



UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA
SISTEMAS DE INFORMAÇÃO

RENAN INACIO

**APROVAÇÕES ACADÊMICAS PÚBLICAS E AUDITÁVEIS COM O USO DE
BLOCKCHAIN IMPLEMENTADAS NO PADRÃO DE SOULBOUND TOKEN**

**Florianópolis
2022**

RENAN INACIO

**APROVAÇÕES ACADÊMICAS PÚBLICAS E AUDITÁVEIS COM O USO DE
BLOCKCHAIN IMPLEMENTADAS NO PADRÃO DE SOULBOUND TOKEN**

Trabalho de Conclusão de Curso apresentado como objetivo parcial à obtenção do grau Bacharel em Sistemas de Informação, pela Universidade Federal de Santa Catarina.

Orientador: Alex Sandro Roschildt Pinto

Florianópolis
2022

RENAN INACIO

**APROVAÇÕES ACADÊMICAS PÚBLICAS E AUDITÁVEIS COM O USO DE
BLOCKCHAIN IMPLEMENTADAS NO PADRÃO DE SOULBOUND TOKEN**

Trabalho de conclusão de curso apresentado como parte dos requisitos para
obtenção do grau de Bacharel em Sistemas de Informação.

Orientador:

Prof. Dr. Alex Sandro Roschildt Pinto

Banca examinadora:

Profa. Dra. Carla Merkle Westphall

Prof. Dr. Jean Everson Martina

Florianópolis
2022

RESUMO

A tecnologia de registro descentralizada (blockchain) foi criada por Nakamoto em 2009, e proporciona um alto nível de confiança e transparência em transações financeiras por meio do Bitcoin. Com o advento da rede Ethereum, este nível de confiança se estendeu para códigos imutáveis de aplicações descentralizadas, cujas transações são validadas por milhares de pontos (nodes) descentralizados pelo mundo. No lugar de organizações centralizadas de poder e necessidade de confiança nas mesmas, precisamos apenas confiar no código. Esta proposta permitiu a criação de tokens intercambiáveis entre si: dois tokens possuem as mesmas propriedades e substituíveis sem prejuízo da qualidade ou valor, o que chamamos de fungível. Por outro lado, precisamos representar itens não-fungíveis, como obras de arte, certificados de propriedade ou um diploma universitário. Por estes motivos, foi criado o conceito de tokens não fungíveis - em inglês, NFTs. Utilizando o conceito de tokens não fungíveis e não transferíveis - chamados de Soulbound Tokens ou SBTs - foi implementada uma aplicação descentralizada (dApp) para geração de tokens públicos e auditáveis no contexto de aprovações acadêmicas. A ideia central foi promover o acesso a 4 personas e seus casos de uso: universidade, professores, alunos e verificadores. A implementação do Smart Contract foi feita na rede principal da Polygon (MATIC) tendo um custo por transação inferior a \$0.02, junto de uma aplicação cliente para interagir com o mesmo.

Palavras-chave: Blockchain, Smart Contract, Soulbound Tokens, Aplicação descentralizada, Polygon.

ABSTRACT

The Decentralized ledger technology (blockchain) was created by Nakamoto in 2009, and provides a high level of trust and transparency in financial transactions with Bitcoin. With the advent of the Ethereum network, this level of trust was extended to immutable codes of decentralized applications, whose transactions are validated by thousands of decentralized points (nodes) around the world. Instead of centralized organizations of power and the need of trust in them, we only need to trust in the code. This proposal allowed the creation of interchangeable tokens: two tokens have the same properties and are interchangeable without loss of quality or value, which we call fungible. On the other hand, we need to represent non-fungible items, like art, ownership certificates or a university degree. For these reasons, the concept of non-fungible tokens - (NFTs) was created. Using the concept of non-fungible and non-transferable tokens - called Soulbound Tokens or SBTs - a decentralized application (dApp) was implemented to generate public and auditable tokens in the context of academic approvals. The central idea was to promote access to 4 personas and their use cases: university, professors, students and verifiers. The implementation of the Smart Contract was carried out on Polygon's main network (MATIC) with a cost per transaction of less than \$0.02, along with a client application to interact with it.

Palavras-chave: Blockchain, Smart Contract, Soulbound Tokens, Decentralized Application, Polygon.

LISTA DE ILUSTRAÇÕES

Figura 1 – Diferenças entre Web1, Web2 e Web3	10
Figura 2 – Esquema de passaporte sanitário proposto pelo autor	17
Figura 3 – Pesquisa realizada com os alunos sobre conhecimentos em NFTs	19
Figura 4 – Participantes e suas interações	21
Figura 5 – Proposta de tela inicial	21
Figura 6 – Proposta de tela de verificação de Soulbounds Tokens (matérias validadas) do usuário	22
Figura 7 – Transação de implementação do contrato	26
Figura 8 – Transação de criação de token	27
Figura 9 – Tela inicial da aplicação cliente	27
Figura 10 – Tela de verificação de tokens da carteira atual	28
Figura 11 – Tela de verificação de tokens	28

LISTA DE TABELAS

Tabela 1 – Endereços utilizados	25
---	----

LISTA DE ABREVIATURAS E SIGLAS

ABNT	Associação Brasileira de Normas Técnicas
DID	Decentralized Identity
DLT	Distributed Ledger Technology
MEC	Ministério da Educação
NFT	Non Fungible Token
P2P	peer-to-peer
SBT	Soulbound Tokens
SUS	Sistema Único de Saúde
UFSC	UNIVERSIDADE FEDERAL DE SANTA CATARINA

SUMÁRIO

1	INTRODUÇÃO	10
2	OBJETIVOS	12
2.1	Objetivo geral	12
2.2	Objetivos específicos	12
3	FUNDAMENTAÇÃO	13
3.1	Bitcoin	13
3.2	Ethereum	13
3.2.1	Smart Contracts	13
3.2.2	Layer 1	14
3.2.3	Layer 2	14
3.2.4	Non-fungible Tokens (NFTs)	14
3.2.5	Soulbound Tokens (SBTs)	15
4	TRABALHOS RELACIONADOS	16
4.1	Blockchain-Based Academic Record System	16
4.2	Um comprovante de vacinação baseado em Identidade Auto-Soberana, Blockchain e Provas de Zero Conhecimento	17
4.3	Estudo da aplicação de estrutura blockchain com Proof of Stake para arquivamento de documentos com registro no tempo	17
4.4	NFT Student Teacher Incentive System (NFT-STIS)	18
5	PROPOSTA	20
5.1	Participantes	20
5.2	Front-end	21
5.3	Smart Contract	22
6	IMPLEMENTAÇÃO	25
6.1	Endereços	25
6.2	Deploy	25
6.3	Custos	26
6.4	Frontend	27
7	CONSIDERAÇÕES FINAIS E TRABALHOS FUTUROS	29
8	REFERÊNCIAS	30
9	APÊNDICES	32

9.1	Smart Contract	32
9.2	Frontend	34
9.3	Artigo	38

1 INTRODUÇÃO

A internet como conhecemos hoje pode ser dividida em 3 categorias: *Web1*, *Web2* e *Web3*, cada qual com sua particularidade e foco. Chamamos de *Web1* nosso primeiro contato com a internet, onde o conteúdo era estático e sem muita interação e gerado previamente por uma organização centralizada. Com o avanço da tecnologia e das interações entre consumidor e empresa, surgiu o conceito que hoje chamamos de *Web2*, trazendo o usuário para dentro do processo criativo e o tornando criador de seu próprio conteúdo. No conceito de *Web3*, no lugar de organizações centralizadas de poder e necessidade de confiança nas mesmas, precisamos apenas confiar no código, cujas transações são validadas por milhares de pontos (nodes) descentralizados pelo mundo (PATIL, 2021).

Decentralized applications (dApps) are Web3 implemented and available now. DApps are applications that exist and use a blockchain or P2P network of computers instead of a single computer, and outside view and control of a single authority. In the context of cryptocurrencies, dApps operate on a blockchain network in a public, open source, low-level environment and are free from the control and interference of any single administrator (PATIL, 2021, p. 1).

A tecnologia de registro descentralizada (em inglês, *Distributed Ledger Technology - DLT*), ou simplesmente *Blockchain* - criada por Nakamoto em 2009 - proporciona um alto nível de confiança e transparência em transações financeiras (*Bitcoin*) no mundo digital. Com o advento da rede Ethereum, foi possível criar códigos imutáveis e validados por uma rede descentralizada, paralela e isolada, sem a necessidade de um mecanismo central, o que chamamos de *Smart Contract* (DUCRÉE; GRAVITT; WALSH; BARTLING; ETZRODT; HARRINGTON, 2020).

Figura 1 – Diferenças entre Web1, Web2 e Web3



<https://medium.com/cardstack/combining-web2-web3-97201bb4c571>

Mesmo com o advento da *Web3* em transações financeiras pelo mundo, a falta de mecanismos de verificação de identidade e reputação força os usuários a depender de estruturas centralizadas dentro da *Web2*. *Web3* hoje se concentra em expressar ativos financeiros e transferências descentralizadas, quando deveríamos incluir também a inclusão de redes de confiança em uma sociedade (WEYL; OHLHAVER; BUTERIN, 2022). O conceito de “**Decentralized Society**” (**DeSoc**) originado por Buterin (2022) foca na criação de *tokens* públicos e não-transferíveis como mecanismo de identidade social descentralizada e auditável, referenciados como “**Soulbound Tokens**” (**SBTs**).

Imagine a world where most participants have Souls that store SBTs corresponding to a series of aliations, memberships, and credentials. For example, a person might have a Soul that stores SBTs representing educational credentials, employment history, or hashes of their writings or works of art. In their simplest form, these SBTs can be “self-certied,” similar to how we share information about ourselves in our CVs (WEYL; OHLHAVER; BUTERIN, 2022).

