando o contrato inteligente para executar os algoritmos selecionados. Todo o processo e resultado da ação é armazenada na *blockchain*. Além disso, é também utilizado o framework *Moralis*[5] para uso da base de dados descentralizada para armazenamento das imagens do histórico acadêmico, assim como algumas informações relacionadas.

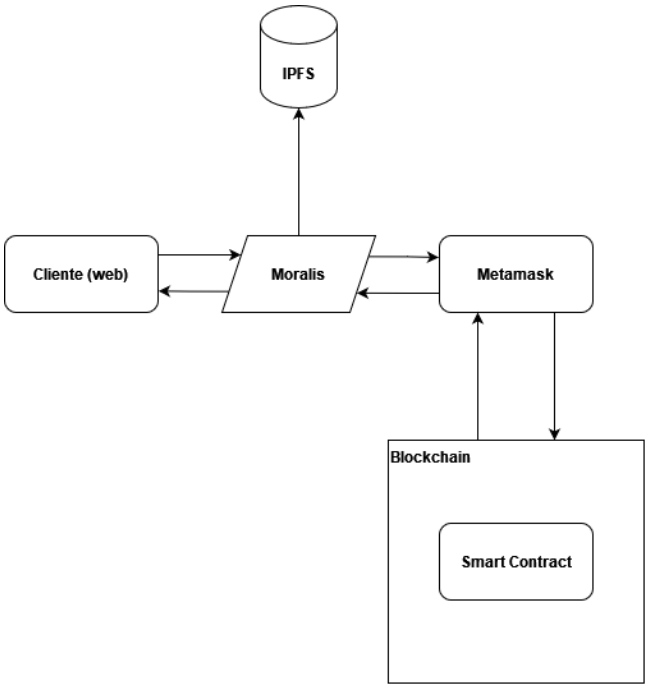

Figura 1: Visão geral da arquitetura da aplicação descentralizada, evidenciando as comunicações feitas com cada tecnologia.

A aplicação possui sua arquitetura baseada em 3 níveis de acesso: Universidade; Professor; e Estudante. Cada nível de acesso pode realizar um conjunto de ações selecionadas. Esses níveis estão presentes tanto na parte da interface como também no processo interno. No *frontend*, há diferentes visualizações de tela para cada nível de acesso, enquanto no *backend*, são usados papéis ligados a cada endereço de carteira que se comunica com o contrato inteligente, dando permissão a apenas um determinado conjunto de algoritmos de acordo com seu papel. O nível de acesso de Universidade funciona como administradora do nível Professor, podendo conceder e revogar esse papel para outros usuários.

As principais funções permitidas de acordo com cada nível de acesso são listadas na tabela 1. Podemos visualizar que funções com

Tabela 1: Funções e nível de acesso

Função	Nível de acesso
Criação de currículo (disciplinas)	Universidade
Vínculo de professores	Universidade
Adição de estudante (NFT)	Universidade
Transferência de estudante (NFT)	Universidade
Cadastro de Notas	Professor
Visualizar histórico	Estudante
Matrícula	Estudante

foco mais administrativo estão voltadas ao papel da Universidade, enquanto ao Professor há o cadastro e armazenamento do histórico do aluno, e ao Estudante há a permissão de visualização do histórico e matrícula em disciplinas.

Os dados armazenados na *blockchain* são feitos através do contrato inteligente. Na Figura 2 é apresentado um diagrama de classes do contrato programado, excluindo métodos importados dos modelos de contratos disponibilizados pelo *OpenZeppelin*[8]. Nela podemos visualizar alguns dos métodos e estruturas necessárias que permitem aos usuários realizarem ações de acordo com seu nível de acesso.

