PONTIFÍCIA UNIVERSIDADE CATÓLICA DE GOIÁS
ESCOLA POLITÉCNICA E DE ARTES
GRADUAÇÃO EM ENGENHARIA DE COMPUTAÇÃO

APLICAÇÃO DA TECNOLOGIA *BLOCKCHAIN* NA AUTENTICIDADE DE DOCUMENTOS

GUSTAVO TOLEDO DE SOUZA

GOIÂNIA
2023

GUSTAVO TOLEDO DE SOUZA

UM EXEMPLO DE APLICAÇÃO DO USO DA *BLOCKCHAIN* PARA AUTENTICIDADE DE DOCUMENTOS

Trabalho de Conclusão de Curso apresentado à Escola Politécnica, da Pontifícia Universidade Católica de Goiás, como parte dos requisitos para a obtenção do título de Bacharel em Engenharia de Computação.

Orientadora:

Profa. Ma. Adriana Silveira de Souza

GOIÂNIA

2023

GUSTAVO TOLEDO DE SOUZA

UM EXEMPLO DE APLICAÇÃO DO USO DA *BLOCKCHAIN* PARA AUTENTICIDADE DE DOCUMENTOS

Trabalho de Conclusão de Curso aprovado em sua forma final pela Escola Politécnica, da Pontifícia Universidade Católica de Goiás, para obtenção do título de Bacharel em Engenharia de Computação, em _____ / _____ / _____.

_____

Orientadora: Profa. Ma. Adriana Silveira de Souza

_____

Prof. 1

_____

Prof. 2

GOIÂNIA

2023

DEDICATÓRIA

Dedico a Deus por me dar força e ajudar a alcançar meus objetivos. A meus pais por serem meus maiores incentivadores e que nunca duvidaram de mim. A minha família como todo por sempre me apoiar.

AGRADECIMENTOS

A professora Ma. Adriana Silveira de Souza, orientadora acadêmica, que teve papel fundamental na elaboração deste trabalho, com seu apoio e dedicação.

Agradeço ao Jonathas Carrijo e ao Israel Buzaym por me darem a oportunidade de ingressar no mercado que por consequência me possibilitou realizar este trabalho.

Agradeço os meus pais Enery e Eliezer por me proporcionarem uma base familiar estável para que eu pudesse me preocupar somente com os estudos.

A minha irmã Gabriela por cuidar de mim como uma irmã mais velha.

RESUMO

A autenticidade de documentos é crucial para o funcionamento eficaz e confiável de instituições em diversos setores, assegurando a integridade dos registros, validade jurídica e autoria, enquanto previne adulterações e falsificações. Esse atributo não apenas contribui para evitar fraudes, proteger direitos individuais e coletivos, mas também facilita a resolução de disputas judiciais. Ao garantir a integridade das transações e registros, a autenticidade fortalece a confiança nas instituições, promovendo relações comerciais e jurídicas mais sólidas e transparentes. No entanto, com o avanço da tecnologia e o surgimento de novas formas de comunicação e armazenamento de dados, também surgiram inúmeros desafios relacionados à segurança e à garantia da autenticidade de documentos. A crescente demanda por soluções seguras e confiáveis no combate às fraudes documentais destaca a importância de abordagens robustas e eficazes. Nesse contexto, a tecnologia *blockchain* se destaca como uma alternativa promissora, permitindo o registro de informações de maneira imutável e permanente. A *blockchain* é um registro distribuído, transparente e imutável, que oferece benefícios significativos no que diz respeito à segurança e confiabilidade das informações registradas. Ao utilizar a *blockchain* para o registro de documentos, é possível garantir a integridade e autenticidade das informações, proporcionando uma camada adicional de segurança e confiança. Após a *blockchain* foi introduzido uma tecnologia capaz de executar programas de forma descentralizada, os contratos inteligentes. Os contratos inteligentes permitem que desenvolvedores criem códigos auditáveis e imutáveis que são executados na *blockchain*. O objetivo deste trabalho é apresentar uma solução baseada em *blockchain* para o registro e garantia de autenticidade de documentos. Para alcançar esse objetivo, foi desenvolvido um contrato inteligente escrito em *Solidity*, uma linguagem de programação específica para a plataforma *blockchain* Ethereum. Esse contrato inteligente possibilita o registro seguro de documentos na *blockchain*, estabelecendo um histórico confiável e imutável de todas as transações. Além disso, para viabilizar a interação com o contrato inteligente e oferecer uma interface amigável aos usuários, uma aplicação web foi desenvolvida. Essa aplicação web permite a comunicação direta com o contrato inteligente, simplificando o processo de registro de documentos e garantindo a transparência das informações.

Palavras-Chave: *Blockchain*, Contratos Inteligentes, Autenticidade de documentos.

ABSTRACT

The authenticity of documents is crucial for the effective and reliable functioning of institutions in various sectors, ensuring the integrity of records, legal validity, and authorship while preventing alterations and forgeries. This attribute not only contributes to preventing fraud, protecting individual and collective rights but also facilitates the resolution of legal disputes. By ensuring the integrity of transactions and records, authenticity strengthens trust in institutions, promoting stronger and more transparent commercial and legal relationships. However, with the advancement of technology and the emergence of new forms of communication and data storage, numerous challenges related to security and document authenticity have also arisen. The growing demand for secure and reliable solutions in combating document fraud underscores the importance of robust and effective approaches. In this context, blockchain technology stands out as a promising alternative, allowing the registration of information in an immutable and permanent manner. Blockchain is a distributed, transparent, and immutable ledger that offers significant benefits in terms of the security and reliability of recorded information. By utilizing blockchain for document registration, it is possible to ensure the integrity and authenticity of information, providing an additional layer of security and trust. After the introduction of blockchain, a technology capable of executing programs in a decentralized manner, smart contracts emerged. These smart contracts enable developers to create auditable and immutable code that runs on the blockchain. The aim of this work is to present a blockchain-based solution for the registration and assurance of document authenticity. To achieve this goal, a smart contract was developed using Solidity, a programming language specific to the Ethereum blockchain platform. This smart contract enables the secure registration of documents on the blockchain, establishing a reliable and immutable history of all transactions. Furthermore, to facilitate interaction with the smart contract and provide a user-friendly interface, a web application was developed. This web application allows direct communication with the smart contract, simplifying the process of document registration and ensuring transparency of information.

Keywords: Blockchain, Smart Contracts, Document Authenticity.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

PDF	*Portable Document Format* (Format de Documento Portátil)
P2P	*Peer-to-Peer* (Ponto a Ponto)
PoW	*Proof-of-Work* (Prova de Trabalho)
PoS	*Proof-of-Stake* (Prova de Participação)
DPoS	*Delegated-Proof-of-Stake* (Prova de Participação Delegada)
TPS	Transações por segundo
DAPP	Aplicativos Descentralizados
API	*Application Programming Interface* (Interface de Programação de Aplicativos)
EVM	*Ethereum Virtual Machine* (Máquina Virtual Ethereum)
TxID	*Transaction Identifier* (Identificador da Transação)

SUMÁRIO

1. INTRODUÇÃO

A autenticidade de documentos e assinaturas desempenha um papel fundamental no funcionamento eficiente e confiável de diversas instituições em diferentes setores. A sua importância reside na garantia da integridade dos registros, na validade jurídica e na prova incontestável da autoria dos documentos, prevenindo potenciais adulterações e falsificações. Além disso, a autenticidade contribui para a prevenção de fraudes, a proteção de direitos individuais e coletivos, e facilita a resolução de disputas judiciais, promovendo a confiança nas transações e relações comerciais.

Contudo, diante do avanço tecnológico e das novas formas de comunicação e armazenamento de dados, surgem desafios significativos relacionados à segurança e à garantia da autenticidade documental. A crescente demanda por soluções seguras no combate às fraudes destaca a necessidade de abordagens robustas e eficazes. Nesse contexto, a tecnologia *blockchain* emerge como uma alternativa promissora, oferecendo a capacidade de registrar informações de maneira imutável e permanente.

Este trabalho tem como objetivo central apresentar uma solução inovadora e eficiente baseada em *blockchain* para o registro e garantia de autenticidade de documentos. Em resposta aos desafios contemporâneos, desenvolvemos um contrato inteligente em Solidity, uma linguagem de programação específica para a plataforma *blockchain* Ethereum. Esse contrato inteligente viabiliza o registro seguro de documentos na *blockchain*, estabelecendo um histórico confiável e imutável de todas as transações. Ao explorar as características da *blockchain*, como sua natureza distribuída, transparência e imutabilidade, esta pesquisa busca oferecer uma contribuição valiosa para a segurança e confiabilidade das informações registradas, promovendo relações comerciais e jurídicas mais sólidas e transparentes.

1.1. Contextualização e Motivação

O registro de documentos em um cartório é um procedimento legal realizado para assegurar autenticidade e segurança jurídica a diversos tipos de documentos. O objetivo é

estabelecer um registro público que sirva como evidência da existência, conteúdo, participantes e data do documento, conferindo-lhe validade legal perante terceiros.

Ao registrar um documento em um cartório, um oficial público, conhecido como tabelião, realiza uma análise minuciosa do documento, verificando sua conformidade com as exigências legais e as normas estabelecidas. Após essa análise, o documento é devidamente autenticado e recebe um número de registro único. Esse registro é arquivado no cartório, tornando-se um documento público acessível a qualquer pessoa interessada. O registro em cartório proporciona segurança e confiabilidade às partes envolvidas, pois assegura a existência e a integridade do documento, além de possibilitar sua utilização como prova em processos judiciais ou negociações comerciais.

A aplicação de uma tecnologia capaz de armazenar arquivos de forma imutável representa uma poderosa aliada na proteção e garantia da autenticidade desses documentos. O uso *blockchain* pode ser uma alternativa altamente segura para armazenar documentos importantes, como certificados e escrituras de imóveis, entre outros.

Além disso, caso essa tecnologia permita a redução dos custos associados ao armazenamento e certificação desses documentos, um cenário disruptivo poderia se configurar, com potencial para revolucionar os processos atualmente adotados por cartórios. Essa transformação poderia impulsionar mudanças significativas na operação das instituições, proporcionando benefícios tanto em termos de segurança, quanto de eficiência. A natureza imutável da *blockchain*, combinada com sua capacidade de auditoria acessível a qualquer pessoa, poderia viabilizar melhorias substanciais nesses setores.

Episódios como o incidente ocorrido no estado de Goiás em que documentos foram furtados, tais como, processos e mandados de prisão do Tribunal de Justiça local (O POPULAR, 2023), têm se tornado cada vez mais frequentes. Nesse caso específico, os envolvidos no crime conseguiram acesso ao sistema do Tribunal de Justiça por meio de senhas de ex-estagiários e, posteriormente, deletaram os mandados de prisão. Entretanto, situações semelhantes poderiam ser evitadas caso o tribunal tivesse adotado uma solução baseada na tecnologia *blockchain*, graças a sua característica principal de ser imutável. *Blockchain* é uma tecnologia de registro descentralizado e transparente que permite o armazenamento seguro e imutável de informações, garantindo a integridade e a confiabilidade dos dados.

Ao adotar uma solução baseada em *blockchain*, seria viável rastrear todas as alterações realizadas nos arquivos, assegurando a integridade e autenticidade dos documentos registrados.

Adicionalmente, um histórico imutável de todas as transações e modificações seria registrado, o que dificultaria consideravelmente qualquer tentativa de remoção indevida dos documentos.

A *blockchain* é imutável porque cada informação registrada nela fica permanentemente fixada e protegida contra alterações. Isso acontece porque os dados são organizados em blocos sequenciais e criptografados, e qualquer tentativa de modificar um bloco exigiria a alteração de todos os blocos subsequentes, o que é extremamente custoso computacionalmente de realizar. Dessa forma, as informações na *blockchain* permanecem seguras e confiáveis ao longo do tempo.