Um ecossistema de SBTs abre alternativa de *bottom-up* e resistente à censura para sistemas de crédito comerciais e sociais. SBTs que representam credenciais acadêmicas, histórico de trabalho e os contratos de aluguel podem servir como um registro histórico descentralizado e público. Empréstimos e linhas de crédito podem ser representados como SBTs intransferíveis (porém revogáveis), até que sejam quitadas e subsequentemente “queimadas”, ou melhor ainda, substituídas por prova de quitação. De forma intrínseca, a intransferibilidade impede os usuários de ocultarem informações (WEYL; OHLHAVER; BUTERIN, 2022).

2 OBJETIVOS

2.1 Objetivo geral

Acelerar a adoção da Web3 no cotidiano universitário, permitindo o gerenciamento de aprovações de disciplinas do curso de Sistemas de Informação da Universidade Federal de Santa Catarina (UFSC) através de um Smart Contract, em uma rede descentralizada, pública e segura.

2.2 Objetivos específicos

- Codificar um Smart Contract com padrão SBT (Soulbound Token)
- Implementar um Smart Contract em uma rede (network) que possua custos baixos de operação (menor que \$0.02 por transação)
- Possibilitar a validação dos dados publicamente através de uma página web

3 FUNDAMENTAÇÃO

Neste capítulo são descritas definições e os conceitos necessários para melhor compreensão do projeto.

3.1 Bitcoin

O Bitcoin (BTC) é a mais conhecida das moedas digitais. Foi o primeiro sistema de pagamento global totalmente descentralizado. Teve seu surgimento na crise financeira global em 2008, com o objetivo de substituir o dinheiro de papel, além de eliminar a necessidade da presença de bancos para intermediar operações financeiras. Segundo o site Bitcoin.org, a primeira especificação do Bitcoin e prova de conceito foram publicados em um artigo assinado por Satoshi Nakamoto, pseudônimo de um programador (ou grupo de programadores) até hoje não identificado. Ele inventou a lógica de funcionamento do blockchain, sistema que possibilitou a existência do Bitcoin. No artigo, Nakamoto estabeleceu que haverá no máximo 21 milhões de bitcoins em circulação. Estima-se que a última moeda será minerada no ano de 2140. As principais vantagens da Criptomoeda Bitcoin são: Liberdade de pagamento, segurança, transparência, grau de satisfação e volatilidade (NAKAMOTO, 2008).

3.2 Ethereum

Ethereum é uma tecnologia que engloba dinheiro digital, pagamentos globais e investimentos. O Ethereum é uma plataforma descentralizada focada na execução dos chamados “contratos inteligentes”: operações feitas automaticamente quando certas condições são cumpridas. Além disso, ela é usada nas operações de sua própria moeda, o Ether, e também de outros ativos que usam este sistema. Diferente do Bitcoin, o Ether não foi criado para ser uma moeda digital, mas sim um ativo para recompensar os desenvolvedores que usam a plataforma Ethereum para seus projetos. Mesmo assim, o Ethereum é uma das três moedas digitais mais negociadas do mundo (INFOMONEY, 2022)

3.2.1 Smart Contracts

Smart contracts (contratos inteligentes) são programas que se executam de forma automática, assim que certas condições acordadas previamente pelas partes são atendidas. Os smart contracts rodam em uma blockchain, e todas as cláusulas contidas neles são gravadas nessa rede. Uma vez que as regras, obrigações e penalidades são inseridas, os contratos são executados de forma automática conforme aquilo que foi combinado. O papel da blockchain, portanto, é garantir que esses acordos aconteçam de forma segura e verificável, e sem manipulação para benefício próprio. Eles podem ser usados tanto em

aplicativos de finanças descentralizadas como em outras áreas, a exemplo do mercado imobiliário e do varejo online. Nem toda criptomoeda possui smart contract, e não são todas as blockchains que permitem a criação deles. O próprio Bitcoin, por exemplo, não tem contratos inteligentes em sua camada base (REY, 2019).

3.2.2 Layer 1

A Layer 1 refere-se a uma rede de base, como Bitcoin, BNB Chain ou Ethereum, e sua infraestrutura subjacente. São capazes de validar e finalizar transações sem a necessidade de outra rede. A Layer 1 é a camada 1 da Ethereum. Em outras palavras, no ecossistema descentralizado, uma rede de primeira camada se refere a um Blockchain nativo. Então, a primeira camada é o termo usado para descrever a arquitetura principal e subjacente do Blockchain. De maneira geral, o principal problema da primeira camada é que à medida que os Blockchains crescem em número de usuários, começam a lidar com mais transações, culminando em preços de taxas de transações que podem subir a níveis insustentáveis. (BINANCE, 2022).

3.2.3 Layer 2

Ethereum Layer 2 é a segunda camada, também chamada de Layer-2 ou rede secundária, é uma estrutura adjacente (ou um protocolo secundário) criada acima de um sistema Blockchain existente, em que o objetivo principal é aumentar a velocidade de transações e resolver as dificuldades de escalabilidade enfrentadas pela rede principal. A Layer 2 é um termo coletivo para designar soluções projetadas para ajudar a dimensionar aplicativos, manipulando transações fora da cadeia principal do Ethereum (Layer 1). As velocidades de transação diminuem quando a rede está ocupada, dificultando a experiência do usuário para certos tipos de dapps.

3.2.4 Non-fungible Tokens (NFTs)

O conceito de Blockchain permitiu a criação de tokens intercambiáveis entre si: dois tokens possuem as mesmas propriedades e substituíveis sem prejuízo da qualidade ou valor, o que chamamos de fungível. Por outro lado, existem objetos não-fungíveis que são impossíveis de representar pela proposta base de tokens digitais, como, por exemplo, uma obra de arte, um certificado de propriedade ou mesmo um diploma de faculdade, que possuem propriedades únicas. Por estes motivos, foi criado o conceito de tokens não fungíveis - em inglês, NFTs. (ELMESSIRY, A.; ELMESSIRY, M.; BRIDGESMITH, Larry. 2021).

3.2.5 Soulbound Tokens (SBTs)

Vitalik Buterin é o homem por trás da rede Ethereum, e enquanto jogava o jogo World of Warcraft, Buterin teve a ideia de tokens intransferíveis. Embora tokens e NFTs sejam transferíveis como ativos, Buterin estava interessado em usá-los como um tipo de “prova”. Pouco tempo depois, Buterin publicou um artigo detalhando uma sociedade descentralizada que ele descreveu como um espaço aumentado baseado em tokens “vinculados à alma”. Um Soulbound Token (SBT) é um token NFT que está permanentemente vinculado ao usuário. Por serem únicos e intransferíveis, os SBTs oferecem aplicações benéficas em ambientes digitais e do mundo real. Os tokens em formato SBT permitem que uma pessoa construa sua identidade digital, tirando o foco da blockchain em representar ativos financeiros. Isso pode mudar fundamentalmente as sociedades digitais e da vida real (BYBIT, 2022).

4 TRABALHOS RELACIONADOS

Neste capítulo, são apresentados os trabalhos similares ao presente trabalho de conclusão de curso.

4.1 Blockchain-Based Academic Record System

O artigo propõe automatizar e reduzir os casos de fraudes relacionados a emissão de certificados de graduação e pós-graduação utilizando a tecnologia Blockchain, com Smart Contracts implementados na rede Ethereum. O autor propõe a criação de 4 personas:

1. Regulatory Agency (RA)

Responsável por regular as instituições de ensino em âmbito nacional, avaliar e aprovar os currículos de cada curso e supervisionar as aprovações de novos certificados. Esta persona é responsável por implementar o Smart Contract que atua como interface entre os participantes. No contexto nacional, esta persona pode ser representada pelo Ministério da Educação (MEC).

2. Higher Education Institute (HEI)

Podem ser representados pelas universidades e institutos de educação. São responsáveis por operar os Smart Contracts implementados pela RA, registrar novos Students e criar diplomas para os mesmos quando completados os requisitos.

3. Student

Pode ser representado por qualquer aluno matriculado em um curso superior de graduação ou pós-graduação. Um Student necessita comparecer as aulas e receber os créditos de acordo com o desempenho, que é registrado pelas HEIs.

4. Verifier

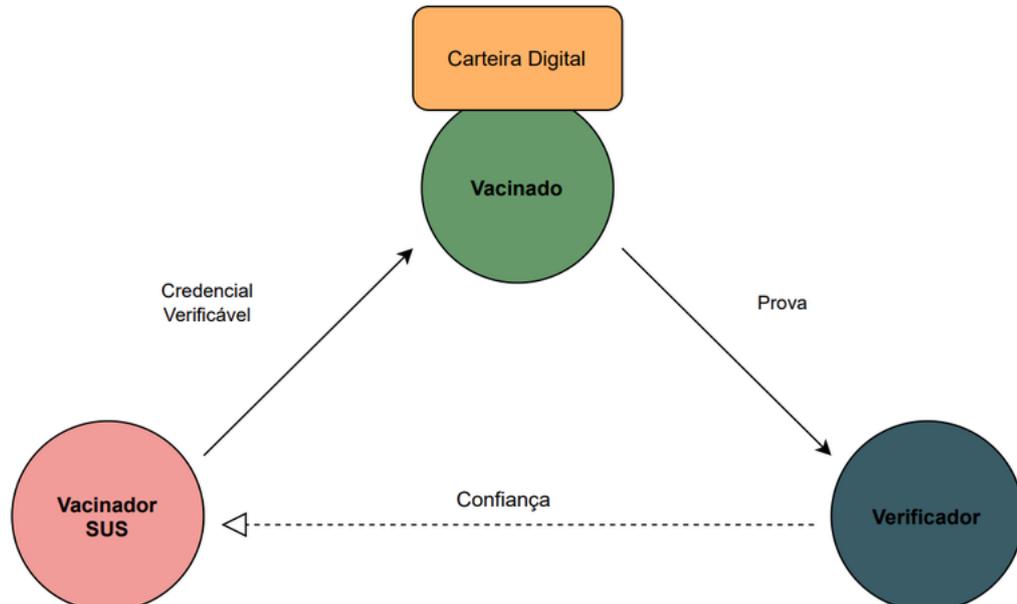
Representa qualquer indivíduo ou entidade interessada em validar os documentos e créditos de um Student; Também pode verificar o status das HEIs para decidir se as informações de um Student são confiáveis.