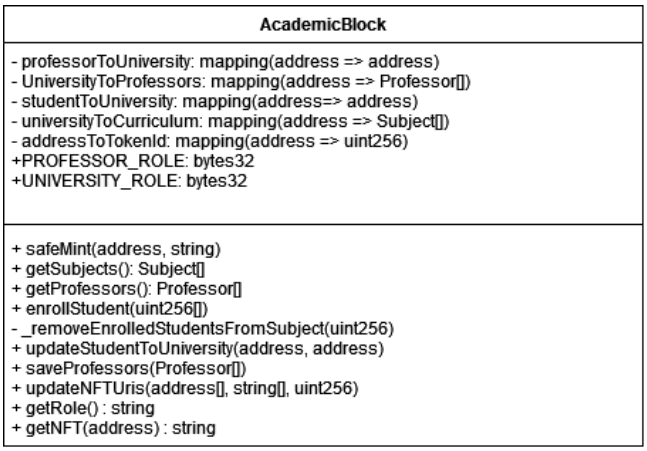

Figura 2: Diagrama de classes do contrato inteligente programado usando Solidity.

3 SISTEMA EM USO

Nesta seção serão apresentadas as telas e funcionalidades de acordo com cada nível de acesso da aplicação.

O contrato inteligente pode ser implantado em qualquer rede de *blockchain* que tenha suporte para o mesmo. Para o desenvolvimento deste trabalho, foi utilizado uma rede local simulada da *Ethereum Blockchain*[1] provida pela biblioteca *hardhat*[3]. Ao iniciar essa rede, é disponibilizado carteiras com dez mil *ETH* cada, criptomoeda utilizada na rede para pagar os custos da comunicação com o contrato inteligente (*GAS*). Tais custos são utilizados apenas em transações que inserem dados na *blockchain*. Métodos que fazem consultas apenas de visualização de dados não possuem custo.

Após a inicialização da rede, é feito a publicação do contrato inteligente na rede, por um *script* feito com a biblioteca *hardhat*[3]. Todo contrato inteligente possui um "dono", e no caso da rede local, esse dono se torna automaticamente o primeiro endereço provido pela biblioteca. Na construção do contrato, também é dado ao dono o papel de universidade.

Ao entrar na aplicação, o usuário é instruído a fazer a conexão da aplicação com a carteira digital, possibilitando a comunicação com a rede *blockchain*.

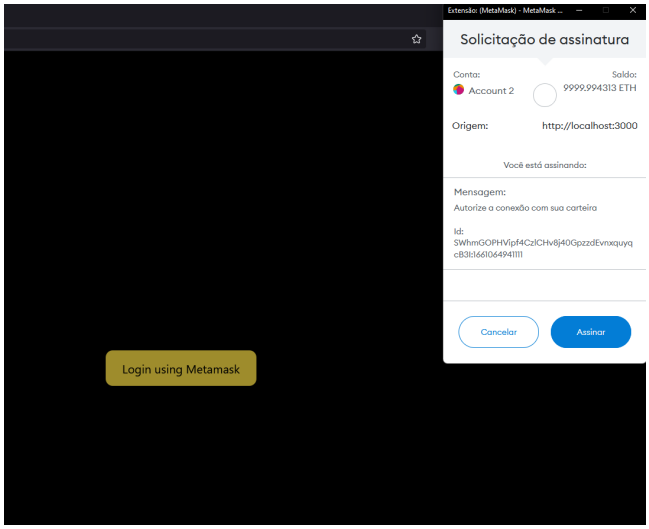

Figura 3: Página inicial de conexão da aplicação com a carteira digital.

Na Figura 3, o usuário confirma a assinatura autorizando a conexão da aplicação. Após isso, é feita a comunicação com o contrato para adquirir o papel relacionado à aquela carteira. Conforme o papel associado, o usuário é redirecionado para suas telas apropriadas.

Uma tela comum a todos os níveis de acesso é a seção de Perfil. O usuário pode alterar seu nome, imagem, e-mail e telefone de contato. A Figura 4 mostra a tela da aplicação. Todas essas informações são salvas no banco de dados providos pelo framework *Moralis*[5].