Diante desse cenário, a adoção de uma solução baseada em *blockchain* pelo Tribunal de Justiça poderia oferecer uma camada adicional de segurança e confiabilidade, mitigando os riscos de incidentes semelhantes e protegendo a integridade dos registros judiciais. Essa tecnologia apresenta um potencial significativo para melhorar a segurança e a transparência no ambiente jurídico, evitando casos de fraude e garantindo a confiabilidade dos processos judiciais.

Além dos benefícios em termos de segurança devido a imutabilidade pontuada anteriormente, a utilização da tecnologia *blockchain* também pode resultar em uma significativa redução de custos, especialmente no que se refere a registro de documentos. Órgãos públicos e instituições privadas podem se beneficiar da simplificação e agilidade proporcionadas pela tecnologia *blockchain*, eliminando a necessidade de processos complexos e dispendiosos de autenticação de documentos. Isso porque a *blockchain* facilita o registro de informações de forma segura já que todo o processo de registro pode ser feito de forma online sem o envolvimento de cartórios ou empresas de certificação.

Dessa forma, a aplicação da tecnologia *blockchain* no registro de documentos não apenas pode fortalecer a segurança e a confiabilidade dos registros, mas também oferecer uma perspectiva econômica vantajosa, permitindo a adoção de processos mais eficientes e econômicos.

A aplicação da tecnologia *blockchain* na autenticação e registro de documentos pode se mostrar uma solução promissora para reforçar a segurança e a confiabilidade dos registros públicos. Ao evitar fraudes e perdas de documentos importantes, essa abordagem oferece benefícios significativos para a proteção e integridade das informações.

Diante do contexto apresentado, este trabalho pretende responder a seguinte questão: Será que a utilização da *blockchain* pode diminuir os custos associados ao registro de documentos em comparação aos serviços cartoriais tradicionais?

1.2. Objetivo

O objetivo deste trabalho é desenvolver uma ferramenta que utilize contratos inteligentes para registrar documentos, demonstrando a viabilidade da aplicação da tecnologia *blockchain*. O objetivo central é oferecer aos usuários uma interface web integrada à *blockchain*, proporcionando uma solução acessível e econômica para o registro de documentos, com ênfase na redução significativa dos custos cartoriais tradicionais. A proposta visa assegurar a imutabilidade e autenticidade dos registros, promovendo a autonomia do usuário e simplificando o processo de registro.

Neste trabalho, os objetivos específicos são:

a) Elucidar o que é *blockchain*;
b) Justificar a escolha da *blockchain* Celo para esse tipo de aplicação;
c) Desenvolver um contrato inteligente para gerenciar o registro dos documentos na *blockchain* escolhida;
d) Desenvolver uma interface web amigável de forma que o usuário consiga de forma fácil registrar novos documentos e verificar a autenticidade de documentos existentes.

Após vivenciar experiências no ano de 2023, o autor deste trabalho constatou que o processo de registro de documentos em cartório apresenta uma série de desafios significativos. Além de demandar consideráveis recursos financeiros, esse procedimento consome uma parcela substancial de tempo e, não raramente, resulta em desgastes emocionais para os envolvidos. Diante dessa realidade, surge a urgente necessidade de explorar alternativas que não apenas otimizem o processo, mas também o tornem mais acessível e menos dispendioso para os cidadãos.

Nesse contexto, a tecnologia *blockchain* emerge como uma potencial solução capaz de revolucionar a dinâmica dos registros cartoriais, com o objetivo adicional de reduzir significativamente os custos associados a esses processos. Este estudo investiga se a implementação dessa tecnologia inovadora pode ser a chave para mitigar as complexidades e dificuldades enfrentadas nesse processo. A proposta central deste trabalho é analisar a viabilidade e os benefícios vinculados à adoção do *blockchain* como uma ferramenta para

agilizar e simplificar os procedimentos cartoriais, com especial atenção aos impactos econômicos e de eficiência operacional, visando a diminuição dos custos envolvidos.

O objetivo primordial deste projeto é contribuir de maneira substancial para aprimorar o acesso ao sistema de registros, promovendo a eficiência, a transparência e a economia de recursos para os usuários e instituições envolvidas, especialmente no que diz respeito aos custos cartoriais. Simultaneamente, a pesquisa também se dedica a uma análise criteriosa das possíveis limitações e desafios que podem surgir com a implementação da tecnologia *blockchain* nesse contexto específico, oferecendo, assim, um olhar abrangente sobre as implicações práticas de sua adoção.

1.3. Metodologia

Essa pesquisa assume uma natureza aplicada, com objetivos exploratórios, descritivos e propositivos. Os objetivos passam a incluir a exploração de novas possibilidades, a descrição minuciosa das características dos objetos de estudo, e a proposição de soluções ou melhorias. Quanto aos procedimentos técnicos, a abordagem se expande para uma pesquisa bibliográfica, uma pesquisa de campo e a proposta de um artefato, utilizando a metodologia de Design Science Research (LACERDA *et al.*, 2013).

1.4. Resultados esperados

Espera-se com esse trabalho:

- Contribuição para o progresso do conhecimento sobre a aplicação da *blockchain* em novas áreas e contextos;
- Desenvolvimento de um contrato inteligente para o registro de documentos;
- Demonstração da capacidade da tecnologia *blockchain* em otimizar e reduzir custos no processo de registro documental.

1.5. Estrutura do trabalho

O Capítulo 1 introduz este trabalho, abordando suas motivações, método e estrutura de desenvolvimento.

O Capítulo 2 desdobra os princípios e conceitos fundamentais associados à tecnologia *blockchain*, estabelecendo uma base teórica sólida para a compreensão do tema.

No Capítulo 3 são delineados os requisitos da aplicação de maneira abrangente, estabelecendo os parâmetros e necessidades do projeto.

O Capítulo 4 a documentação da aplicação é apresentada, oferecendo uma visão detalhada do seu funcionamento e estrutura.

O Capítulo 5 aprofunda-se na construção da aplicação, discorrendo sobre os softwares utilizados, a arquitetura da interface web e o processo de criação e integração dos contratos inteligentes, enriquecido com capturas de tela ilustrativas da aplicação em funcionamento.

Por fim, o Capítulo 6 encerra o trabalho com as conclusões obtidas e propõe sugestões para pesquisas e desenvolvimentos futuros nessa área.

2. CONCEITOS BÁSICOS

Neste capítulo,-são apresentados os conceitos fundamentais necessários para se obter uma visão geral da tecnologia *blockchain* e as soluções adotadas na proposta deste trabalho.

2.1. Registro de documentos e a tecnologia *blockchain* no Brasil

No Brasil existe um projeto de lei que determina que registro de título, documento e imóveis devem ser feitos em *blockchain*. Caso este seja aprovado isso só irá fortalecer a tese deste trabalho que a *blockchain* pode ser uma forte aliada ou até mesmo uma alternativa aos cartórios no Brasil.

O Projeto de Lei 2876/2020, de autoria do Senador Acir Gurgacz (GURGACZ, 2020) tem como objetivo estabelecer que cada registro de título e documento seja realizado no Sistema Eletrônico de *Blockchain* Nacional de Registro de Títulos e Documentos. Adicionalmente, o projeto visa determinar que cada registro de imóvel seja efetuado no Sistema Eletrônico de *Blockchain* Nacional de Registro de Imóveis, ambos disponibilizados pelo Conselho Nacional de Justiça.

2.2. Custos com cartórios no Brasil

No contexto brasileiro, o processo de registro de documentos apresenta inconvenientes como taxas, filas e a obrigatoriedade de comparecer presencialmente ao cartório, o que implica custos financeiros e demanda tempo dos cidadãos.

No contexto desta questão, realizou-se uma pesquisa no site do Cartório Bruno Quintiliano para avaliar os custos vigentes de diversos tipos de registros em 28 de novembro de 2023. A análise desses custos é crucial para avaliar a viabilidade da substituição dessas práticas pelo emprego da tecnologia *blockchain*.

Os resultados obtidos são os seguintes:

Tabela 1 - Valores de registro no cartório

Tipo de registro	**Preço**
Registro e arquivamento da firma	R$ 10,00
Reconhecimento de firma em documento sem valor econômico	R$ 6,67
Reconhecimento de firma em contratos particulares relacionados a bens imóveis, por assinatura	R$ 51,65

Fonte: Cartório Bruno Quintiliano adaptado pelo autor.

O registro de documentos em cartórios é acompanhado por diversos custos, além das taxas já expostas. Esses custos adicionais, embora não sejam facilmente mensuráveis em termos monetários, desempenham um papel significativo no processo. Um dos custos é o tempo despendido em filas e espera nos cartórios. O processo muitas vezes envolve longas esperas, especialmente em períodos de alta demanda, resultando em perda de tempo para os indivíduos envolvidos. Além disso, o deslocamento até o cartório implica em custos de transporte, como combustível, tarifas de transporte público ou despesas com estacionamento, dependendo da localização do cartório.

Outro custo não mensurável é o estresse e a frustração associados ao processo burocrático. Lidar com a papelada, seguir os procedimentos exigidos e lidar com a complexidade administrativa pode ser cansativo e emocionalmente desgastante para os indivíduos envolvidos.

Portanto, embora os custos monetários sejam uma parte essencial do registro de documentos em cartórios, é importante reconhecer e considerar os custos adicionais, como tempo, deslocamento e impacto emocional, ao avaliar a viabilidade de alternativas mais eficientes e econômicas.

2.3. A tecnologia *Blockchain*

O conceito de *blockchain* foi criado cerca de 20 anos antes de sua primeira aplicação prática, os pesquisadores Stuart Haber e W Scott Stornetta buscavam uma solução para assegurar que documentos não pudessem ser plagiados ou adulterados após sua publicação.

Na década de 80, Stornetta testemunhou um escândalo relacionado à publicação de um resultado notável por um pesquisador proeminente em biologia; esse resultado notável foi o produto de manipulação. A manipulação foi comprovada pela análise de cadernos de laboratório nos quais os resultados originais foram alterados usando uma tinta diferente daquela usada nos demais cadernos. Stornetta ficou preocupado com o futuro do registro de informações, pois seu trabalho na Xerox indicava que, no futuro, todos os registros seriam digitais. Ele também sabia que registros digitais eram facilmente modificáveis após a publicação (BHARATHAN, 2020).

A solução criada por eles veio a público através do artigo “How to Time-Stamp a Digital Document” (HABER; STORNETTA, 1991). O propósito inicial era desenvolver um sistema no qual os *timestamps* de documentos fossem invulneráveis a adulterações ao longo do tempo.

Com base nesse trabalho, o artigo "Bitcoin: A Peer-to-Peer Electronic Cash System" foi publicado em 2008 (NAKAMOTO, 2008), assinado pelo pseudônimo de Satoshi Nakamoto. Nessa obra, é delineado o conceito que ficou conhecido como *blockchain*. Apesar de, originalmente, as palavras "*block*" (bloco) e "*chain*" (cadeia) terem sido utilizadas de forma separada no artigo de Nakamoto, a expressão consolidou-se como uma única entidade.