4.2 Um comprovante de vacinação baseado em Identidade Auto-Soberana, Blockchain e Provas de Zero Conhecimento

O artigo se propõe a implementar uma prova de conceito de certificado de vacinação de covid-19 usando Provas de Zero Conhecimento (ZKProof) para manter a privacidade do usuário. A discussão sobre identidade auto-soberana inicia com a ideia que o usuário deveria possuir seus dados, possuir sua identidade digital e apresentar estes dados somente quando necessário. Esta identidade descentralizada (DID) foi implementada pelo autor utilizando o software Hyperledger Indy, uma ledger distribuída criada para o gerenciamento e implementação de identidades auto-soberanas, em conjunto do software Hyperledger Ursa, que gerencia as credenciais e utiliza as premissas de ZKProof.

Neste contexto, o autor implementa um “passaporte covid”: um emissor, no nosso caso, alguma entidade relacionada ao SUS, emite uma credencial verificável para o paciente vacinado. Esse paciente armazena tal credencial em uma carteira digital, e por meio dessa carteira ele é capaz de apresentar as provas de sua vacinação para as partes interessadas. O verificador confia nas provas pois ele confia no emissor.

Figura 2 – Esquema de passaporte sanitário proposto pelo autor



Barros, Maurício de Vasconcelos (2021)

4.3 Estudo da aplicação de estrutura blockchain com Proof of Stake para arquivamento de documentos com registro no tempo

O artigo propõe reduzir os casos de fraude e aumentar a escalabilidade em transferências de documentos (comprovações de imóveis, decisões jurídicas, até processos

dentro das empresas), criando um mecanismo de validade de uma assinatura digital. Para isso foi criada uma solução carimbo do tempo que comprova que em determinada data o documento foi assinado, registrando os dados em um banco de dados distribuído baseado em blockchain. Esta solução garante o armazenamento dos arquivos em nuvem e de forma confiável, eliminando problemas de fraude por alterações.

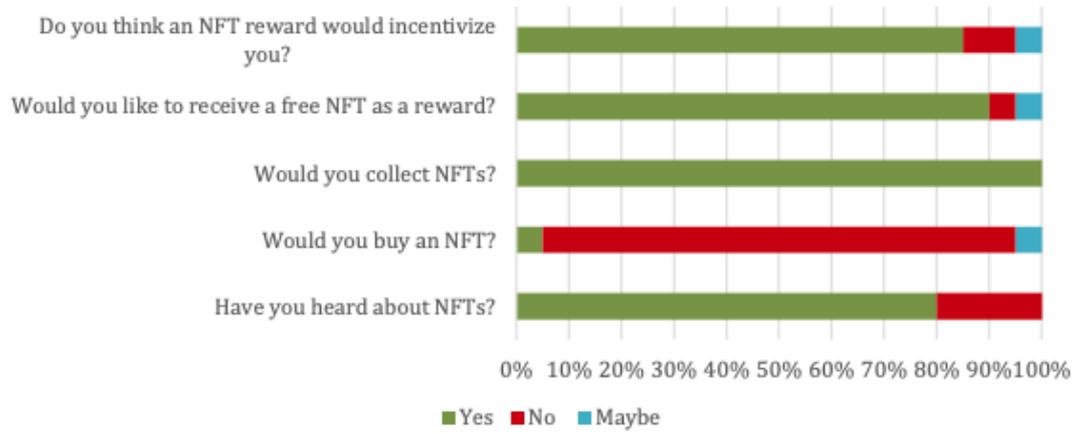
A utilização de blockchains privadas como Hyperledger Fabric cumpre a maioria dos requisitos, datação confiável, sem custo de transação e a utilização de Proof of Stake. Por outro lado foi possível confirmar que a utilização de blockchains de serviço público podem garantir todos os requisitos do caso de uso proposto específico, o autor entende que para determinadas situações é possível a utilização mesmo com as arquiteturas aqui analisadas.

4.4 NFT Student Teacher Incentive System (NFT-STIS)

O processo de avaliação e incentivos ao aluno são elementos críticos do processo educacional, sendo ingredientes-chave para melhorar a aquisição de conhecimento. Com a pandemia global iniciada em 2020, se iniciou um rápido processo de digitalização do processo de ensino, gerando uma crise nas instituições educacionais, e forçando cada vez mais a inovação no setor. O autor propõe uma plataforma de gamificação do conhecimento baseada em blockchain com o uso de tokens não-fungíveis (NFTs), focado no processo de criação conjunta entre educador e estudante. A persona educador representa uma pessoa designada pela instituição de ensino a promover atividades educacionais, tendo como responsabilidade manter uma carteira no blockchain para receber e controlar as NFTs geradas pela instituição. Ao fim de cada período, o professor é responsável por receber as NFTs geradas e transferir para os respectivos alunos.

O autor também realizou uma pesquisa com os 111 estudantes acerca dos conhecimentos dos mesmos sobre NFTs e gamificação. Os resultados iniciais mostram um grande interesse por parte dos alunos sobre o sistema de incentivo baseado em NFTs proposto pelo artigo, demonstrando a eficácia do sistema.

Figura 3 – Pesquisa realizada com os alunos sobre conhecimentos em NFTs



Elmissiry, A.; Elmissiry M.; Bridgesmith, Larry. (2021)

5 PROPOSTA

Neste capítulo, propomos a criação de um Smart Contract transparente e seguro para gerenciamento de aprovações de disciplinas do curso de Sistemas de Informação da Universidade Federal de Santa Catarina (UFSC). Primeiramente apresentamos os participantes envolvidos e como os mesmos interagem com o projeto, apresentamos também o esboço da aplicação cliente (front-end) e o *smart contract* proposto para atingir os objetivos.

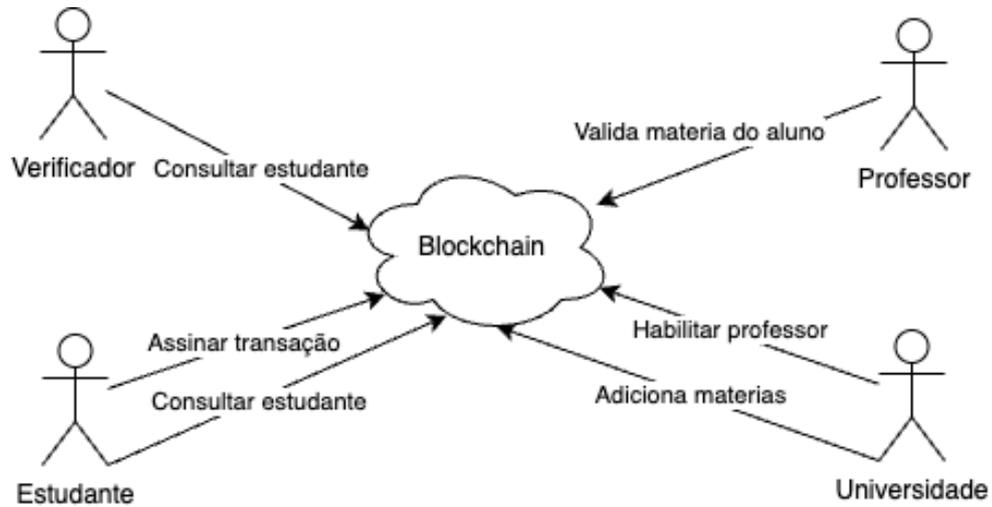
5.1 Participantes

A proposta inclui 4 participantes. A Figura 3 exibe as principais interações de cada participante na rede. O primeiro participante é a Universidade, ela é responsável por habilitar novos participantes Professores para interagirem na rede, bem como adicionar novas matérias a serem validadas e adicionadas futuramente á carteira do Estudante.

O participante Estudante consegue consultar as matérias já validadas para sua matrícula e suas informações. O participante Estudante também pode assinar transações para verificar sua identidade ao participante Validador. Este terceiro participante chamado de Validador representa qualquer usuário cujo fim é validar um cadastro de um estudante (um empregador, imprensa, outras universidades, etc.).

Por último temos o participante Professor que, após devidamente cadastrado pela Universidade, pode validar as matérias habilitadas para o mesmo nos cadastros (carteiras) dos Estudantes. Todas as interações são salvas na rede Blockchain permitindo futura auditoria dos dados.

Figura 4 – Participantes e suas interações



Elaborado pelo autor

5.2 Front-end

Como parte do projeto, será elaborada uma página Front-end para cobrir os casos de usos propostos na seção 5.1, preferencialmente disponibilizada via IPFS. Como se trata de uma aplicação descentralizada (DApp), os dados serão acessíveis após o usuário conectar sua carteira de criptomoedas utilizando seu aplicativo preferido (MetaMask, Coinbase Wallet, Rainbow Wallet, etc.).

Ao entrar na aplicação, o usuário poderá verificar os tokens associados a sua própria conta (no caso de ser um estudante) ou verificar os tokens cadastrados para a conta de outra pessoa (no caso de ser um verificador).

Figura 5 – Proposta de tela inicial



Elaborado pelo autor

Ao clicar em uma *Verificar meus tokens*, será possível visualizar os *Soulbound Tokens* (matérias validadas) que o usuário possui, a data de verificação, o usuário que efetuou a criação do token (Professor ou Coordenadoria) e uma seção para visualizar os dados da transação e conferir a autenticidade do mesmo.