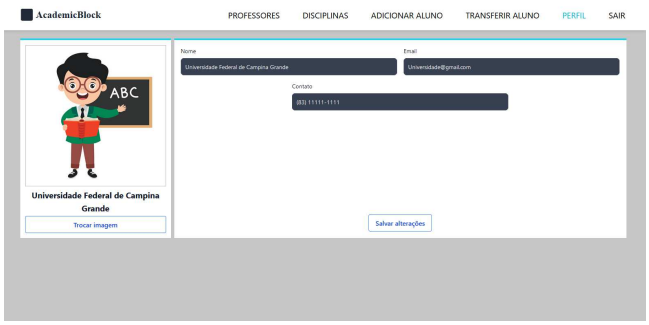

Figura 4: Página de perfil da aplicação.

3.1 Universidade

As telas associadas a uma universidade possuem visualizações e ações voltadas a parte administrativa. A tela inicial, apresentada na Figura 5 é um painel de controle evidenciando todas as funcionalidades permitidas para a sessão.

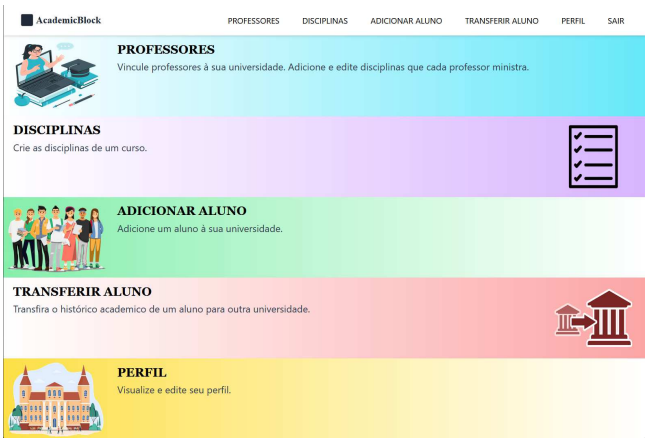

Figura 5: Página inicial de uma carteira com papel de Universidade, evidenciando as ações possíveis.

Ao selecionar a opção "Disciplinas", o usuário é direcionado à tela mostrada na Figura 6. Na tela de disciplinas, o usuário pode realizar a criação e remoção das disciplinas oferecidas pelo curso. Para cada disciplina, é possível incluir os campos de: código; nome; quantidade de créditos; carga horária; e se é obrigatória ou optativa. Após todas as edições necessárias, o usuário pode salvar as disciplinas, que serão enviadas a partir do contrato inteligente e salvas na *blockchain*. Um exemplo de *log* da transação na rede pode ser vista na figura 7, evidenciando a chamada ao contato, qual método foi utilizado, de qual endereço, e a quantidade de *GAS* utilizada.

Figura 6: Página de disciplinas, permitindo a edição de disciplinas de um curso.

Na opção "Professores", o usuário pode adicionar professores que serão vinculados à sua universidade. A Figura 8 mostra a tela dessa opção. Nessa tela, são listados todos os professores vinculados à universidade, além de oferecer a opção de remoção e adição de disciplinas ministradas para cada professor. Também é possível realizar a adição de um novo professor, inserindo seu nome,

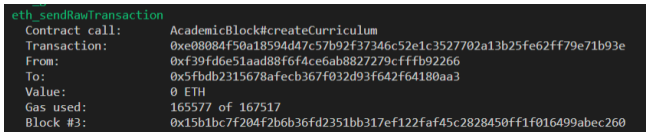

Figura 7: *Log* da rede da *blockchain*, identificando uma transação feita de chamada de contrato.

endereço da carteira e uma disciplina ministrada. Após todas as edições necessárias, o usuário pode salvar os professores, e todas essas informações serão enviadas à *blockchain*.

Figura 8: Página de edição de professores. Permite adição de professores vinculados à universidade.