2.3.1. *Definição*

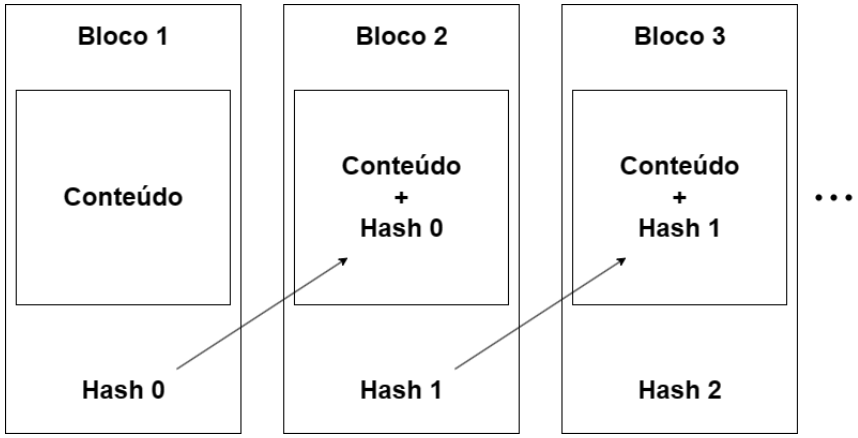
A *Blockchain* é uma estrutura de dados compartilhada que permite o armazenamento de transações de forma digital e descentralizada, sem a necessidade de uma autoridade central. Dessa forma, diversos usuários podem efetuar alterações simultâneas no registro de transações, que é conhecido como livro razão. Essa tecnologia é baseada em uma cópia do livro razão, que é distribuída para todas as máquinas conectadas na rede e armazena todas as transações realizadas. Como resultado, diferentes versões do livro razão são criadas, cujas alterações podem ser verificadas pelos usuários em sua própria máquina (ANDONI *et al.*, 2019).

Todas as mudanças realizadas na rede são validadas pelos usuários através do livro razão e é altamente improvável que qualquer transação seja desfeita. Isso ocorre porque as novas transações são sempre vinculadas às transações anteriores no código, proporcionando transparência e confiança aos contratos (ANDONI *et al.*, 2019).

Na *blockchain*, os dados são organizados em blocos e replicados em todas as máquinas da rede, em vez de serem armazenados centralmente. A manutenção da consistência desses dados em todos os computadores é assegurada pelo algoritmo de consenso, que será abordado na Seção 2.9.

A *blockchain* guarda suas informações de forma comparável a uma lista encadeada, seguindo uma ordem sequencial a partir do bloco inicial. Cada bloco subsequente mantém consistentemente a referência ao bloco anterior, incorporando alguns dados específicos como transações dos usuários e o *hash* próprio do bloco, conforme ilustrado na Figura 1.

Figura 1 - Diagrama de uma *blockchain*



Fonte: Elaborado pelo autor.

Os hashes apresentados na Figura 1 desempenham o papel de funções matemáticas que têm a responsabilidade de transformar um dado específico de entrada, podendo ser uma mensagem ou um arquivo, gerando um código alfanumérico criptografado a partir desse dado.

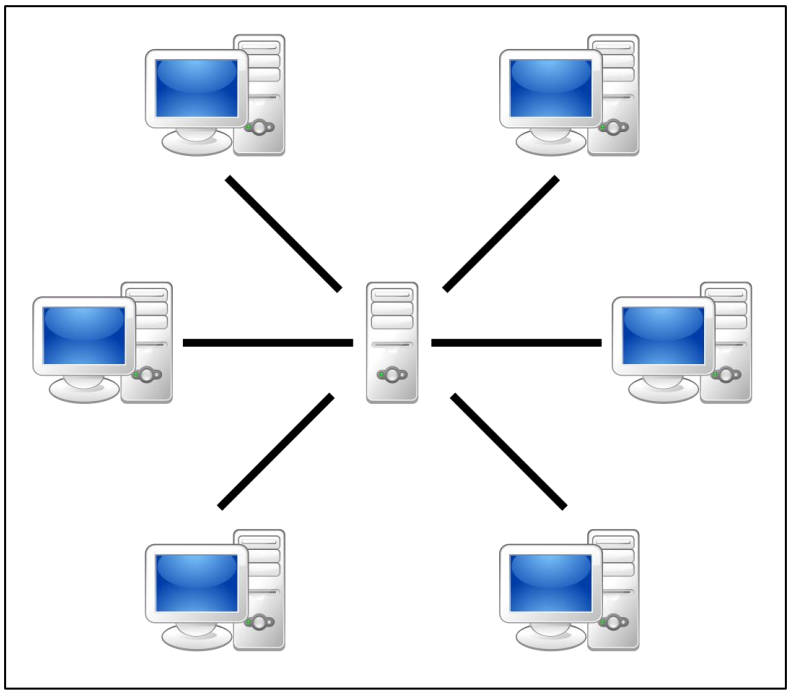
Quando um novo bloco é gerado, o *hash* dele também inclui a assinatura do *hash* do bloco anterior. Isso cria uma espécie de selo, e se ocorrerem alterações em qualquer um dos blocos anteriores, é possível invalidá-lo. Todos os hashes são registrados no livro razão, e uma vez registrados, não podem ser removidos (ANDONI *et al.*, 2019).

2.3.2. *Arquitetura*

Ao discutir sobre *blockchain*, torna-se imprescindível compreender essa tecnologia em conjunto com a arquitetura de uma rede descentralizada. Isso é ainda mais relevante quando

estamos habituados a arquiteturas mais básicas, onde, por exemplo, os usuários de um aplicativo específico simplesmente se conectam ao servidor da empresa e o utilizam sem realmente entender como ele opera de fato. Este sistema opera sob a arquitetura cliente-servidor, caracterizada por um computador central, o servidor, que atende e processa as solicitações feitas pelos clientes. Estes, por sua vez, se conectam diretamente ao servidor, permitindo uma interação eficiente e organizada entre as diferentes partes do sistema como mostra a Figura 2.

Figura 2 - Ilustração de uma rede Cliente-Servidor.



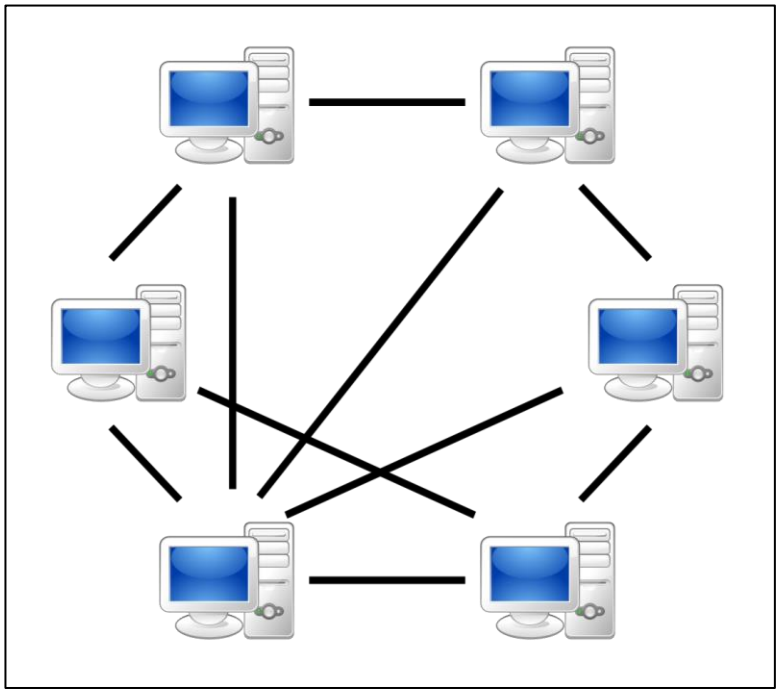
Fonte: Mauro Bieg, Public domain, via Wikimedia Commons.

O modelo cliente-servidor se distingue significativamente de um sistema ponto a ponto (P2P), no qual cada computador possui a flexibilidade de atuar tanto como cliente quanto como servidor. Essa dinâmica permite que em um sistema P2P, cada nó da rede possa solicitar e fornecer recursos, contrastando com a estrutura mais hierárquica e centralizada do modelo cliente-servidor.

A ideia fundamental de uma rede de compartilhamento de arquivos P2P (Peer-to-Peer) é que diversos computadores se unem e combinam seus recursos para formar um sistema de distribuição de conteúdo como mostra a Figura 3. Muitas vezes, esses computadores são simplesmente computadores domésticos, não sendo necessário que sejam máquinas em centros de dados da Internet. Os computadores são chamados de "peers" porque cada um pode alternadamente atuar como cliente para outro peer, buscando seu conteúdo, e como servidor,

fornecendo conteúdo para outros peers. O que torna os sistemas peer-to-peer interessantes é a ausência de uma infraestrutura dedicada. Todos participam na tarefa de distribuir conteúdo, e frequentemente não há um ponto central de controle (TANENBAUM; WETHERALL, 2010).

Figura 3 - Ilustração de uma rede P2P.



Fonte: Mauro Bieg, Public domain, via Wikimedia Commons.

A estrutura Peer-To-Peer é fundamental para as *blockchains*, visto que o título do documento no qual a primeira criptomoeda foi apresentada é intitulado "Bitcoin: A peer-to-peer electronic cash system." (NAKAMOTO, 2008).

2.4. Consenso distribuído

O consenso é um desafio em uma *blockchain* devido à sua natureza descentralizada e distribuída. Em uma rede *blockchain*, não há uma autoridade central que possa tomar decisões unilaterais sobre a validade das transações ou a ordem em que elas são adicionadas ao registro. Em vez disso, várias entidades independentes, conhecidas como nós (“nodes”), participam do processo de validação e consenso.

O problema do consenso surge porque esses nós podem ter interesses divergentes ou podem ser maliciosos. Isso significa que pode haver tentativas de falsificar transações, criar bifurcações na cadeia ou interromper o funcionamento da rede de alguma forma.

Para garantir a integridade e a segurança da *blockchain*, é necessário estabelecer um consenso sobre o estado atual e a validade das transações. Existem diferentes algoritmos de consenso utilizados em *blockchains*, como Proof of Work (PoW), Proof of Stake (PoS), Delegated Proof of Stake (DPoS) e outros. Esses algoritmos definem regras e incentivos para que os nós concordem sobre quais transações são válidas e em que ordem elas devem ser adicionadas ao registro.

Para garantir um bom desempenho em redes *Blockchain*, é necessário escolher os algoritmos levando em conta aspectos como escalabilidade, velocidade de transação, finalização da transação, segurança e consumo de recursos, como energia elétrica. Esses critérios são fundamentais para garantir a operação mais eficiente e segura possível do Sistema (ANDONI *et al.*, 2019).

Em uma rede *Blockchain*, um bloco contendo uma série de transações é criado por um nó e inserido na rede para validação. Os demais *nodes* da rede utilizam o algoritmo definido para chegar a um consenso sobre a validade do bloco. Após essa validação, o bloco é adicionado à *Blockchain* e recebe um endereço associado aos novos blocos criados. Dependendo do algoritmo escolhido, pode haver um período até que o bloco se torne parte permanente da rede, o que ajuda a prevenir ataques maliciosos (ANDONI *et al.*, 2019).

2.4.1. Proof-of-Work (Prova de Trabalho)

Este é o algoritmo de consenso mais amplamente utilizado, inaugurado no lançamento do Bitcoin. Ele funciona como uma prova de que recursos computacionais suficientes foram dedicados para produzir um bloco válido. Nesse modelo, os nós, também conhecidos como mineradores, competem para resolver um problema criptográfico específico. Esse problema consiste em encontrar um valor tal que, quando concatenado com os outros dados do bloco (transações, timestamps, *hash* do bloco anterior etc.) na entrada de uma função *hash*, resulte em uma saída menor que um valor alvo determinado pelo sistema (BASHIR, 2020).

Esse valor é procurado no campo do bloco chamado "nonce", que é incrementalmente ajustado até encontrar um valor que satisfaça a condição estabelecida. Quando essa condição é atendida, um novo bloco é minerado, e o nó responsável por isso é recompensado. Transações são selecionadas, processadas e, por fim, armazenadas nesse novo bloco. Posteriormente, o bloco recém-criado é transmitido para todos os nós na *blockchain*.