Figura 6 – Proposta de tela de verificação de Soulbounds Tokens (matérias validadas) do usuário

Tokens verificados				
Token ID	Referencia	Verificado em	Verificado por	Detalhes
ebf72c84-72df	INE0001	20/07/2022 14:34:01	Professor 1	👁
671236gs-ghs51	INE0002	22/07/2022 16:47:12	Professor 2	👁

Elaborado pelo autor

5.3 Smart Contract

Para atender a proposta do trabalho, foi codificado o Smart Contract abaixo. Foi definida uma estrutura para o token (SoulboundToken) contendo:

- tokenId: ID único para aquele token;
- tokenReference: Uma referência externa para o token, neste caso sendo o código da disciplina;
- sender: Endereço da carteira do professor que criou o token;
- receiver: Endereço da carteira do aluno que recebeu o token;
- timestamp: Data do bloco que contém a transação;

Código 5.1 – Código solidity 1 - Struct

```

1 struct SoulboundToken {
2   string tokenId;
3   string tokenReference;

```

```

4  address sender;
5  address receiver;
6  uint256 timestamp;
7 }

```

Os eventos emitidos pelo Smart Contract servem para posterior agrupamento pelo software Front-end e pelos atores interessados em escutar as atualizações de estado.

Código 5.2 – Código solidity 2 - Eventos

```

1 event Mint(string indexed _tokenId, string indexed _
    tokenReference, address indexed _receiver);
2 event Burn(string indexed _tokenId);
3 event EnabledOperator(address indexed _address, string indexed _
    tokenReference);
4 event DisabledOperator(address indexed _address, string indexed _
    tokenReference);

```

Apenas é permitido ao dono do Smart Contract (creator) o gerenciamento de operadores (habilitar e desabilitar). Cada operador possui uma lista de referências (*tokenReference*) que informa quais tokens o mesmo pode adicionar ou remover das carteiras dos alunos. Uma referência é representada pelo código da disciplina - por exemplo, "INE5660" - e deve ser associada pela aplicação cliente.

Código 5.3 – Código Solidity 3 - Funções de operador

```

1 function enableOperator(address _address, string memory _
    tokenReference) external {
2     require(msg.sender == creator, "Only the creator can allow new
        operators");
3     require(!_isOperatorAllowedTo(_address, _tokenReference), "
        Operator are already enabled to manage this identifier");
4     (...)
5     emit EnabledOperator(_address, _tokenReference);
6 }
7
8 function disableOperator(address _address, string memory _
    tokenReference) external {
9     require(msg.sender == creator, "Only the creator can allow new
        operators");
10    require(_isOperatorAllowedTo(_address, _tokenReference), "
        Operator arent enabled to manage this identifier");
11    (...)
12    emit DisabledOperator(_address, _tokenReference);
13 }

```

O conceito de Mint e Burn das NFTs também é aplicado para o Soulbound Token - sendo que um token só pode ser destruído pelo operador ou dono do contrato, e não pelo usuário (estudante).

Código 5.4 – Código solidity 4 - Mint e Burn

```
1 function mint(string memory _tokenId, string memory _
    tokenReference, address _receiver) external {
2   require(!_exists(_tokenId), "There is already a SoulboundToken
    with given tokenId");
3   require(!_isOperatorAllowedTo(msg.sender, _tokenReference) ||
    msg.sender == creator, "You arent enabled to manage this
    tokenReference");
4   soulboundTokens[_tokenId] = SoulboundToken(_tokenId, _
    tokenReference, msg.sender, _receiver, block.timestamp);
5
6   emit Mint(_tokenId, _tokenReference, _receiver);
7 }
8
9 function burn(string memory _tokenId) external {
10  require(_exists(_tokenId), "There isnt a SoulboundToken with
    given tokenId");
11  require(!_isOperatorAllowedTo(msg.sender, soulboundTokens[_
    tokenId].tokenReference) || msg.sender == creator, "You
    arent enabled to manage this tokenReference");
12
13  delete soulboundTokens[_tokenId];
14  emit Burn(_tokenId);
15 }
```

6 IMPLEMENTAÇÃO

6.1 Endereços

Para realizar os testes na rede principal (*mainnet*), foram criados 12 endereços no padrão Ethereum, sendo transferido um total de 10 MATIC (equivalente a \$7.95 no dia 21/11/2022) para ser utilizado para pagar as taxas das transações da rede. Todos os endereços podem ser verificados em <https://polygonscan.com>.

Tabela 1 – Endereços utilizados

Endereço	Persona
0xfc11cf5bcadd8b34ae74ffa9d0a23fed3a380975	Universidade
0xB4ab92133e4b5869453253B690174Cd85075d9bb	Professor 1
0x994fABd6AD250BC84A0081f12336AD396e69185E	Professor 2
0x49c90F5722c68afdd1C5b0aC8e5835bb52F1f294	Professor 3
0x6359eF0343F103E4846142019c4a035999B15D5c	Aluno 1
0x07F288CabdAA5C02C9F7CF90C4a43F59778c399f	Aluno 2
0x45368DF1ff1735d70065D61556424DB41e12d6F4	Aluno 3
0x69E5D090389ab016ba606db83E1d86E4322c4025	Aluno 4
0xceBc565d8c2e27001262f5641FF7118441CA1c75	Aluno 5
0xd2c16504400CdcACAB3e64E94D9A84d04e72B3fc	Aluno 6
0xB07bDA63af0f16683aA4785154d2e6442C90a948	Aluno 7
0xe34166ef0bb12c7F53808482149eEd65249999Ca	Aluno 8

Elaborado pelo autor

6.2 Deploy

O *Smart Contract* foi implementado na *mainnet* da rede Polygon (MATIC), no endereço `0x08b694591e845ea2700eb0f058f65c04d2eefa0f`. A transação pode ser verificada em: <https://polygonscan.com/tx/0x603134a950d5c68cdb9f67811a878108249e5699bea6f3088283604cc23761ef>.

Figura 7 – Transação de implementação do contrato

The screenshot shows the PolygonScan interface for a transaction. At the top, there's a search bar and navigation links. The main content is titled 'Transaction Details' and includes a 'Gaming' button. Below this, there are tabs for 'Overview', 'Logs (1)', and 'Comments'. The 'Overview' tab is active, displaying the following transaction details:

Transaction Hash:	0x603134a950d5c68cdb9f67811a878108249e5699bea6f3088283604cc23761ef
Status:	Success
Block:	35885937 27 Block Confirmations
Timestamp:	1 min ago (Nov-21-2022 01:24:04 PM +UTC)
From:	0xfc11cf5bcadd8b34ae74ffa9d0a23fed3a380975
To:	[Contract 0x08b694591e845ea2700eb0f058f65c04d2eefa0f Created]
Value:	0 MATIC (\$0.00)
Transaction Fee:	0.055790927115727294 MATIC (\$0.05)
Txn Type:	0 (Legacy)
Private Note:	To access the Private Note feature, you must be Logged In

Disponível em <https://polygonscan.com/tx/0x603134a950d5c68cdb9f67811a878108249e5699bea6f3088283604cc23761ef>

6.3 Custos

Todos os custos abaixo consideram o valor da criptomoeda MATIC no dia 21/11/2022. A implementação do *Smart Contract* custou 0,055790927115727294 MATIC (equivalente a \$0.04). O custo para habilitar um novo operador foi em média de 0,00350555 MATIC (\$0.0028). Com 146 transações de criação de um Soulbound Token na rede Polygon (MATIC), o custo médio por transação ficou em 0,02029048014 MATIC (\$0.016).

Figura 8 – Transação de criação de token

The screenshot displays the PolygonScan interface for a transaction. At the top, the PolygonScan logo is on the left, and navigation links for Home, Blockchain, Tokens, Resources, and More are on the right. A search bar is also present. Below the navigation, the page title is 'Transaction Details'. The transaction information is as follows:

Transaction Hash:	0x63176c5ddc32ca6421b6abdf42dde9f212dcd0f16bee3b3ac5098073ce441c42
Status:	Success
Block:	35893012 (7257 Block Confirmations)
Timestamp:	4 hrs 11 mins ago (Nov-21-2022 05:35:25 PM +UTC)
From:	0x994fabd6ad250bc84a0081f12336ad396e69185e
To:	Contract 0x08b694591e845ea2700eb0f058f65c04d2eefa0f
Value:	0 MATIC (\$0.00)
Transaction Fee:	0.0206927 MATIC (\$0.02)
Txn Type:	2 (EIP-1559)

Additional details include a 'Click to see More' link and a note that the Private Note feature requires being logged in.

Disponível em <https://polygonscan.com/tx/0x63176c5ddc32ca6421b6abdf42dde9f212dcd0f16bee3b3ac5098073ce441c42>

6.4 Frontend

Foi desenvolvida a aplicação cliente inteiramente utilizando *Javascript* e não sendo necessário qualquer código *Backend*, uma vez que todos os eventos são lidos pela carteira do usuário no lado do cliente (Metamask, Coinbase Wallet, etc.). O código da aplicação está disponível no apêndice deste trabalho ou via *Github*, pelo endereço <https://github.com/inacior/sbt-ufsc-tcc/tree/master/frontend/ufsc-sbt>. A última versão está implementada no link <https://sbt-ufsc-j4ve3wfji-renaninacio.vercel.app>.

Figura 9 – Tela inicial da aplicação cliente

The screenshot shows the initial screen of the client application. It features the text 'UFSC SBT' at the top, followed by the wallet address: 'Wallet: 0x6359eF0343F103E4846142019c4a035999B15D5c'. Below the wallet address are two buttons: 'My tokens' and 'Verify address'.