Para cadastrar um novo aluno no curso, o usuário pode entrar na opção "Adicionar Aluno". A tela referente à esta opção é mostrada na Figura 9. Nessa tela, o usuário precisa inserir as informações referentes ao aluno, como: endereço da carteira; matrícula; curso; currículo; nome; documento de identificação; CPF; cidade; estado; país; e data de nascimento. Após todas as informações preenchidas, é possível cadastrar o histórico, dando início ao NFT do estudante com seu histórico acadêmico. No banco de dados *IPFS*, provido pelo framework *Moralis*[5], é armazenado os dados que correspondem ao histórico acadêmico do estudante, como a imagem gerada do histórico e informações em formato de texto (*metadata*). O *URI* que aponta para esse arquivo no banco de dados é armazenado na *blockchain*, tendo o estudante como seu proprietário. Também é armazenado um vínculo da carteira do estudante com a própria universidade.

Para a opção de transferência de aluno, o usuário pode prover a carteira de um estudante de sua universidade, e a carteira de outra universidade para a qual o estudante será transferido. A Figura 10 mostra a tela dessa opção. Ao realizar a transferência, o vínculo do estudante com a universidade será substituído pela nova.

3.2 Professor

A tela associada ao professor é voltada para o cadastro de notas no histórico acadêmico de um aluno.

A tela inicial do professor consta com uma lista de suas disciplinas ministradas, e para cada uma, os estudantes matriculados naquela disciplina. A Figura 11 mostra a visualização da página. Para cada um dos estudantes, o professor pode cadastrar uma nota, e ao final da inserção de notas dos alunos, pode realizar a ação de

Figura 9: Página de cadastro de aluno. Inicia o histórico acadêmico do aluno na universidade.

Figura 10: Página de transferência de um aluno para outra universidade.

envio de notas. Essa ação irá realizar a edição do *NFT* do estudante, incrementando a disciplina e sua nota no seu histórico. Para isso, é modificado o *URI* que o *NFT* do estudante aponta. Todo esse processo é armazenado na *blockchain*.

Figura 11: Página inicial do professor, permite o cadastro de notas de alunos matriculados nas disciplinas.

3.3 Estudante

Para o estudante, é permitido realizar a matrícula nas disciplinas disponíveis e a visualização do seu histórico acadêmico.

A tela de visualização do histórico é apresentada na Figura 12. A visualização do histórico do aluno é feita adquirindo o *NFT* associado à carteira. O *NFT* apresenta as informações pessoais cadastradas pela universidade, assim como o histórico de disciplinas cadastrado pelos professores.

A tela de matrícula em disciplinas é apresentada na Figura 13. O aluno pode realizar matrículas em disciplinas, tendo a informação

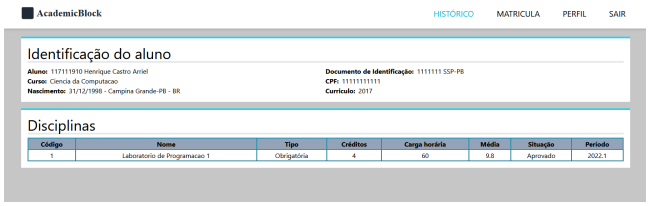

Figura 12: Página inicial do estudante. Permite visualização do seu próprio histórico acadêmico.

salva na *blockchain*, necessária para validar a edição de um professor no *NFT* de um estudante.

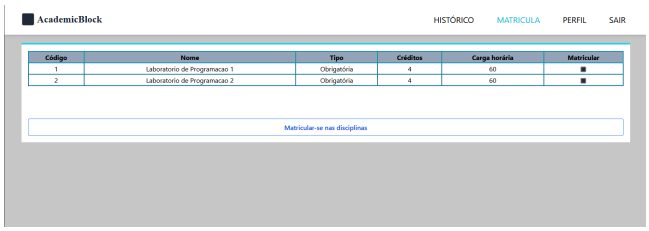

Figura 13: Página de matrículas em disciplinas.

4 AVALIAÇÃO

Visando avaliar a aplicação, foi solicitado a diversos estudantes que utilizassem a aplicação para avaliar seu uso em cada um dos três níveis de acesso.