O esforço computacional para resolver o problema é exponencial em relação ao número de zeros à esquerda do alvo, e essa resolução pode ser verificada com uma única execução da função hash (NAKAMOTO, 2008). No contexto do Bitcoin, a função *hash* utilizada é a SHA-256. Para manter uma taxa constante de geração de blocos, o valor do alvo pode ser ajustado, sendo aumentado ou diminuído conforme o número de nós na rede. Isso resulta em tornar a resolução mais difícil quando há muitos mineradores e mais fácil quando há poucos.

2.4.2. Proof-of-Stake *(Prova de Participação)*

O Proof of Stake (PoS) difere do Proof of Work (PoW). Trata-se de um mecanismo de consenso no qual o sistema faz a seleção do nó minerador, que terá a capacidade de criar um novo bloco. A forma mais comum de seleção do nó é realizada por meio de um sorteio, em que a probabilidade de um nó ser escolhido é proporcional à quantidade de moedas que ele possui. Uma desvantagem desse mecanismo é a possível concentração de riqueza, uma vez que quanto mais moedas um nó detém, maior é a probabilidade de ser escolhido. Isso resulta em nós com uma alta concentração de moedas tendo mais chances de criar novos blocos e receber recompensas adicionais, ampliando ainda mais suas posses.

Por outro lado, uma vantagem desse mecanismo é que, ao contrário do PoW, não depende do poder computacional, resultando em um consumo de energia significativamente menor. *Blockchains* como o Ethereum e Celo são exemplos que usam a mineração PoS.

2.4.3. Delegate Proof-of-Stake *(Prova de Participação Delegada*)

O Delegated Proof of Stake (DPoS) é um mecanismo de consenso derivado do Proof of Stake (PoS), mas com a distinção de que, nesse caso, as partes interessadas escolhem seus representantes para criar e validar blocos. O número de nós encarregados dessa verificação é consideravelmente menor. Além disso, a geração de blocos é mais rápida se a validação da transação estiver no mesmo formato.

2.5. Funções *hash*

As funções *hash* referem-se a algoritmos que podem converter dados de tamanhos variados em dados de tamanho constante, impedindo a reversão ao dado original com base no *hash* gerado (ANTONOPOULOS, 2014).

Conforme NARAYANAN *et al.*, 2016, uma função *hash* é uma expressão matemática que apresenta as seguintes características:

- O conteúdo de entrada pode variar em tamanho e natureza;
- O conteúdo de saída permanece constante, independentemente das dimensões da entrada. O tamanho do resultado é determinado pelo algoritmo de *hash*, podendo ser de 128, 256, 512 bits, entre outras opções;
- O algoritmo de *hash* deve operar com uma complexidade temporal de O(n), assegurando que, para qualquer entrada n, a geração do *hash* seja realizada em um tempo viável, independentemente das dimensões do texto de entrada.

2.6. Contratos Inteligentes

No ano de 1994, Nick Szabo introduziu o termo contratos inteligentes com o artigo "Smart Contracts" (SZABO, 1994) muito antes do surgimento da tecnologia *blockchain*. O conceito por trás dos contratos inteligentes era criar uma representação digital de contratos

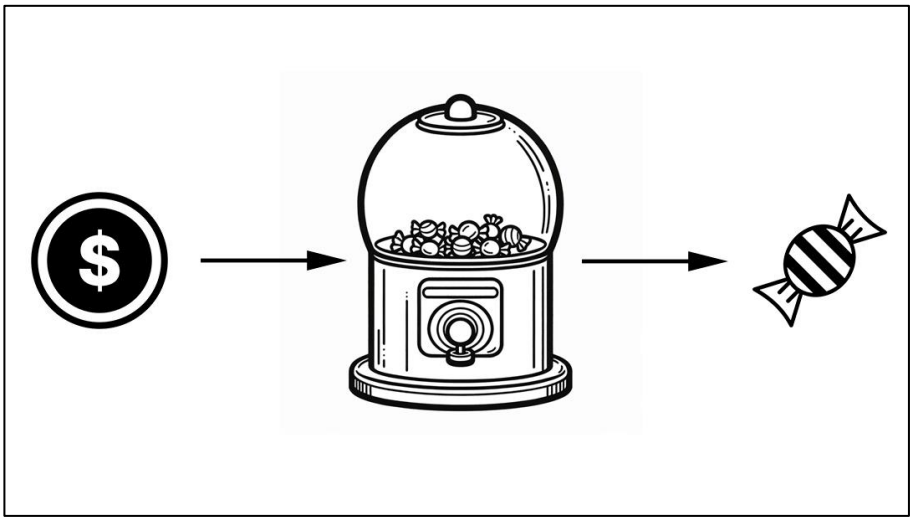
convencionais, visando reduzir a dependência de intermediários e a necessidade de confiança nas transações. Szabo acreditava que essa abordagem poderia trazer maior eficiência, segurança e transparência para diversos setores da economia. Isso se deve ao fato de que um contrato inteligente, sendo um programa de computador autoexecutável, poderia facilitar, verificar ou fazer cumprir negociações ou execuções, dependendo de como foi programado.

O conceito de "contratos inteligentes" ganhou popularidade em 1996 após a publicação de outro artigo intitulado "Smart Contracts: Building Blocks for Digital Markets" (SZABO, 1996) na revista Extropy.

A implementação dos contratos inteligentes não se concretizou até 2009, quando surgiu a primeira criptomoeda, o Bitcoin, junto com a *blockchain*, proporcionando, finalmente, um ambiente propício para os contratos inteligentes. Vale destacar que Nick Szabo concebeu um mecanismo para uma moeda digital descentralizada chamada Bit Gold em 1998. Embora não tenha sido implementada, ela já possuía muitas das características que o Bitcoin destacaria cerca de 10 anos depois (COINTELEGRAPH).

Imagine uma máquina de doces (Figura 4) como um contrato inteligente. Quando um usuário insere moedas (inicia a transação), o contrato (máquina de doces) é acionado. As regras predefinidas, codificadas como algoritmos, garantem que, ao selecionar um doce específico, o usuário receba o item desejado.

Figura 4 - Analogia de um contrato inteligente



Fonte: Elaborado pelo autor.

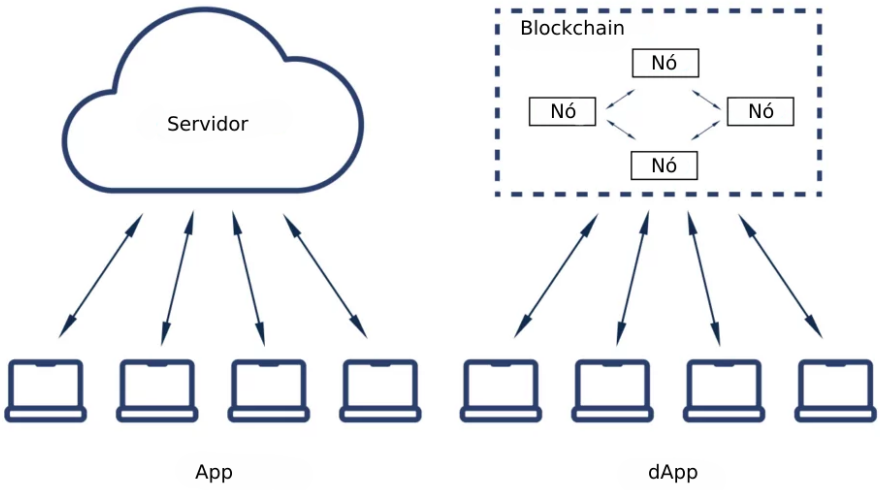
O contrato inteligente automatiza o processo, assegurando a transparência e a execução confiável da transação, sem a necessidade de intermediários.

2.7. Aplicativos descentralizados

Geralmente, quando as pessoas realizam a compra em uma loja virtual por meio de um aplicativo, estão utilizando uma plataforma centralizada. Esses aplicativos correspondem a sistemas de empresas específicas que estabelecem suas próprias taxas e políticas operacionais, além de atuarem como intermediários confiáveis que conectam os usuários com outros indivíduos e estabelecimentos. Por outro lado, as aplicações descentralizadas, também conhecidas como dapps (aplicativos descentralizados), representam um novo modelo de aplicativos desenvolvidos em uma rede descentralizada.

Esses aplicativos combinam contratos inteligentes armazenados em uma *blockchain* com uma interface de usuário *front-end*, conforme exemplificado na Figura 5. Embora as dapps compartilhem semelhanças com as aplicações web em relação à experiência do usuário, a forma como o *backend* opera é totalmente distinta.

Figura 5 - Aplicação centralizada vs aplicação descentralizada.



Fonte: Adaptada de https://www.abtosoftware.com/blog/how-to-create-a-reactjs-dapp-with-the-onflow-emulator

Enquanto aplicativos e sites usualmente utilizam APIs (interfaces de programação de aplicativos) para se comunicarem com seus bancos de dados, nos dapps são empregados contratos inteligentes para interagirem com a *blockchain*.

Pode-se observar na Figura 5, que, em contraste com as aplicações centralizadas, as aplicações descentralizadas processam e armazenam os dados de forma distinta. Em vez de serem armazenados em servidores controlados por uma única entidade central, esses dados são gravados na *blockchain* de forma distribuída e preservados de maneira permanente, ficando acessíveis a todos os participantes da rede. Essa característica intrínseca das aplicações descentralizadas as torna imunes a bloqueios e censuras, garantindo sua disponibilidade ininterrupta, 24 horas por dia, 7 dias por semana.

2.8. Bitcoin

Apesar de a tecnologia de registros distribuídos, conhecida como *Blockchain*, ter sido desenvolvida há algumas décadas, sua popularidade está intrinsecamente ligada ao surgimento da criptomoeda Bitcoin, em meados de 2008, após a publicação do artigo "Bitcoin: A peer-to-Peer Electronic Cash System" (NAKAMOTO, 2008).

O Bitcoin funciona de forma similar a moedas fiduciárias convencionais, como Libra, Dólar e Real (e o futuro real digital, ou Drex). No entanto, suas principais distinções residem no fato de ser uma moeda virtual, sem forma física, e em seu sistema descentralizado, o que elimina a necessidade de intermediários para a validação de transações e escapa da supervisão e controle de qualquer país. Em essência, o Bitcoin desafia a existência de instituições bancárias e retira o poder governamental sobre a moeda, o que tem gerado polêmicas e debates significativos ao longo da última década.

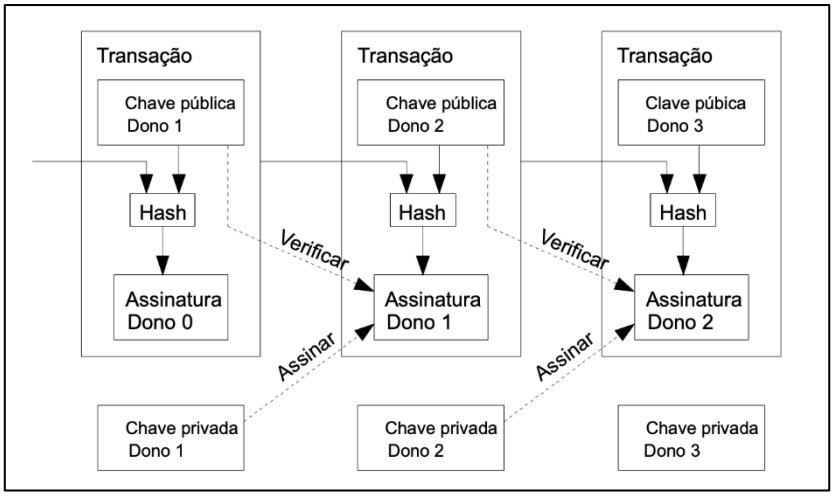
O surgimento do Bitcoin ocorreu durante um período turbulento, a crise financeira nos Estados Unidos em 2008, que resultou no colapso de várias instituições financeiras. Nakamoto, o criador do protocolo original da moeda virtual, desempenhou um papel fundamental nos primeiros anos de desenvolvimento. Em 2010, Nakamoto desapareceu, levando a especulações sobre sua verdadeira identidade. Em 2016, Craig Wright, um empreendedor australiano, afirmou ser o verdadeiro Nakamoto, mas sua alegação foi amplamente contestada e considerada como tendo provas forjadas pela comunidade Bitcoin. No entanto, a identidade do

criador não é de grande relevância, uma vez que o sistema por trás da criptomoeda é descentralizado e não depende de indivíduos ou organizações específicas para funcionar.