Figura 10 – Tela de verificação de tokens da carteira atual

UFSC SBT

Wallet: 0x6359eF0343F103E4846142019c4a035999B15D5c

<- Back

tokenId	tokenReference	blockNumber	creator	TX
408b1479-4668-4795-855b-c1760d1c141c	INE0001	35887058	0x49c90F5722c68afdd1C5b0aC8e5835bb52F1f294	TX
3f9e3404-a5f8-48e2-94b7-101f773a1bed	INE0002	35887058	0x49c90F5722c68afdd1C5b0aC8e5835bb52F1f294	TX
2a10a7ee-866a-4a01-be65-da19aca7fe25	INE0003	35887058	0x49c90F5722c68afdd1C5b0aC8e5835bb52F1f294	TX

Elaborado pelo autor

Figura 11 – Tela de verificação de tokens

UFSC SBT

Wallet: 0x6359eF0343F103E4846142019c4a035999B15D5c

<- Back

0x07F288CabdaA5C02C9F7CF90C4a43F59778c399f
verify

tokenId	tokenReference	blockNumber	creator	TX
61018a4d-db11-4d74-9156-6c253444264f	INE0001	35887058	0x49c90F5722c68afdd1C5b0aC8e5835bb52F1f294	TX
2ff33308-8a53-4499-9371-5a5f9d4b4ed9	INE0002	35887058	0x49c90F5722c68afdd1C5b0aC8e5835bb52F1f294	TX

Elaborado pelo autor

7 CONSIDERAÇÕES FINAIS E TRABALHOS FUTUROS

Com a realização deste trabalho, foi possível aprender na prática como trazer para a Web3 operações do cotidiano universitário que atualmente estão dependentes de uma entidade centralizada e privada. A proposta implementada em linguagem Solidity e Javascript conseguiu atingir os objetivos propostos do projeto mantendo o baixo custo de operação (menor que \$0.02 por transação). Implementada em uma rede pública, auditável e descentralizada (Polygon MATIC), os dados podem ser verificados por agentes internos (revistas, pesquisadores e empregadores) e externos (universidades, professores e alunos), trazendo uma alta confiabilidade nos dados e a possibilidade de análise e uso dos dados para as mais diversas finalidades.

Como melhoria a ser implementada em trabalhos futuros está a implementação de *batch transactions* para diminuir o custo médio da transação - hoje, o *Smart Contract* proposto cria um token por transação iniciada pelo usuário. Além disso, se pode melhorar a governança e gestão de operadores via criação de uma DAO (*Decentralized Autonomous Organization*) para a gestão de acessos, permitindo assim que se recrie toda a cadeia de permissões existentes em um ambiente acadêmico *on-chain*.

8 REFERÊNCIAS

NAKAMOTO, Satoshi. **Bitcoin: A peer-to-peer electronic cash system**. 2008. Disponível em: <https://bitcoin.org/bitcoin.pdf>. Acesso em: 02 out. 2022.

WEYL, E. Glen; OHLHAVER, Puja; BUTERIN, Vitalik. **Decentralized Society: Finding Web3's Soul**. 2022. Disponível em: <http://dx.doi.org/10.2139/ssrn.4105763>. Acesso em: 01 set. 2022.

PATIL, Aditya Sachin. **Decentralized apps of the future implementing Web3**. 2021. Disponível em: <http://dx.doi.org/10.13140/RG.2.2.17477.17129/1>. Acesso em: 01 set. 2022.

ČUČKO, Špela; BEĆIROVIĆ, Šeila; KAMIĀLALIĆ, Aida; MRDOVIĆ, Saša; TURKANOVIĆ, Muhamed. **Towards the classification of Self-Sovereign Identity properties**. IEEE Access 10, 2022. Disponível em: <http://dx.doi.org/10.1093/polsoc/puac018>. Acesso em: 01 set. 2022

DUCRÉE, Jens; GRAVITT, Max; WALSH, Ray; BARTLING, Sönke; ETZRODT, Martin; HARRINGTON, Tomás. **Open Platform Concept for Blockchain-Enabled Crowdsourcing of Technology Development and Supply Chains**. *Frontiers In Blockchain*, 2020. Frontiers Media SA. Disponível em: <http://dx.doi.org/10.3389/fbloc.2020.586525>.

PALMA, Lucas Machado da. **Blockchain-Based Academic Record System**, 2020. Disponível em: <https://repositorio.ufsc.br/handle/123456789/216496>.

Barros, Maurício de Vasconcelos. **Um comprovante de vacinação baseado em Identidade Auto-Soberana, Blockchain e Provas de Zero Conhecimento**, 2021. Disponível em: <https://repositorio.ufsc.br/handle/123456789/228616>.

Correa, Otavio Augusto. **Estudo da aplicação de estrutura blockchain com proof of stake para arquivamento de documentos com registro no tempo**, 2018. Disponível em: <https://repositorio.ufsc.br/handle/123456789/187862>.

Elmissiry, A.; Elmissiry M.; Bridgesmith, Larry. **NFT Student Teacher Incentive System (NFT-STIS)**, 2021. Disponível em: <http://dx.doi.org/10.2139/ssrn.4120879>.

INFOMONEY; **Ethereum: como surgiu a segunda criptomoeda mais valiosa do mundo?**. 2022. Disponível em: <https://www.infomoney.com.br/guias/o-que-e-ethereum/>. Acesso em: 11 nov. 2022.

REY, Jorge Feliu. **Smart contract: conceito, ecossistema e principais questões**

de direito privado. 2019. Disponível em: <https://doi.org/10.18316/redes.v7i3.6120>. Acesso em 22 nov. 2022.

BINANCE. **What Is Layer 1 in Blockchain?**. 2022. Disponível em: <https://academy.binance.com/en/articles/what-is-layer-1-in-blockchain/>. Acesso em: 22 nov. 2022.

BYBIT. **How Soulbound Tokens (SBTs) Can Improve Society With Web 3.0.** 2022. Disponível em: <https://learn.bybit.com/nft/soulbound-tokens-sbts-web3>. Acesso em: 22 nov. 2022.

9 APÊNDICES

9.1 Smart Contract

Código 9.1 – Código do Smart Contract

```

1 // SPDX-License-Identifier: UNLICENSED
2 pragma solidity ^0.8.13;
3
4 contract UFSCSBT {
5     event Mint(string indexed _tokenId, string indexed _
        tokenReference, address indexed _receiver);
6     event Burn(string indexed _tokenId);
7     event EnabledOperator(address indexed _address, string indexed
        _tokenReference);
8     event DisabledOperator(address indexed _address, string
        indexed _tokenReference);
9
10    struct SoulboundToken {
11        string tokenId;
12        string tokenReference;
13        address sender;
14        address receiver;
15        uint256 timestamp;
16    }
17
18    mapping (string => SoulboundToken) soulboundTokens;
19    mapping (address => string[]) enabledOperators;
20
21    address public creator;
22    string public name;
23    string public id;
24
25    constructor(string memory _name, string memory _id) {
26        name = _name;
27        id = _id;
28        creator = msg.sender;
29    }
30
31    function _exists(string memory tokenId) internal view virtual
        returns (bool) {
32        return keccak256(abi.encodePacked(soulboundTokens[tokenId].
            tokenId)) != keccak256(abi.encodePacked(""));
33    }
34
35    function _isOperator(address _address) internal view virtual
        returns (bool) {
36        if(enabledOperators[_address].length > 0){
37            return true;
38        }
39
40        return false;
41    }
42
43    function _isOperatorAllowedTo(address _address, string memory
        _tokenReference) internal view virtual returns (bool) {

```

```

44     if (!_isOperator(_address)) return false;
45
46     for (uint i = 0; i < enabledOperators[_address].length; i
47         ++) {
48         if (keccak256(abi.encodePacked(enabledOperators[_address
49             ][i])) == keccak256(abi.encodePacked(_tokenReference)
50             )) {
51             return true;
52         }
53     }
54     return false;
55 }
56
57 function getOperator(address _address) public view returns (
58     string[] memory) {
59     return enabledOperators[_address];
60 }
61
62 function ownerOf(string memory tokenId) public view returns (
63     address) {
64     return soulboundTokens[tokenId].receiver;
65 }
66
67 function mint(string memory _tokenId, string memory _
68     tokenReference, address _receiver) external {
69     require(!_exists(_tokenId), "There is already a
70         SoulboundToken with given tokenId");
71     require(!_isOperatorAllowedTo(msg.sender, _tokenReference)
72         || msg.sender == creator, "You arent enabled to manage
73         this tokenReference");
74     soulboundTokens[_tokenId] = SoulboundToken(_tokenId, _
75         tokenReference, msg.sender, _receiver, block.timestamp);
76     emit Mint(_tokenId, _tokenReference, _receiver);
77 }
78
79 function burn(string memory _tokenId) external {
80     require(!_exists(_tokenId), "There isnt a SoulboundToken
81         with given tokenId");
82     require(!_isOperatorAllowedTo(msg.sender, soulboundTokens[_
83         tokenId].tokenReference) || msg.sender == creator, "You
84         arent enabled to manage this tokenReference");
85
86     delete soulboundTokens[_tokenId];
87     emit Burn(_tokenId);
88 }
89
90 function enableOperator(address _address, string memory _
91     tokenReference) external {
92     require(msg.sender == creator, "Only the creator can allow
93         new operators");
94     require(!_isOperatorAllowedTo(_address, _tokenReference), "
95         Operator are already enabled to manage this identifier")
96     ;
97
98     enabledOperators[_address].push(_tokenReference);

```

```

84     emit EnabledOperator(_address, _tokenReference);
85 }
86
87 function disableOperator(address _address, string memory _
88     tokenReference) external {
89     require(msg.sender == creator, "Only the creator can allow
90         new operators");
91     require(!_isOperatorAllowedTo(_address, _tokenReference), "
92         Operator arent enabled to manage this identifier");
93
94     string[] memory newTokenReferenceList = new string[](
95         enabledOperators[_address].length - 1);
96     uint256 resultCount = 0;
97
98     for (uint i = 0; i < enabledOperators[_address].length; i
99         ++) {
100         if (keccak256(abi.encodePacked(enabledOperators[_address
101             ][i])) != keccak256(abi.encodePacked(_tokenReference)
102             )) {
103             newTokenReferenceList[resultCount] = enabledOperators
104                 [_address][i];
105             resultCount++;
106         }
107     }
108
109     enabledOperators[_address] = newTokenReferenceList;
110
111     emit DisabledOperator(_address, _tokenReference);
112 }
113 }
114 }
115 }