Após o uso do sistema, o estudante é solicitado a relatar sua experiência de acordo com cada papel, além de funcionalidades que poderiam ser adicionadas ou alteradas. Os relatos de aproximadamente dez estudantes com relação à experiência de uso do sistema foram sumarizados nessa seção.

Para o nível de acesso de Estudante, os usuários relataram ser exatamente o esperado para a visualização do histórico. Porém, para a seção de matrícula, ficaram faltando informações como período da disciplina e professor que a ministra.

Para o nível de acesso como Professor, a avaliação foi positiva, sendo uma página simples que permite realizar o cadastro das notas de forma rápida e prática para cada aluno de suas disciplinas. Uma funcionalidade que poderia ser incrementada é a adição de mais campos de notas para cada aluno, retratando ao fato de a nota final ser resultado de um cálculo das notas de cada estágio da disciplina.

Para o nível de acesso como Universidade, foi relatado que algumas telas poderiam ser alteradas para realizar mais operações de uma vez, como, por exemplo, o cadastro de um aluno. É possível reaproveitar diversos campos do cadastro (período, curso, etc.), ficando necessário apenas as informações pessoais de cada aluno. Essa funcionalidade facilitaria o cadastro de novos ingressantes de forma mais rápida, sem necessitar fazer uma operação para cada aluno. Para a tela de disciplinas, foi-se reclamado a falta de período do curso relacionado à aquela disciplina. Por fim, a tela de "professores"teve, em geral, uma avaliação positiva, constando com todas as opções que eram esperadas.

Desconsiderando os níveis de acesso, uma funcionalidade adicional considerada é a situação das requisições à *blockchain*. Na aplicação atual, essas situações ficam evidenciadas na extensão *Metamask*[4], sendo interessante trazer esse *status* para a própria página do navegador.

5 EXPERIÊNCIA E LIÇÕES APRENDIDAS

O desenvolvimento do sistema relatado no trabalho necessitou do estudo de diversas áreas de conhecimento. Com relação à parte de interface da aplicação, o processo de desenvolvimento seguiu o fluxo comum para essa área, realizando primeiramente a prototipação das telas, seguidas da estruturação do código *HTML* e *CSS* para *design* da tela, após a estruturação, ocorre a integração de funcionalidades mais robustas do componente com *Javascript*, e por fim a integração com o *backend*, através das requisições à *blockchain*.

O maior desafio da parte do *frontend* é a parte de design das telas. A prototipação e uso de *HTML* e *CSS* podem ser passos que necessitam de muita tentativa e erro em desenvolvedores que não são tão acostumados com o processo de desenvolvimento de interface. O uso de novas ferramentas que tem como promessa a agilidade (como *Tailwind CSS*[11] para o *design*, mostram uma curva de aprendizado íngreme que atrasa o desenvolvimento no início, porém após o processo de habituação da ferramenta, entrega um desenvolvimento rápido e controlado.

Com relação ao *backend*, a tecnologia de contratos inteligentes e a *blockchain* ainda é muito nova, sendo um dos principais limitadores com relação à busca de informações e soluções para problemáticas que aparecem ao longo do desenvolvimento. A linguagem de programação *Solidity*, utilizada para a programação dos contratos inteligentes, se mostrou uma grande complexidade para o desenvolvimento por conta das suas limitações, afinal, é uma linguagem muito jovem, além de ser voltada para a *blockchain*, a qual também possui suas próprias limitações. A integração da rede da *blockchain* com a interface é um processo que se mostra difícil à medida que funções mais específicas são necessárias.

A *blockchain*, por mais que possua diversas características vantajosas, entregam uma certa limitação e desvantagens existentes por sua própria estrutura. Uma limitação presente no trabalho apresentado, é com relação aos *NFTs*, sendo a não possibilidade de armazenar dados grandes na própria *blockchain* por conta do custo. Para isso, a solução é armazenar os dados fora da blockchain (*off-chain*), apenas referenciando o local em que está armazenado na *blockchain*. Apesar de ser uma solução válida que permite conectar os dados à *blockchain*, é um ponto de possível falha. Além disso, a característica da *blockchain* de armazenamento permanente também entrega um problema de lixo eletrônico, em que nenhum dado é possível ser realmente deletado da rede. Para aplicações *web*, em que métodos de deleção de conteúdo são comumente entregues, é necessário adaptar-se para essa característica, como, por exemplo, com a utilização de *flags* para evidenciar se um conteúdo foi deletado.