Devido à natureza intrínseca de anonimato da *blockchain* do Bitcoin, é difícil determinar com precisão o número de usuários da rede. Porém, pesquisas chegaram a um número de 172 milhões de carteiras de bitcoins ativas (CHAINALYSIS, 2018).

Nakamoto definiu a moeda virtual como uma sequência de assinaturas digitais, em que cada proprietário realiza uma transferência para outro usuário assinando digitalmente um hash da transação anterior, juntamente com a chave pública do próximo proprietário, e adiciona-os ao final da cadeia. No entanto, o destinatário da moeda não possui meios para verificar se o proprietário original a negociou mais de uma vez. A solução comumente adotada envolve uma autoridade central confiável para verificar transações duplicadas, mas esse modelo não está alinhado com a proposta do Bitcoin, que busca eliminar essa necessidade. Portanto, é necessário estabelecer um mecanismo para verificar se os proprietários anteriores não assinaram transações relacionadas à moeda em questão. Nakamoto propôs publicar as transações de forma transparente e estabelecer um sistema em que os participantes concordem com uma única ordem histórica em que as transações foram recebidas.

Figura 6 - Cadeia de transações.



Fonte: https://bitcoin.org/files/bitcoin-paper/bitcoin_pt_br.pdf

A viabilidade de um servidor de carimbo de data/hora (“timestamp”) distribuído em uma rede peer-to-peer (P2P) é alcançada por meio da implementação do mecanismo de Prova de Trabalho (Proof of Work - PoW) no sistema do Bitcoin. Nesse contexto, o Bitcoin utiliza o algoritmo Hashcash desenvolvido por Adam Back (BACK, 2002), o qual envolve a verificação de um valor, como por exemplo o SHA-256. A PoW aborda o desafio de determinar a representação na tomada de decisão por parte da maioria dos participantes. Diferentemente de

uma abordagem baseada em endereços IP, onde um indivíduo mal-intencionado poderia manipular o resultado ao alocar diversos IPs, a PoW atribui a cada unidade de processamento central (CPU) um voto na rede. A decisão da maioria é determinada pela cadeia mais longa, que exige o maior esforço computacional de PoW. Quando a maioria das CPUs é controlada por entidades honestas, a cadeia honesta cresce de forma mais rápida e prevalece sobre eventuais cadeias concorrentes (NAKAMOTO, 2008). Modificar um bloco anterior requer que um invasor refaça toda os cálculos computacionais desse bloco e de todos os blocos subsequentes, e ainda seja capaz de superar os nós honestos na rede.

2.9. Ethereum

Após o êxito na aplicação prática do conceito de *blockchain*, observou-se o surgimento de inúmeras outras *blockchains* com propósitos distintos. Em 2014, Vitalik Buterin lançou o Ethereum, vislumbrando a capacidade de que desenvolvedores criassem aplicativos descentralizados em sua *blockchain* (BUTERIN, 2014). Para alcançar tal objetivo, o Ethereum foi concebido de forma a diferenciar-se significativamente das outras *blockchains* existentes na época. Enquanto as *blockchains* convencionais se limitavam a armazenar informações simples e facilitar transferências de valores, o Ethereum propôs-se a armazenar essas informações, viabilizar transferências e, inovadoramente, executar programas programados especificamente para ele.

O principal obstáculo enfrentado pelo Ethereum era a ausência de uma *blockchain* cuja linguagem de programação fosse Turing-complete. O Bitcoin, por exemplo, não implementou uma linguagem Turing-complete para evitar a ocorrência de loops infinitos durante a execução de transações. A solução encontrada pelo Ethereum para contornar esse problema foi a introdução de uma variável nas transações denominada "Gas" (BUTERIN, 2014).

Com a chegada do Ethereum, surgiu a linguagem de programação Solidity. Por meio dessa linguagem, os programadores podem desenvolver "contratos inteligentes", termo utilizado para designar os programas criados para execução em *blockchains*. Esse avanço representa um marco importante no desenvolvimento da tecnologia *blockchain*, oferecendo uma abordagem mais abrangente e funcional para a implementação de contratos e aplicativos descentralizados.

2.9.1. *Gas*

No ecossistema Ethereum, o termo "gas" representa a unidade de medida utilizada para quantificar o esforço computacional necessário à execução de operações e contratos inteligentes na rede. Cada operação consome uma quantidade específica de *gas*, sendo que a soma total de *gas* utilizado em uma transação determina o custo computacional inerente a essa operação.

Essa taxa de gás representa a quantidade de gás utilizada para realizar uma operação específica, multiplicada pelo custo por unidade de gás. A taxa é cobrada independentemente do êxito ou falha da transação.

2.9.2. *Gas Price*

O "gas price" representa a taxa de câmbio entre Ether (ETH) e *gas*. Essa variável indica o preço que o remetente está disposto a pagar por cada unidade de *gas* consumida. Mineradores, responsáveis por processar transações na rede Ethereum, são incentivados a incluir transações em blocos com base no *gas price*. Transações com valores mais elevados são priorizadas, uma vez que os mineradores preferem incluir operações que gerem maiores compensações financeiras. O custo total de uma transação é derivado da multiplicação do *gas* utilizado pelo *gas* price.

2.9.3. *Gas Limit*

O termo "gas limit", em português limite de gás, diz respeito à quantidade máxima de gás que você está disposto a utilizar em uma transação. Estabelecido pelo remetente ao enviar uma transação para a rede Ethereum, este limite representa a quantia máxima de recursos computacionais que o remetente está disposto a alocar para a execução da transação em

questão. Transações que ultrapassam o limite de *gas* são revertidas, e quaisquer alterações realizadas até aquele ponto são anuladas.

2.10. Bitcoin x Ethereum

O Bitcoin e o Ethereum são duas das criptomoedas mais conhecidas atualmente, embora tenham algumas diferenças importantes em suas funcionalidades e objetivos.

O Bitcoin foi a primeira criptomoeda criada e é frequentemente considerado como um meio de pagamento digital descentralizado e reserva de valor. Seu principal objetivo é fornecer uma forma segura e eficiente de realizar transações financeiras peer-to-peer, sem a necessidade de intermediários tradicionais, como bancos. O Bitcoin utiliza *blockchain* para registrar e verificar as transações, garantindo a sua segurança e integridade.

Por outro lado, o Ethereum é uma plataforma descentralizada que permite o desenvolvimento e execução de contratos inteligentes e aplicativos descentralizados (dApps). Enquanto o Bitcoin se concentra principalmente em ser uma moeda digital, o Ethereum visa oferecer uma infraestrutura para a criação de aplicativos descentralizados que possam executar contratos inteligentes de forma autônoma e confiável.

Uma das principais melhorias que o Ethereum trouxe em relação ao Bitcoin é a capacidade de programação. Enquanto o Bitcoin oferece apenas funcionalidades básicas de transação, o Ethereum introduziu uma linguagem de programação Turing-complete que permite aos desenvolvedores criar contratos inteligentes personalizados. Esses contratos inteligentes são autônomos e podem ser executados automaticamente, sem a necessidade de confiar em intermediários.

Outra diferença significativa é a abordagem do Ethereum em relação à escalabilidade. Enquanto o Bitcoin é limitado em termos de número de transações que pode processar em um determinado período o Ethereum está trabalhando em atualizações para melhorar sua capacidade de processamento.

Tabela 2 - Comparação entre o Bitcoin e o Ethereum

Característica	**Bitcoin**	**Ethereum**
Transações por segundo	7	30
Tamanho do bloco	1 MB	- *
Tempo para produzir um novo bloco	10 minutos	12 segundos
Tempo para confirmar uma transação	10 segundos	2 a 10 minutos
Pode executar contratos inteligentes	Não	Sim

Fonte: Elaborado pelo autor.

* o que limita um bloco no Ethereum não é o tamanho do bloco em si, mas sim o *gas limit*.

Em resumo, enquanto o Bitcoin é considerado uma moeda digital e uma reserva de valor, o Ethereum vai além e oferece uma plataforma para o desenvolvimento de aplicativos descentralizados e execução de contratos inteligentes. Com sua capacidade de programação e flexibilidade, o Ethereum trouxe inovações significativas para o espaço das criptomoedas e tem o potencial de revolucionar vários setores com sua tecnologia *blockchain*.

2.11. Problemas do Ethereum

O principal problema enfrentado pelo Ethereum atualmente é a alta taxa de transação, conhecida como taxa de gás (“gas fee”). Essa taxa é cobrada sempre que ocorre uma transação na rede Ethereum e é utilizada para incentivar os validadores a validar e processar as transações. No entanto, devido ao aumento da demanda e ao congestionamento da rede, as taxas de gás do Ethereum têm aumentado significativamente, tornando as transações caras e inacessíveis para muitos usuários.

Essa alta taxa de transação no Ethereum tem impactos negativos em vários aspectos. Em primeiro lugar, encarece o uso do Ethereum como meio de pagamento diário, tornando inviável para pequenas transações ou micropagamentos. Além disso, a alta taxa de transação

limita a escalabilidade da rede, pois dificulta a execução de um grande volume de transações em períodos reduzidos.

A escalabilidade é um desafio crucial para o Ethereum, especialmente quando se trata de aplicativos descentralizados e contratos inteligentes que exigem uma grande quantidade de transações para funcionar de forma eficiente. A alta taxa de transação limita o potencial de adoção em massa do Ethereum e dificulta a criação de soluções escaláveis e acessíveis para os usuários.

Enquanto soluções para esses problemas não são implementadas, os usuários do Ethereum podem optar por utilizar outras *blockchains* compatíveis ou explorar opções de camadas de escalabilidade existentes, como a utilização de redes de segunda camada, que oferecem taxas de transação mais baixas e tempos de confirmação mais rápidos.

2.12. Celo

No cenário atual, a adoção generalizada das criptomoedas como forma de pagamento ainda enfrenta diversas barreiras significativas. Em primeiro lugar, devido às regras determinísticas de oferta e à natureza imprevisível da demanda das moedas, há uma tendência à instabilidade de preços, muitas vezes deflacionários, para as criptomoedas bem-sucedidas. Isso leva os usuários a optarem por utilizar essas moedas como uma reserva de valor, em vez de utilizá-las como meio de troca convencional.

Em segundo lugar, mesmo quando os indivíduos expressam o desejo de utilizar criptomoedas voláteis como meio de pagamento, eles precisam lidar com a tarefa de gerar um par de chaves privada/pública para receber um pagamento e inserir a chave pública de outra pessoa para realizar um pagamento. Embora isso possa parecer obstáculos aparentemente pequenos, a experiência tem demonstrado que até mesmo pequenas diferenças na usabilidade podem resultar em grandes disparidades nos níveis de adoção.