```

9.2 Frontend

Código 9.2 – Código da aplicação cliente

```

1 import { ethers, Signer, Contract, Event } from "ethers";
2
3 import Head from 'next/head'
4 import { useEffect, useState } from 'react';
5 import styles from '../styles/Home.module.css'
6
7 import contractABI from '../SBTContract.json'
8
9 type SBTToken = {
10     transactionId: string
11     tokenId: string
12     tokenReference: string
13     blockNumber?: number
14     creator: string
15 }
16
17 const INITIAL_BLOCK = 35885937
18
19 export default function Home() {
20     const [screen, setScreen] = useState<'INITIAL' | 'TOKENS_LIST'
21         | 'VERIFY_ADDRESS'>('INITIAL')

```

```

21  const [contract, setContract] = useState()
22  const [provider, setProvider] = useState()
23  const [address, setAddress] = useState()
24  const [signer, setSigner] = useState()
25  const [tokens, setTokens] = useState([])
26  const [isLoading, setIsLoading] = useState(true)
27  const [addressToSearch, setAddressToSearch] = useState('')
28
29  const iface = new ethers.utils.Interface(contractABI.abi)
30
31  useEffect(() => {
32    handleConnect()
33    // eslint-disable-next-line react-hooks/exhaustive-deps
34  }, [])
35
36  const handleConnect = async () => {
37    // @ts-ignore
38    const provider = new ethers.providers.Web3Provider(window.
      ethereum, 137)
39    await provider.send("eth_requestAccounts", []);
40    const _signer = provider.getSigner()
41    const _contract = new ethers.Contract('0
      x08b694591e845ea2700eb0f058f65c04d2eeefa0f', contractABI.
      abi, _signer)
42
43    setSigner(_signer)
44    setContract(_contract)
45    setProvider(provider)
46    setAddress(await _signer.getAddress())
47
48    setIsLoading(false)
49  }
50
51  const handleTokenList = async (addressToSearch: string) => {
52    if (!contract || (!provider)) return;
53    setIsLoading(true)
54    setTokens([])
55    const chain = []
56
57    const latestBlock = await provider.getBlock("latest")
58
59    for (let index = 0; index < Math.floor((latestBlock.number -
      INITIAL_BLOCK)/1000) + 1; index++) {
60      const blockFrom = INITIAL_BLOCK + (1000 * index);
61      const blockTo = (blockFrom + 1000) > latestBlock.number ?
        latestBlock.number : blockFrom + 1000;
62      const _events = await contract.queryFilter(contract.filters.
        Mint(null, null, addressToSearch), blockFrom, blockTo)
63
64      chain.push(_events)
65    }
66
67    await Promise.all(chain)
68      .then((events) => {
69        events.flatMap((event) => event)
70          .forEach((event) => appendMintEvent(event))
71      })

```

```

72     .catch(e => {
73         console.error(e)
74     })
75
76     setIsLoading(false)
77 }
78
79 const handleBack = () => {
80     setScreen('INITIAL')
81     setTokens([])
82 }
83
84 const appendMintEvent = async (event: Event) => {
85     const transaction = await event.getTransaction()
86     let decodedData = iface.parseTransaction({ data: transaction.
      data, value: transaction.value });
87
88     const _token : SBTToken = {
89         transactionId: transaction.hash,
90         tokenId: decodedData.args[0],
91         tokenReference: decodedData.args[1],
92         blockNumber: transaction.blockNumber,
93         creator: transaction.from
94     }
95
96     setTokens((prevValue) => ([ ...prevValue, _token ]))
97 }
98
99 const MyTokens = ({ tokens } : { tokens: SBTToken[] }) => {
100     return (
101         <>
102             {'<- Back'}
103             {isLoading && (Loading...)}
104
105     )
106 }
107
108
109 const TokensList = ({ tokens } : { tokens: SBTToken[] }) => {
110     return (
111         {tokens.map((t) => (
112
113
114         ))}
115
116         tokenId
117         tokenReference
118         blockNumber
119         creator
120         TX
121
122         {t.tokenId}
123         {t.tokenReference}
124         {t.blockNumber}
125         {t.creator}
126         TX
127

```

```

128
129   )
130 }
131
132 const VerifyAddress = () => {
133   return (
134     <>
135     {'<- Back'}
136
137     setAddressToSearch(e.target.value)} />
138     handleTokenList(addressToSearch)}>verify
139
140     {isLoading && (Loading...)}
141
142   )
143 }
144
145 return (
146
147   UFSC SBT
148
149   UFSC SBT
150
151   UFSC SBT
152
153   UFSC SBT
154
155   {
156     signer
157     ? (
158       Wallet: {address}
159
160       {(() => {
161         switch (screen) {
162           case 'INITIAL':
163             return (
164               {
165                 setScreen('TOKENS_LIST');
166                 handleTokenList(address || '')}
167               }>My tokens
168               setScreen('VERIFY_ADDRESS')}>Verify
169               address
170             )
171           case 'TOKENS_LIST':
172             return ()
173           case 'VERIFY_ADDRESS':
174             return ()
175           default:
176             break;
177         }
178       })()
179     )
180
181   : (
182     {isLoading ? "Loading..." : "error" }

```

184)
185 }
186
187
188
189)
190 }

9.3 Artigo

Aprovações acadêmicas públicas e auditáveis com o uso de blockchain implementadas no padrão de soulbound token

Renan Inacio¹, Alex Sandro Roschildt Pinto¹

¹ Departamento de Informática e Estatística – Universidade Federal de Santa Catarina
(UFSC)
Florianópolis, SC – Brazil

inacior.dev@gmail.com, a.r.pinto@ufsc.br

Abstract. *The Decentralized ledger technology (blockchain) was created by Nakamoto in 2009, and provides a high level of trust and transparency in financial transactions with Bitcoin. With the advent of the Ethereum network, this level of trust was extended to immutable codes of decentralized applications, whose transactions are validated by thousands of decentralized points (nodes) around the world. Instead of centralized organizations of power and the need of trust in them, we only need to trust in the code. This proposal allowed the creation of interchangeable tokens: two tokens have the same properties and are interchangeable without loss of quality or value, which we call fungible. On the other hand, we need to represent non-fungible items, like art, ownership certificates or a university degree. For these reasons, the concept of non-fungible tokens - (NFTs) was created. Using the concept of non-fungible and non-transferable tokens - called Soulbound Tokens or SBTs - a decentralized application (dApp) was implemented to generate public and auditable tokens in the context of academic approvals. The central idea was to promote access to 4 personas and their use cases: university, professors, students and verifiers. The implementation of the Smart Contract was carried out on Polygon's main network (MATIC) with a cost per transaction of less than \$0.02, along with a client application to interact with it.*

Resumo. *A tecnologia de registro descentralizada (blockchain) foi criada por Nakamoto em 2009, e proporciona um alto nível de confiança e transparência em transações financeiras por meio do Bitcoin. Com o advento da rede Ethereum, este nível de confiança se estendeu para códigos imutáveis de aplicações descentralizadas, cujas transações são validadas por milhares de pontos (nodes) descentralizados pelo mundo. No lugar de organizações centralizadas de poder e necessidade de confiança nas mesmas, precisamos apenas confiar no código. Esta proposta permitiu a criação de tokens intercambiáveis entre si: dois tokens possuem as mesmas propriedades e substituíveis sem prejuízo da qualidade ou valor, o que chamamos de fungível. Por outro lado, precisamos representar itens não-fungíveis, como obras de arte, certificados de propriedade ou um diploma universitário. Por estes motivos, foi criado o conceito de tokens não fungíveis - em inglês, NFTs. Utilizando o conceito de tokens não fungíveis e não transferíveis - chamados de Soulbound Tokens ou SBTs - foi implementada uma aplicação descentralizada (dApp) para geração de tokens públicos e auditáveis no contexto de aprovações acadêmicas. A ideia central foi promover o acesso á*

4 personas e seus casos de uso: universidade, professores, alunos e verificadores. A implementação do Smart Contract foi feita na rede principal da Polygon (MATIC) tendo um custo por transação inferior a \$0.02, junto de uma aplicação cliente para interagir com o mesmo.

1. Introdução

A internet como conhecemos hoje pode ser dividida em 3 categorias: Web1, Web2 e Web3, cada qual com sua particularidade e foco. Chamamos de Web1 nosso primeiro contato com a internet, onde o conteúdo era estático e sem muita interação e gerado previamente por uma organização centralizada. Com o avanço da tecnologia e das interações entre consumidor e empresa, surgiu o conceito que hoje chamamos de Web2, trazendo o usuário para dentro do processo criativo e o tornando criador de seu próprio conteúdo. No conceito de Web3, no lugar de organizações centralizadas de poder e necessidade de confiança nas mesmas, precisamos apenas confiar no código, cujas transações são validadas por milhares de pontos (nodes) descentralizados pelo mundo [Patil, 2021].

A tecnologia de registro descentralizada (em inglês, Distributed Ledger Technology - DLT), ou simplesmente Blockchain - criada por Nakamoto em 2009 - proporciona um alto nível de confiança e transparência em transações financeiras (Bitcoin) no mundo digital. Com o advento da rede Ethereum, foi possível criar códigos imutáveis e validados por uma rede descentralizada, paralela e isolada, sem a necessidade de um mecanismo central, o que chamamos de Smart Contract [Ducrée; Gravitt; Walshe; Bartling; Etzrodt; Harrington, 2020].