5.1 Trabalhos Futuros

A aplicação desenvolvida é uma prova de conceito usada para demonstrar como construir para aplicações descentralizadas que permitam a interação dos usuários com a tecnologia *blockchain* no âmbito acadêmico. Uma funcionalidade adicional que pode ser implementada na aplicação é a maior administração de diferentes cursos de uma universidade. Para isso, é necessário realizar alterações nas estruturas e métodos do contrato inteligente, por estruturas que mapeiem universidades aos cursos, e cada curso mapeado para suas disciplinas, além de métodos otimizados para realizar buscas para cada uma dessas estruturas. Além disso, também é necessário realizar alterações no *frontend* para suportar essas alterações e apresentar ao usuário mais possibilidades de ações.

A interface da aplicação pode ser utilizada para diversas universidades, como também é possível servir de base para a criação de outras interfaces que usem o mesmo *backend* para cada universidade específica, apenas seguindo as regras do contrato inteligente.

O trabalho proposto também pode servir de base para outras aplicações em diferentes setores que possam se beneficiar das características promovidas pela tecnologia *blockchain*, apenas fazendo alterações no contrato inteligente e utilizando diferentes interfaces para se comunicar com a rede.

Por fim, o contrato inteligente pode ser implantado em alguma rede de *blockchain* ampla, estando disponível para qualquer usuário que tenha acesso àquela rede. Para isso, é necessário pagar os custos necessários de utilização da *blockchain* (*GAS*), que podem variar de acordo com métodos utilizados e com o preço da criptomoeda utilizada para as transações. Os custos de uso da aplicação para cadastrar as informações na *blockchain* são baixos, porém é necessária uma análise mais aprofundada do custo de uso durante um determinado período para uma universidade.

REFERÊNCIAS

[1] Ethereum 2015. Plataforma descentralizada capaz de executar contratos inteligentes usando *blockchain*. Retrieved 21 de agosto de 2022 from https://ethereum.org/
[2] Figma 2016. Editor gráfico de vetor e prototipagem de telas. Retrieved 21 de agosto de 2022 from https://www.figma.com/
[3] hardhat 2019. Ambiente de desenvolvimento para Ethereum. Retrieved 21 de agosto de 2022 from https://hardhat.org/
[4] Metamask 2016. Carteira de criptomoeda de software para interação com blockchain Ethereum. Retrieved 21 de agosto de 2022 from https://metamask.io/
[5] Moralis 2021. Serviço web3 provedor de funcionalidades para construção de dApps. Retrieved 21 de agosto de 2022 from https://moralis.io/
[6] NextJs 2016. Framework de React para produção. Retrieved 21 de agosto de 2022 from https://nextjs.org/
[7] OpenSea 2017. O primeiro e maior mercado de NFTs. Retrieved 21 de agosto de 2022 from https://opensea.io/
[8] OpenZeppelin 2017. Modelos de contratos inteligentes escritos na linguagem de programação Solidity. Retrieved 21 de agosto de 2022 from https://www.openzeppelin.com/
[9] React 2013. Uma biblioteca JavaScript para criar interfaces de usuário baseado em componentes. Retrieved 21 de agosto de 2022 from https://pt-br.reactjs.org/
[10] Solidity 2014. Linguagem de programação orientada a objetos para implementação de contratos inteligentes na *blockchain*. Retrieved 21 de agosto de 2022 from https://soliditylang.org/
[11] Tailwind 2019. Framework de CSS para compor e criar design de telas diretamente no HTML. Retrieved 21 de agosto de 2022 from https://tailwindcss.com/