A fim de promover a popularização de um sistema de pagamentos descentralizado, é crucial que o processo de envio de pagamentos seja tão intuitivo quanto enviar uma simples mensagem de texto, além de minimizar a volatilidade da moeda. Por exemplo, quando um consumidor deseja adquirir um café utilizando Bitcoin, ele deve primeiro converter o valor do café em reais para a equivalente quantidade em bitcoin, e em seguida, inserir com precisão o

endereço da carteira (chave pública) do estabelecimento, atentando-se a cada caractere, para evitar erros que possam resultar no envio dos fundos para outra pessoa inadvertidamente.

O protocolo Celo foi desenvolvido com o intuito de solucionar essas questões. A fim de simplificar o envio de pagamentos, o protocolo Celo introduz um esquema criptográfico conhecido como criptografia baseada em endereço, onde os participantes estabelecem correspondências entre números de telefone celular e chaves públicas. Isso permite que os usuários utilizem os números de telefone de seus contatos como chaves públicas, substituindo os extensos e propensos a erros endereços utilizados no Bitcoin e Ethereum.

Para lidar com a volatilidade da moeda, o protocolo Celo apresenta diferentes moedas estáveis, como o cUSD (Celo Dollar) e o cEUR (Celo Euro), que possuem seu valor estabilizado em relação ao dólar americano e ao euro, respectivamente. Além disso, o protocolo Celo também inclui o cREAL, uma moeda estável com preço equivalente ao real brasileiro.

O cREAL é projetado para manter um valor fixo e previsível em relação à moeda fiduciária brasileira, permitindo aos usuários realizar transações utilizando uma moeda digital que está diretamente vinculada ao real brasileiro. Essa estabilidade de preço proporciona confiança aos usuários e facilita o uso cotidiano da criptomoeda para transações no Brasil.

Por exemplo, um usuário brasileiro pode utilizar o cREAL para fazer compras online, pagar contas ou transferir fundos de forma rápida e segura, com o benefício adicional de não estar sujeito à volatilidade das criptomoedas tradicionais. Dessa forma, o cREAL proporciona uma opção estável e confiável para a realização de pagamentos no Brasil, impulsionando a adoção das criptomoedas como meio de troca no país.

Por fim, o protocolo Celo incorpora um mecanismo de recompensas em blocos móveis, onde todos os usuários envolvidos em transações têm a oportunidade de participar da verificação desses blocos, resultando em uma ampla base de participantes e tornando as recompensas em blocos mais acessíveis para os usuários comuns. Em conjunto, esses elementos estabelecem as bases de um protocolo de pagamentos atrativo.

2.13. Celo x Ethereum

Celo e Ethereum são redes diferentes com propósitos distintos. A Celo possui taxas de transação mais baixas do que o Ethereum, variando de cerca de US$ 0,0000063 a US$

0,000009 por transação média. Por outro lado, o Ethereum é mais descentralizado que a Celo, pois possui mais de 9000 nós (https://etherscan.io/nodetracker) em comparação com os 110 nós validadores da Celo (https://stats.celo.org/).

Tabela 3 - Comparação entre a Celo e o Ethereum

Característica	**Celo**	**Ethereum**
Transações por segundo na teoria	200 a 400	30 a 50
Tempo para produzir um novo bloco	5 segundos	12 segundos
Tempo para confirmar uma transação	10 segundos	2 a 10 minutos
Taxa média de uma transação simples	< $0.01 USD	Cerca de $0.40 USD até mais de $16.00 USD em horários de pico

Fonte: Elaborado pelo autor.

2.13.1. *Processamento*

A rede Celo possui uma capacidade estável de processamento de aproximadamente 200 transações por segundo, com potencial para chegar a 400 TPS. Em contraste, o Ethereum começa a enfrentar congestionamento a partir de 30 TPS, o que geralmente resulta em taxas de transação mais altas.

A Celo utiliza o mecanismo de Tolerância a Falhas Bizantinas (BFT) para determinar quais validadores podem publicar blocos. Como resultado, o número máximo de validadores permitidos simultaneamente é de 110.

No Ethereum, as transações tendem a levar mais tempo para serem confirmadas devido à maior demora na produção de blocos. Durante o mercado de baixa, as taxas de transação no Ethereum geralmente são mais baixas. No entanto, durante o mercado de alta, essas taxas aumentam significativamente.

Essas diferenças entre Celo e Ethereum em relação à capacidade de processamento, mecanismos de validação e comportamento das taxas de transação podem influenciar a escolha da plataforma adequada para diferentes necessidades e condições de mercado.

2.13.2. *Dificuldades técnicas*

Ao criar contratos inteligentes nas *blockchains* Ethereum e Celo, a dificuldade técnica é semelhante, uma vez que ambas utilizam a linguagem Solidity para o desenvolvimento desses contratos inteligentes. No entanto, uma diferença significativa está no custo monetário associado ao envio desses contratos em cada *blockchain*.

No Ethereum, as taxas de transação, conhecidas como "gas fees", são geralmente mais altas devido à demanda e ao congestionamento da rede. Portanto, o custo de enviar e executar contratos inteligentes no Ethereum pode ser consideravelmente mais alto em comparação com a Celo.

Por outro lado, na Celo, os custos de transação tendem a ser mais baixos, tornando a plataforma mais acessível para o envio e a execução de contratos inteligentes. Essa diferença de custos pode ser um fator determinante para os desenvolvedores ao decidirem em qual *blockchain* implantar seus contratos inteligentes, especialmente se estiverem lidando com aplicações que envolvam transações frequentes ou de menor valor.

Portanto, embora a dificuldade técnica seja semelhante nas duas *blockchains*, é importante considerar o custo monetário associado ao envio e à execução dos contratos inteligentes. A escolha entre Ethereum e Celo dependerá das necessidades específicas do projeto, do orçamento disponível e do equilíbrio entre os recursos técnicos e os custos operacionais.

3. DESCRIÇÃO GERAL

Para demonstrar como a tecnologia *blockchain* pode garantir a autenticidade de documentos, foi necessário criar uma interface. Essa interface poderia ter sido apenas uma aplicação de linha de comando, mas o autor optou por desenvolver um sistema web para tornar o entendimento mais acessível. Este capítulo oferece uma visão geral do sistema, explica as interfaces e apresenta os requisitos necessários.

3.1. Aspecto geral do produto

O sistema será projetado com uso da linguagem JavaScript em conjunto com o framework React.js com o objetivo de permitir o registro de documentos em PDF e posteriormente a verificação se um PDF já foi registrado anteriormente.

3.1.1. *Interface do usuário*

O sistema será desenvolvido com compatibilidade para qualquer navegador do momento que autor escreve este documento.

3.1.2. *Interface do software*

Para acessar o sistema WEB será necessário um navegador compatível com JavaScript moderno (Edge, Firefox, Chrome, Safari etc.).

3.2. Funções do sistema

O sistema terá as seguintes funções definidas:

- Registro de usuários;
- Realizar login;
- Recuperar senha;
- Registrar documento em PDF;
- Verificar autenticidade de um documento registrado.

3.3. Limitações do sistema

O aplicativo web possui as seguintes limitações:

- Necessidade de conexão com a internet;
- Não é possível consultar o documento original que foi registrado, somente verificar a autenticidade de um registro;
- Não é possível registrar o documento como um todo na *blockchain*, então somente um *hash* SHA256 será registrado.

3.4. Características do usuário

O aplicativo é direcionado ao público geral, trazendo uma interface intuitiva e simples para usuários com pouca escolaridade e pouca experiência com produtos tecnológicos.

4. DOCUMENTAÇÃO DO APLICATIVO

Este capítulo apresenta os critérios para o desenvolvimento do sistema, categorizados em: Requisitos de Usuário - Necessidades (RUN) e Requisitos Funcionais (RF), além do diagrama de casos de uso. Estes critérios foram delineados a partir da simulação de um levantamento de requisitos em uma reunião com um potencial cliente, refletindo as demandas e expectativas identificadas durante o processo.

4.1. Requisitos de Usuários - Necessidades

A Tabela 4 apresenta os requisitos e necessidades apresentados pelo usuário.

Tabela 4 - Requisitos Usuário – Necessidades.

ID	**Descrição**	**Fonte**
RUN 001	O usuário deve ser capaz de se cadastrar.	Autoral
RUN 002	O usuário deve ser capaz de efetuar o login, caso já possua conta.	Autoral
RUN 003	O usuário deve ser capaz de recuperar a senha através do e-mail cadastrado.	Autoral
RUN 004	O usuário deve ser capaz de registrar um arquivo PDF.	Autoral
RUN 005	O usuário deve ser capaz de verificar se um arquivo PDF já foi registrado.	Autoral

Fonte: Elaborado pelo autor.

4.2. Requisitos Funcionais

As Tabelas 5 a 9 apresentam os requisitos responsáveis pela estruturação do aplicativo, apresentando as opções e funções disponíveis ao usuário.

Tabela 5 - RF 001: Cadastrar usuário.

Identificador RF 001	**Nome** Cadastrar usuário
Casos de uso CSU 001	**Autor** Gustavo Toledo de Souza
Descrição O aplicativo deve ter a opção de cadastrar um novo usuário.	
Dependência	**Prioridade** Essencial

Fonte: Elaborado pelo autor.

Tabela 6 - RF 002: Efetuar login.

Identificador RF 002	**Nome** Efetuar login
Casos de uso CSU 002	**Autor** Gustavo Toledo de Souza
Descrição O aplicativo deve ter a opção de realizar login.	
Dependência	**Prioridade** Essencial

Fonte: Elaborado pelo autor.

Tabela 7 - RF 003: Recuperar senha,

Identificador	**Nome**
RF 003	Recuperar senha
Casos de uso	**Autor**
CSU 003	Gustavo Toledo de Souza
Descrição	
O aplicativo deve permitir o usuário recuperar sua senha.	
Dependência	**Prioridade**
	Essencial

Fonte: Elaborado pelo autor.

Tabela 8 - RF 004: Registrar documento.

Identificador	**Nome**
RF 004	Registrar documento.
Casos de uso	**Autor**
CSU 004	Gustavo Toledo de Souza
Descrição	
O aplicativo deve ter a opção de registrar um arquivo PDF.	
Dependência	**Prioridade**
	Essencial

Fonte: Elaborado pelo autor.

Tabela 9 - RF 005: Verificar documento.