Mesmo com o advento da Web3 em transações financeiras pelo mundo, a falta de mecanismos de verificação de identidade e reputação força os usuários a depender de estruturas centralizadas dentro da Web2. Web3 hoje se concentra em expressar ativos financeiros e transferências descentralizadas, quando deveríamos incluir também a inclusão de redes de confiança em uma sociedade [Weyl; Ohlhaber; Buterin, 2022]. O conceito de “Decentralized Society” (DeSoc) originado por Buterin (2022) foca na criação de tokens públicos e não-transferíveis como mecanismo de identidade social descentralizada e auditável, referenciados como “Soulbound Tokens” (SBTs).

2. Proposta

Neste artigo, propomos a criação de um Smart Contract transparente e seguro para gerenciamento de aprovações de disciplinas do curso de Sistemas de Informação da Universidade Federal de Santa Catarina (UFSC). Primeiramente apresentamos os participantes envolvidos e como os mesmos interagem com o projeto, apresentamos também o esboço da aplicação cliente (front-end) e o smart contract proposto para atingir os objetivos.

2.1. Participantes

A proposta inclui 4 participantes. A Figura 3 exhibe as principais interações de cada participante na rede. O primeiro participante é a Universidade, ela é responsável por

habilitar novos participantes Professores para interagirem na rede, bem como adicionar novas matérias a serem validadas e adicionadas futuramente á carteira do Estudante.

O participante Estudante consegue consultar as matérias já validadas para sua matrícula e suas informações. O participante Estudante também pode assinar transações para verificar sua identidade ao participante Validador. Este terceiro participante chamado de Validador representa qualquer usuário cujo fim é validar um cadastro de um estudante (um empregador, imprensa, outras universidades, etc.).

Por último temos o participante Professor que, após devidamente cadastrado pela Universidade, pode validar as matérias habilitadas para o mesmo nos cadastros (carteiras) dos Estudantes. Todas as interações são salvas na rede Blockchain permitindo futura auditoria dos dados.

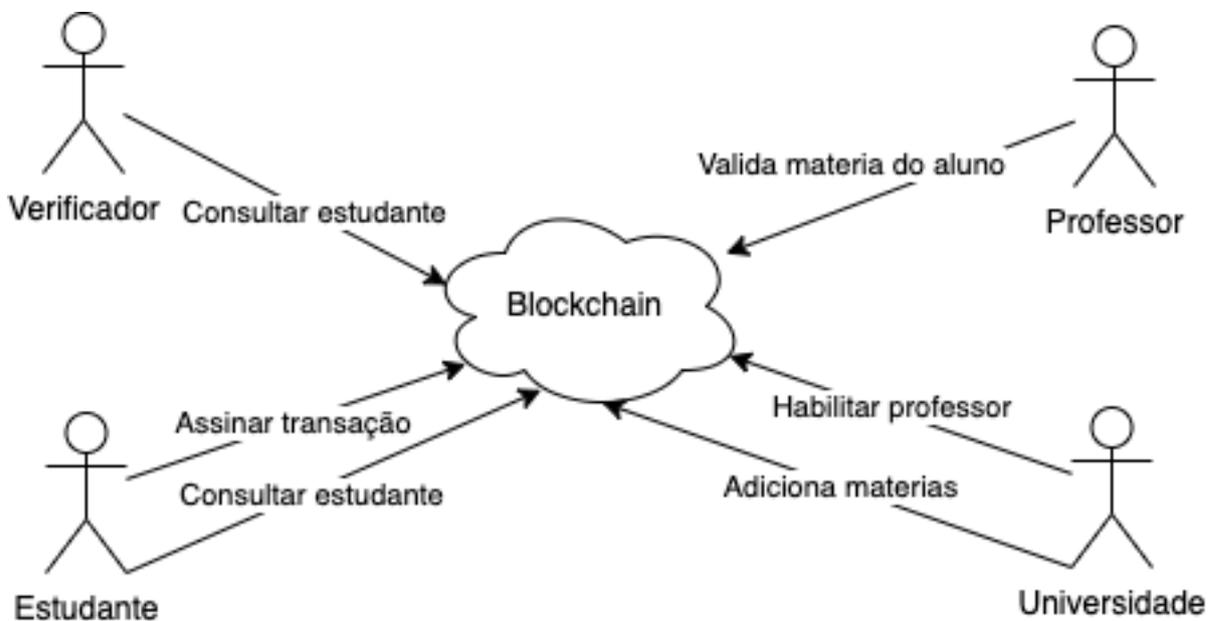


Figura 1. Participantes e suas interações

2.2. Front-end

Como parte do projeto, será elaborada uma página Front-end para cobrir os casos de usos propostos na seção anterior, preferencialmente disponibilizada via IPFS. Como se trata de uma aplicação descentralizada (DApp), os dados serão acessíveis após o usuário conectar sua carteira de criptomoedas utilizando seu aplicativo preferido (MetaMask, Coinbase Wallet, Rainbow Wallet, etc.).

Ao entrar na aplicação, o usuário poderá verificar os tokens associados a sua própria conta (no caso de ser um estudante) ou verificar os tokens cadastrados para a conta de outra pessoa (no caso de ser um verificador).



Figura 2. Proposta de tela inicial

Ao clicar em uma Verificar meus tokens, será possível visualizar os Soulbound Tokens (matérias validadas) que o usuário possui, a data de verificação, o usuário que efetuou a criação do token (Professor ou Coordenadoria) e uma seção para visualizar os dados da transação e conferir a autenticidade do mesmo.

Tokens verificados				
Token ID	Referencia	Verificado em	Verificado por	Detalhes
6ff2c8v-72d	IME0001	20/07/2022 14:34:01	Professor 1	👁️
6f236g-9s7f	IME0002	22/07/2022 16:47:12	Professor 2	👁️

Figura 3. Proposta de tela de verificação de Soulbound Tokens (matérias validadas) do usuário

2.3. Smart Contract

Para atender a proposta do trabalho, foi codificado o Smart Contract abaixo. Foi definida uma estrutura para o token (SoulboundToken) contendo:

- tokenId: ID único para aquele token;

- tokenReference: Uma referência externa para o token, neste caso sendo o código da disciplina;
- sender: Endereço da carteira do professor que criou o token;
- receiver: Endereço da carteira do aluno que recebeu o token;
- timestamp: Data do bloco que contém a transação;

Código 1. Código solidity 1 - Struct

```

1 struct SoulboundToken {
2   string tokenId;
3   string tokenReference;
4   address sender;
5   address receiver;
6   uint256 timestamp;
7 }

```

Os eventos emitidos pelo Smart Contract servem para posterior agrupamento pelo software Front-end e pelos atores interessados em escutar as atualizações de estado.

Código 2. Código solidity 2 - Eventos

```

1 event Mint(string indexed _tokenId, string indexed _
   tokenReference, address indexed _receiver);
2 event Burn(string indexed _tokenId);
3 event EnabledOperator(address indexed _address, string indexed
   _tokenReference);
4 event DisabledOperator(address indexed _address, string
   indexed _tokenReference);

```

Apenas é permitido ao dono do Smart Contract (creator) o gerenciamento de operadores (habilitar e desabilitar). Cada operador possui uma lista de referências (token-Reference) que informa quais tokens o mesmo pode adicionar ou remover das carteiras dos alunos. Uma referência é representada pelo código da disciplina - por exemplo, "INE5660" - e deve ser associada pela aplicação cliente.

Código 3. Código Solidity 3 - Funções de operador

```

1 function enableOperator(address _address, string memory _
   tokenReference) external {
2   require(msg.sender == creator, "Only the creator can allow
   new operators");
3   require(!_isOperatorAllowedTo(_address, _tokenReference), "
   Operator are already enabled to manage this identifier");
4   (...)
5   emit EnabledOperator(_address, _tokenReference);
6 }
7
8 function disableOperator(address _address, string memory _
   tokenReference) external {
9   require(msg.sender == creator, "Only the creator can allow
   new operators");
10  require(_isOperatorAllowedTo(_address, _tokenReference), "
   Operator arent enabled to manage this identifier");
11  (...)
12  emit DisabledOperator(_address, _tokenReference);
13 }

```

O conceito de Mint e Burn das NFTs também é aplicado para o Soulbound Token - sendo que um token só pode ser destruído pelo operador ou dono do contrato, e não pelo usuário (estudante).

Código 4. Código solidity 4 - Mint e Burn

```

1 function mint(string memory _tokenId, string memory _
  tokenReference, address _receiver) external {
2   require(!_exists(_tokenId), "There is already a
  SoulboundToken with given tokenId");
3   require(_isOperatorAllowedTo(msg.sender, _tokenReference) ||
  msg.sender == creator, "You arent enabled to manage this
  tokenReference");
4   soulboundTokens[_tokenId] = SoulboundToken(_tokenId, _
  tokenReference, msg.sender, _receiver, block.timestamp);
5
6   emit Mint(_tokenId, _tokenReference, _receiver);
7 }
8
9 function burn(string memory _tokenId) external {
10  require(_exists(_tokenId), "There isnt a SoulboundToken with
  given tokenId");
11  require(_isOperatorAllowedTo(msg.sender, soulboundTokens[_
  tokenId].tokenReference) || msg.sender == creator, "You
  arent enabled to manage this tokenReference");
12
13  delete soulboundTokens[_tokenId];
14  emit Burn(_tokenId);
15 }

```

3. Implementação

3.1. Endereços

Para realizar os testes na rede principal (mainnet), foram criados 12 endereços no padrão Ethereum, sendo transferido um total de 10 MATIC (equivalente a \$7.95 no dia 21/11/2022) para ser utilizado para pagar as taxas das transações da rede. Todos os endereços podem ser verificados em <https://polygonscan.com>.