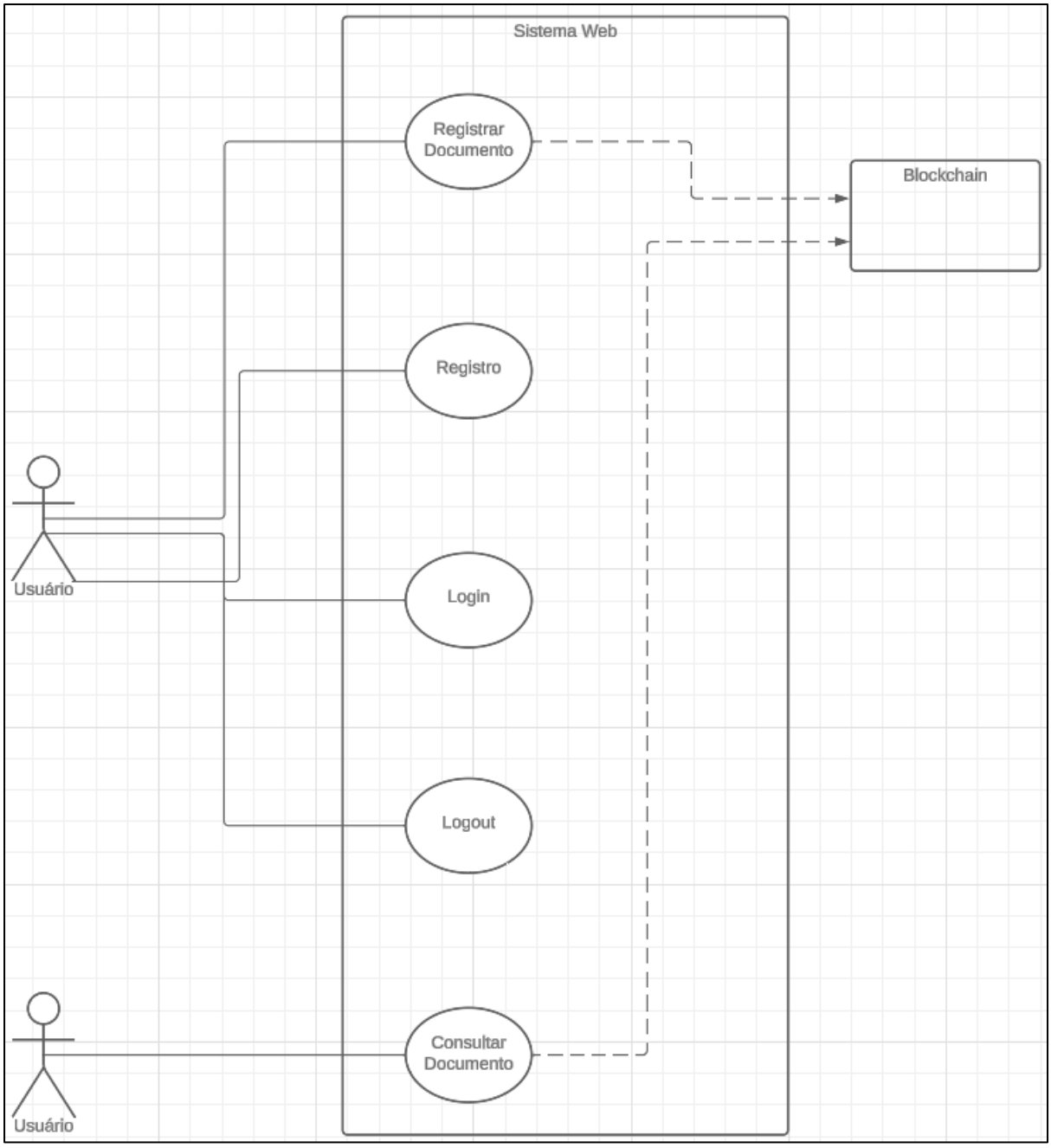
Identificador	**Nome**
RF 005	Verificar documento
Casos de uso	**Autor**
CSU 005	Gustavo Toledo de Souza
Prioridade	
Essencial	
Descrição	
O aplicativo deve ter a opção de verificar se um arquivo PDF já foi registrado.	

Fonte: Elaborado pelo autor.

4.3. Diagrama de casos de uso

A Figura 7 apresenta os casos de uso que serão abordados para a criação do aplicativo.

Figura 7 - Diagrama de casos de uso.



Fonte: Elaborado pelo autor.

4.4. Casos de uso descritivos

Neste capítulo serão apresentados os casos de usos descritivos da aplicação desenvolvida.

Cada caso de uso será composto por um nome, um identificador, requisito funcional, descrição do caso de uso, o ator, pré e pós condições, fluxo principal, fluxo alternativo e cenários de exceção.

As tabelas 7 a 10 apresentam os casos de uso de forma descritiva.

Tabela 10 - CSU 001: Cadastrar usuário.

Nome	Cadastrar usuário
Identificador	CSU 001
Requisito	RF 001
Descrição	Permite que o ator cadastre sua conta no aplicativo.
Ator	Usuário
Pré-condições: O usuário deve possuir acesso à internet e um e-mail válido. **Pós-condições:** O usuário tem sua conta cadastrada no sistema.	
Fluxo principal: 1. O sistema apresenta a tela de cadastro (Figura 9); 2. O usuário preenche os campos “E-mail”, "Senha" e “Nome”; 3. O usuário seleciona a opção "Registrar"; 4. O sistema verifica se o Usuário, o E-mail e as credenciais são válidos; 5. O sistema direciona o usuário para a tela de login e o caso de uso CSU 001 é iniciado.	
Fluxo Alternativo: 1. O usuário seleciona a opção “Faça login; a. O caso de uso CSU 002 é iniciado.	
Cenários de Exceção: 1. O usuário deixa algum campo em branco;	

a. O sistema exibe a mensagem padrão de campo obrigatório do HTML5, no campo que não foi preenchido. 2. O usuário digita um e-mail já cadastrado no sistema; a. O sistema emite a mensagem "Usuário já existe".

Fonte: Elaborado pelo autor.

Tabela 11 - CSU 002: Realizar login.

Nome	Realizar login
Identificador	CSU 002
Requisito	RF 002
Descrição	Permite que o ator realize login no aplicativo.
Ator	Usuário
Pré-condições: O usuário deve possuir uma conta previamente cadastrada no sistema e possuir acesso à internet. **Pós-condições:** O usuário é autenticado e tem acesso ao aplicativo.	
Fluxo principal: 1. O sistema apresenta a tela de login (Figura 8); 2. O usuário preenche os campos "E-mail" e "Senha"; 3. O usuário seleciona a opção "Entrar"; 4. O sistema verifica se o Usuário já está cadastrado e se as credenciais estão corretas; 5. O sistema autentica o usuário e concede acesso ao aplicativo; 6. O usuário é direcionado para a tela de registrar documento e o caso de uso CSU 004 é iniciado.	
Fluxo Alternativo: 1. O usuário clica em “Esqueci a senha”; a. O caso de uso CSU 003 é iniciado. 2. O usuário clica em “Registre-se aqui” a. O caso de uso CSU 001 é iniciado.	
Cenários de Exceção: 1. O usuário deixa algum campo em branco;	

a. O sistema exibe a mensagem “Campo obrigatório não preenchido”, acima do campo que não foi preenchido. 2. O usuário digita um e-mail não cadastrado no sistema; a. O sistema emite a mensagem "E-mail inválido". 3. O usuário digita uma senha diferente da cadastrada no sistema; a. O sistema emite a mensagem "Senha inválido".

Fonte: Elaborado pelo autor.

Tabela 12 - CSU 003: Recuperar senha.

Nome	Recuperar senha
Identificador	CSU 003
Requisito	RF 003
Descrição	Permite que o ator recupere a senha de sua conta no aplicativo.
Ator	Usuário
Pré-condições: O usuário deve possuir uma conta previamente cadastrada no sistema e possuir acesso à internet. **Pós-condições:** O usuário recebe deverá possuir uma nova senha.	
Fluxo principal: 1. O sistema apresenta a tela de recuperar a senha (Figura 10); 2. O usuário preenche o campo “E-mail”; 3. O usuário seleciona a opção “Enviar”; 4. O sistema verifica se o e-mail informado está cadastrado; 5. O sistema envia um e-mail para o usuário com um link para recuperar a senha; 6. O usuário clica no link e é direcionado para a tela “Redefinição de senha”; 7. O usuário preenche o campo “Nova senha”; 8. O usuário seleciona a opção “Redefinir senha”; 9. O sistema salva essa nova senha para o usuário; 10. O usuário é direcionado para a tela de login;	

11. O caso de uso CSU 002 é iniciado.
Fluxo Alternativo: 1. Usuário clica em “Faça login” b. O caso de uso CSU 002 é iniciado.
Cenários de Exceção: 1. O usuário deixa algum campo em branco; a. O sistema exibe a mensagem “Campo obrigatório não preenchido”, acima do campo que não foi preenchido. 2. O usuário digita um e-mail não cadastrado no sistema; a. O sistema emite a mensagem "E-mail inválido".

Fonte: Elaborado pelo autor.

Tabela 13 - CSU 004: Registrar documento PDF.

Nome	Registrar documento PDF
Identificador	CSU 004
Requisito	RF 004
Descrição	Permite que o ator registre um documento em PDF no aplicativo.
Ator	Usuário
Pré-condições: O usuário deve estar logado no aplicativo e possuir acesso à internet. **Pós-condições:** O documento é devidamente registrado.	
Fluxo principal: 1. O sistema apresenta a tela de registro de documento (Figura 13); 2. Usuário seleciona um arquivo PDF; 3. Usuário clica no botão “Registrar” 4. O sistema exibe uma mensagem de sucesso contendo as informações que comprovam o registro.	
Fluxo Alternativo: 1. Não tem.	

Cenários de Exceção: 1. O usuário tenta registrar um arquivo PDF que já foi registrado anteriormente a. O sistema exibe um aviso informando que o arquivo já foi registrado.

Fonte: Elaborado pelo autor.

Tabela 14 - CSU 005: Verificar documento PDF.

Nome	Verificar documento PDF
Identificador	CSU 005
Requisito	RF 005
Descrição	Permite que o ator verifique a autenticidade de um documento PDF no aplicativo.
Ator	Usuário
Pré-condições: O usuário deve possuir acesso à internet. **Pós-condições:** O usuário consegue verificar se o documento foi registrado ou não.	
Fluxo principal: 1. O sistema apresenta a tela de verificação de documento (Figura 17); 2. Usuário seleciona um arquivo PDF; 3. Usuário clica no botão “Verificar” 4. Se o documento selecionado já estiver sido registrado deverá exibir as seguintes informações: bloco em que ele foi registrado e *hash* da transação.	
Fluxo Alternativo: 1. Não tem.	
Cenários de Exceção: 1. O usuário seleciona um arquivo PDF que o *hash* seja diferente dos registrados pelo sistema; a. O sistema exibe um aviso informando que o documento não foi registrado ou pode ter tido o seu conteúdo alterado.	

Fonte: Elaborado pelo autor.

5. RESULTADOS E DISCUSSÃO

Neste capítulo, são apresentados os resultados alcançados até a conclusão do projeto. São discutidos alguns fatores que foram relevantes para a obtenção desses resultados, bem como suas implicações.

5.1. Contrato inteligente

Para finalizar o projeto, foi essencial não apenas desenvolver o aplicativo web, mas também criar um contrato inteligente para armazenar os hashes dos arquivos escolhidos e registrados pelos usuários. A opção de salvar somente o *hash* do arquivo se deu em virtude de uma restrição da *blockchain* Celo, que desaconselha o armazenamento de arquivos extensos. Optou-se, portanto, por preservar apenas o *hash* sha256, calculado a partir dos bytes do arquivo enviado, como medida para contornar essa limitação.

O código completo do contrato inteligente foi disponibilizado no Apêndice A.

5.1.1. *Estrutura de dados*

Para armazenar os dados de um documento PDF foi criado uma *struct* de nome “FileEntry” como pode ser visto na linha 20 do código no apêndice A.

A estrutura (struct) apresentada no código é denominada "FileEntry" e é projetada para armazenar informações relacionadas a um arquivo. Cada instância dessa estrutura contém os seguintes campos:

- “fileName” (tipo: “bytes32”): Este campo armazena o nome do arquivo, utilizando o tipo “bytes32”, que é uma variável de tamanho fixo em Solidity. Essa escolha de tipo é eficiente para armazenar strings com comprimento predefinido.

- “fileSize” (tipo: “uint256”): O campo “fileSize” representa o tamanho do arquivo em bytes e é do tipo “uint256”, indicando que ele só aceitará valores não negativos inteiros.
- “timestamp” (tipo: “uint256”): O campo “timestamp” armazena a marca de data e hora em que o arquivo foi registrado. Aqui o valor de data e hora é armazenado como um carimbo de tempo UNIX, que representa o número de segundos desde 1º de janeiro de 1970.
- “blockNumber” (tipo: “uint256”): Este campo contém o número do bloco no qual a transação relacionada a este arquivo foi registrada na *blockchain*. Isso é útil para rastrear a localização específica da transação na cadeia de blocos.