Tabela 1. Endereços utilizados

Endereço	Persona
0xfc11cf5bcadd8b34ae74ffa9d0a23fed3a380975	Universidade
0xB4ab92133e4b5869453253B690174Cd85075d9bb	Professor 1

Endereço	Persona
0x994fABd6AD250BC84A0081f12336AD396e69185E	Professor 2
0x49c90F5722c68afdd1C5b0aC8e5835bb52F1f294	Professor 3
0x6359eF0343F103E4846142019c4a035999B15D5c	Aluno 1
0x07F288CabdAA5C02C9F7CF90C4a43F59778c399f	Aluno 2
0x45368DF1ff1735d70065D61556424DB41e12d6F4	Aluno 3
0x69E5D090389ab016ba606db83E1d86E4322c4025	Aluno 4
0xceBc565d8c2e27001262f5641FF7118441CA1c75	Aluno 5
0xd2c16504400CdcACAB3e64E94D9A84d04e72B3fc	Aluno 6
0xB07bDA63af0f16683aA4785154d2e6442C90a948	Aluno 7
0xe34166ef0bb12c7F53808482149eEd65249999Ca	Aluno 8

3.2. Deploy

O Smart Contract foi implementado na mainnet da rede Polygon (MATIC), no endereço 0x08b694591e845ea2700eb0f058f65c04d2eefa0f. A transação pode ser verificada em: <https://polygonscan.com/tx/0x603134a950d5c68cdb9f67811a878108249e5699bea6f3088283604cc23761ef>.

The screenshot displays the 'Transaction Details' page on Polygonscan. At the top, there's a search bar and navigation links. The main content area shows the transaction details for a successful contract deployment. The transaction hash is 0x63176c5ddc32ca6421b6abd42dde9f212dcd0f16bee3b3ac5098073ce441c42. The status is 'Success'. The transaction occurred in block 35893012, which has 7257 block confirmations. The timestamp is 4 hours and 11 minutes ago from November 21, 2022, at 05:35:25 PM UTC. The transaction was sent from the address 0x994fabd6ad250bc84a0081f12336ad396e69185e to the contract address 0x08b694591e845ea2700eb0f058f65c04d2eefa0f. The value of the transaction is 0 MATIC (\$0.00), and the transaction fee is 0.0206927 MATIC (\$0.02). The transaction type is 2 (EIP-1559). There is a link to see more details and a note about the Private Note feature requiring a login.

Figura 4. Transação de implementação do contrato

3.3. Custos

Todos os custos abaixo consideram o valor da criptomoeda MATIC no dia 21/11/2022. A implementação do Smart Contract custou 0,055790927115727294 MATIC (equivalente a \$0.04). O custo para habilitar um novo operador foi em média de

0,00350555 MATIC (\$0.0028). Com 146 transações de criação de um Soulbound Token na rede Polygon (MATIC), o custo médio por transação ficou em 0,02029048014 MATIC (\$0.016).

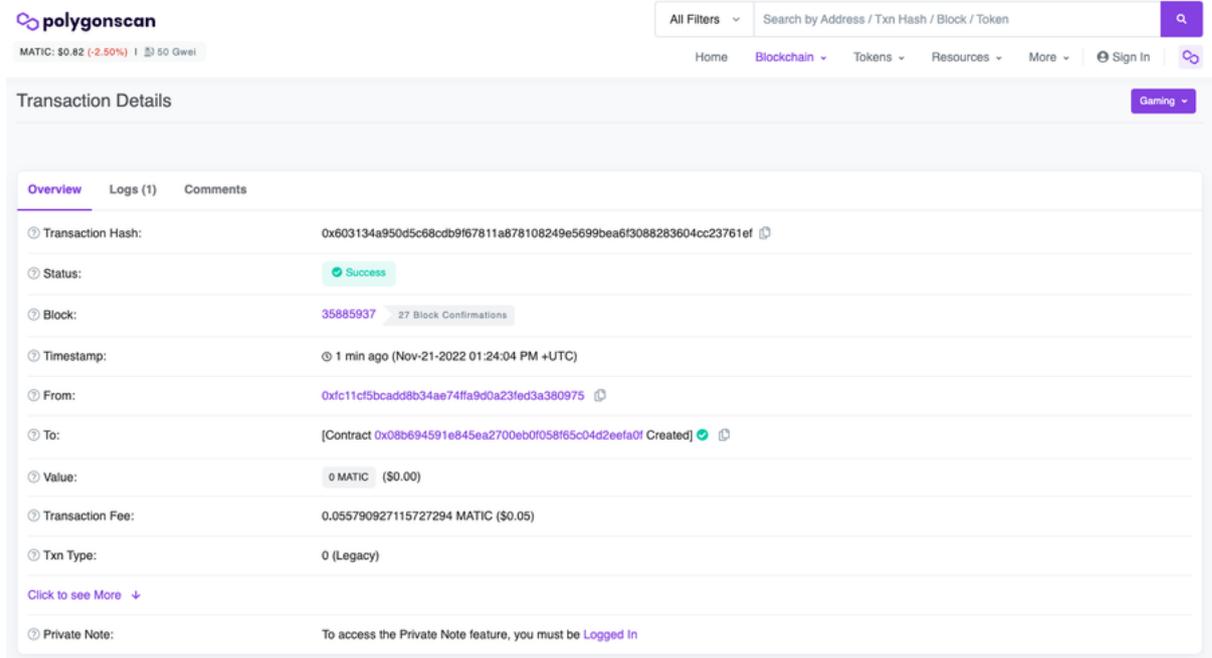


Figura 5. Transação de criação de token

3.4. Frontend

Foi desenvolvida a aplicação cliente inteiramente utilizando Javascript e não sendo necessário qualquer código Backend, uma vez que todos os eventos são lidos pela carteira do usuário no lado do cliente (Metamask, Coinbase Wallet, etc.). O código da aplicação está via Github pelo endereço <https://github.com/inacior/sbt-ufsc-tcc/tree/master/frontend/ufsc-sbt>. A última versão está implementada no link <https://sbt-ufsc-j4ve3wfji-renaninacio.vercel.app>.



Figura 6. Tela inicial da aplicação cliente

UFSC SBT

Wallet: 0x6359eF0343F103E4846142019c4a035999B15D5c

<- Back

tokenId	tokenReference	blockNumber	creator	TX
408b1479-4668-4795-855b-c1760d1c141c	INE0001	35887058	0x49c90F5722c68afdd1C5b0aC8e5835bb52F1f294	TX
3f9e3404-a5f8-48e2-94b7-101f773a1bed	INE0002	35887058	0x49c90F5722c68afdd1C5b0aC8e5835bb52F1f294	TX
2a10a7ee-866a-4a01-be65-da19aca7fe25	INE0003	35887058	0x49c90F5722c68afdd1C5b0aC8e5835bb52F1f294	TX

Figura 7. Tela de verificação de tokens da carteira atual

UFSC SBT

Wallet: 0x6359eF0343F103E4846142019c4a035999B15D5c

<- Back

0x07F288CabdaA5C02C9F7CF90C4a43F59778c399f verify

tokenId	tokenReference	blockNumber	creator	TX
61018a4d-db11-4d74-9156-6c253444264f	INE0001	35887058	0x49c90F5722c68afdd1C5b0aC8e5835bb52F1f294	TX
2ff33308-8a53-4499-9371-5a5f9d4b4ed9	INE0002	35887058	0x49c90F5722c68afdd1C5b0aC8e5835bb52F1f294	TX

Figura 8. Tela de verificação de tokens

4. Conclusão

Com a realização deste trabalho, foi possível aprender na prática como trazer para a Web3 operações do cotidiano universitário que atualmente estão dependentes de uma entidade centralizada e privada. A proposta implementada em linguagem Solidity e Javascript conseguiu atingir os objetivos propostos do projeto mantendo o baixo custo de operação (menor que \$0.02 por transação). Implementada em uma rede pública, auditável e descentralizada (Polygon MATIC), os dados podem ser verificados por agentes internos (revistas, pesquisadores e empregadores) e externos (universidades, professores e alunos), trazendo uma alta confiabilidade nos dados e a possibilidade de análise e uso dos dados para as mais diversas finalidades.

Como melhoria a ser implementada em trabalhos futuros está a implementação de batch transactions para diminuir o custo médio da transação - hoje, o Smart Contract proposto cria um token por transação iniciada pelo usuário. Além disso, se pode melhorar a governança e gestão de operadores via criação de uma DAO (Decentralized Autonomous Organization) para a gestão de acessos, permitindo assim que se recrie toda a cadeia de permissões existentes em um ambiente acadêmico on-chain.

5. Referências

WEYL, E. Glen; OHLHAVER, Puja; BUTERIN, Vitalik. **Decentralized Society: Finding Web3's Soul**. 2022. Disponível em: <http://dx.doi.org/10.2139/ssrn.4105763>. Acesso em: 01 set. 2022.

PATIL, Aditya Sachin. **Decentralized apps of the future implementing Web3**. 2021. Disponível em: <http://dx.doi.org/10.13140/RG.2.2.17477.17129/1>. Acesso em: 01 set. 2022.

ČUČKO, Špela; BEĆIROVIĆ, Šeila; KAMIŁALIĆ, Aida; MRDOVIĆ, Saša; TURKANOVIĆ, Muhamed. **Towards the classification of Self-Sovereign Identity properties.** Ieee Access 10, 2022. Disponível em: <http://dx.doi.org/10.1093/polsoc/puac018>. Acesso em: 01 set. 2022

DUCRÉE, Jens; GRAVITT, Max; WALSH, Ray; BARTLING, Sönke; ETZRODT, Martin; HARRINGTON, Tomás. **Open Platform Concept for Blockchain-Enabled Crowdsourcing of Technology Development and Supply Chains.** Frontiers In Blockchain, 2020. Frontiers Media SA. Disponível em: <http://dx.doi.org/10.3389/fbloc.2020.586525>.