Essa estrutura fornece uma maneira organizada de armazenar metadados essenciais sobre um arquivo na *blockchain*, facilitando a recuperação e o gerenciamento dessas informações durante a execução do contrato inteligente.

5.1.2. *Variáveis*

A partir da linha 33 do código no apêndice A está definido as variáveis necessárias para o funcionamento do exemplo. A linha 33 do código apresenta uma variável pública do tipo “uint256” denominada “registered”. Esta variável provê o número total de registros realizados e, por ser pública, pode ser acessada externamente por qualquer pessoa.

A linha 38 introduz um mapeamento chamado “fileEntries”, cujo domínio consiste em valores do tipo “bytes32” (que será os hashes dos arquivos) e cujo contradomínio consiste em estruturas denominadas “FileEntry”. Este mapeamento é público, o que significa que é acessível externamente.

A declaração do mapeamento é fundamental para associar os hashes (representados por “bytes32”) a informações específicas sobre os arquivos (representadas pela estrutura “FileEntry”), permitindo uma organização eficiente e acessível dos dados.

5.1.3. *Funções*

Escritas nas linhas 54 e 78 do código no apêndice A estão as duas únicas funções necessárias para o exemplo funcionar.

A função “registerFileHash” localizada na linha 54 é uma função pública que permite o registro de informações associadas a um *hash* de arquivo. A função exige três parâmetros: “fileHash” (o *hash* do arquivo a ser registrado), “fileName” (o nome do arquivo) e “fileSize” (o tamanho do arquivo). Esses parâmetros são fornecidos externamente à função.

A função utiliza o modificador “onlyRole(DEFAULT_ADMIN_ROLE)”, o que significa que apenas uma entidade que possui o papel (role) de administrador padrão pode invocar essa função. Essa restrição de acesso garante que apenas administradores, no caso o próprio sistema, possam registrar novos arquivos no sistema.

Antes de realizar o registro, a função verifica se o *hash* do arquivo já está associado a algum bloco registrado. Isso é feito através da condição “if (fileEntries[fileHash].blockNumber != 0)”. Se o *hash* do arquivo já estiver registrado (ou seja, se o número do bloco associado a esse *hash* não for zero), a função reverte a transação, lançando uma exceção do tipo “FileAlreadyRegistered”.

Se o *hash* do arquivo não estiver previamente registrado, a função cria uma entrada no mapeamento “fileEntries”. A estrutura “FileEntry” associada ao *hash* do arquivo é inicializada com as informações fornecidas: “fileName”, “fileSize”, timestamp do bloco atual (“block.timestamp”), e número do bloco atual (“block.number”).

Após o registro bem-sucedido do arquivo, o contador público “registered” é incrementado.

Essa função serve como um mecanismo de controle para evitar o registro duplicado de arquivos, garantindo que cada arquivo seja associado a um *hash* único e que apenas usuários autorizados possam realizar esse registro.

Já a função “getFileEntryByHash” descrita na linha 78 do apêndice A é uma função de leitura (view) que permite que os usuários obtenham informações associadas a um determinado *hash* de arquivo sem realizar alterações no estado do contrato. A função recebe um parâmetro “fileHash” (o hash do arquivo) e retorna uma tupla contendo informações relevantes sobre o arquivo associado a esse *hash*.

Dentro da função, a entrada correspondente ao *hash* do arquivo é obtida do mapeamento “fileEntries” e armazenada em uma variável local “FileEntry memory file”. Esta variável é do tipo “memory”, indicando que ela só existe durante a execução da função e não persiste no estado do contrato.

A função retorna uma tupla contendo quatro elementos:

- “bytes32”: Nome do arquivo (“file.fileName”).
- “uint256”: Tamanho do arquivo (“file.fileSize”).
- “uint256”:Timestamp associado ao bloco em que o arquivo foi registrado (“file.timestamp”).
- “uint256”: Número do bloco em que o arquivo foi registrado (“file.blockNumber”).

Essa função proporciona uma maneira eficiente de obter informações sobre um arquivo específico usando seu *hash*, sem a necessidade de modificar o estado do contrato. Isso é útil para consultas e verificações de estado.

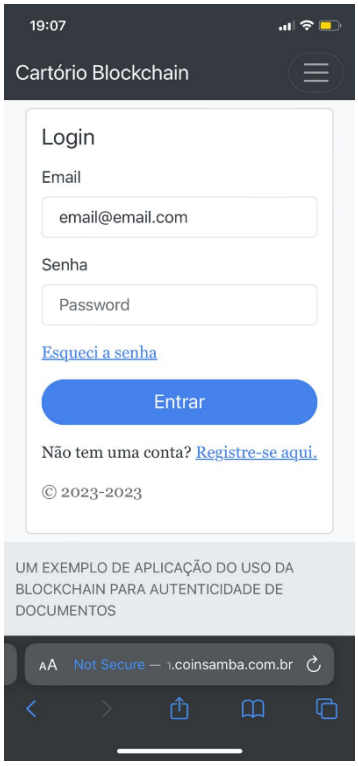
5.2. Telas do Aplicativo

Neste capítulo são apresentadas as telas do aplicativo, descrevendo os seus componentes e as ações disponíveis aos usuários.

5.2.1. *Tela de Login*

A Figura 8 apresenta a tela inicial do aplicativo, onde o usuário tem a opção de realizar o login, recuperar senha ou cadastrar-se.

Figura 8 - Tela de Login



Fonte: Elaborado pelo autor.

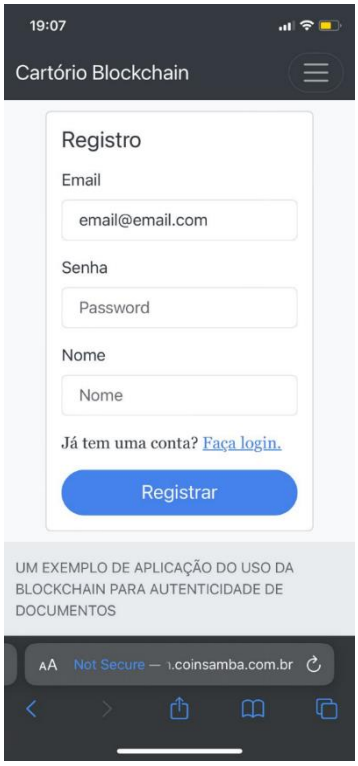
Essa tela é composta pelos componentes:

- E-mail: Campo que o usuário informa um e-mail já cadastrado;
- Senha: Campo que o usuário informa a senha correspondente ao e-mail cadastrado;
- Login: Botão que ao ser pressionado irá realizar a validação dos campos de e-mail e senha. Caso sejam validados, o usuário será redirecionado para a Tela Principal; caso contrário, será exibida uma mensagem contento o erro;
- Esqueci a senha: Botão que ao ser pressionado realiza a validação do campo de e-mail e envia o mesmo o procedimento de recuperação de senha;
- Registre-se aqui: Botão que irá redirecionar o usuário para a Tela de Cadastro.

5.2.2. *Tela de Cadastro*

A Figura 9 exibe a tela onde o usuário realiza o seu cadastro no aplicativo, informando os dados:

Figura 9 - Tela de Cadastro



Fonte: Elaborado pelo autor.

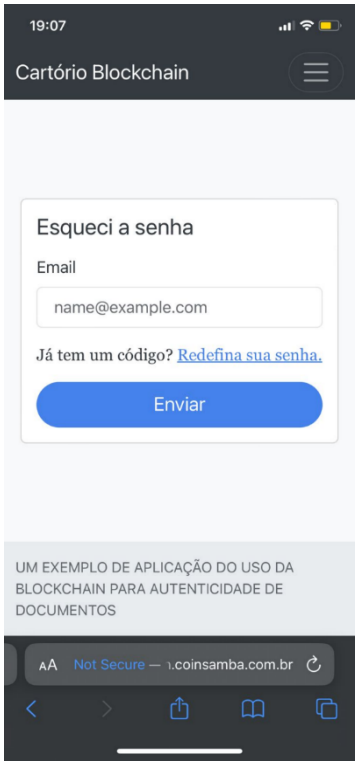
- E-mail: Campo onde o usuário deve informar o seu e-mail;
- Senha: Campo onde o usuário deve informar uma senha;
- Nome: Campo onde o usuário deve informar o seu nome.

Ao pressionar o botão “Criar conta”, o aplicativo irá validar se o e-mail informado já está cadastrando, se o campo “Senha” corresponde ao campo “Confirme a Senha” e se possuem no mínimo 6 caracteres, exibindo uma mensagem de erro para cada caso.

5.2.3. *Tela de Redefinição de Senha*

Ao acessar a tela de redefinição de senha representada na Figura 10 o usuário deve informar seu e-mail no campo de e-mail e pressionar o botão “Enviar”, o aplicativo irá validar se o e-mail é valido e corresponde à algum cadastro.

Figura 10 - Tela de Redefinição de Senha



Fonte: Elaborado pelo autor.

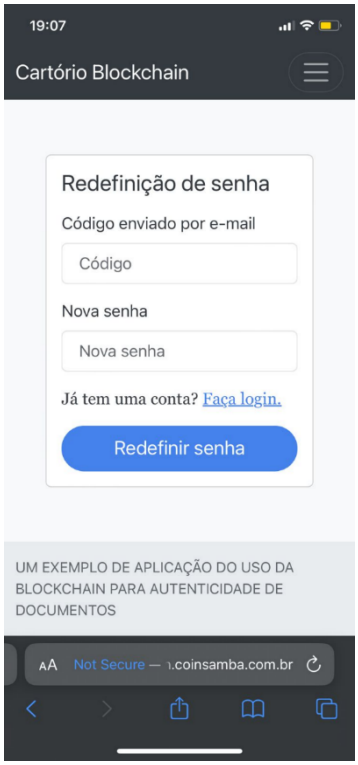
Caso a validação obtenha êxito, o e-mail com as instruções de troca de senha será enviado. O modelo de e-mail que é enviado ao usuário para alterar sua senha de acesso é exibido na Figura 11.

Figura 11 - Modelo de e-mail de redefinição de senha.

Fonte: Elaborado pelo autor.

No e-mail é enviado um token temporário, que é a identificação deste processo de recriação de senha para este usuário específico, pode-se observar que este código também está no final do link enviado ao usuário que direciona para a tela representada na Figura 12.

Figura 12 - Tela de Confirmação de Redefinição de Senha

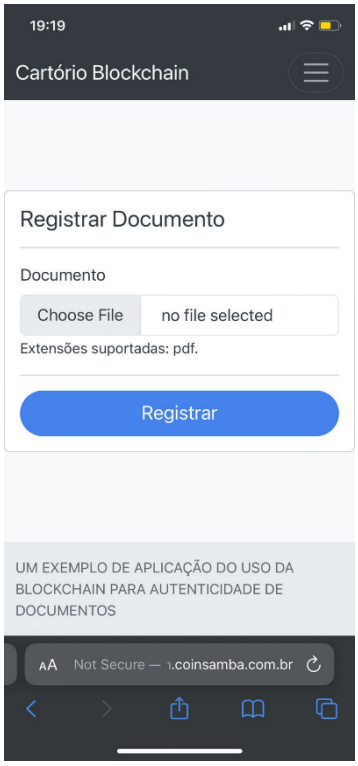


Fonte: Elaborado pelo autor.

5.2.4. *Tela de Registro de Documento*

Ao acessar a tela de Registro de Documento na Figura 13 o usuário deve selecionar um arquivo PDF de seu dispositivo e clicar em “Registrar”, caso o documento n’ao tenha sido registrado anteriormente, ele irá receber um alerta de sucesso com o endereço da transação na *blockchain* comprovando o registro.

Figura 13 - Tela de Registro de Documento



Fonte: Elaborado pelo autor.

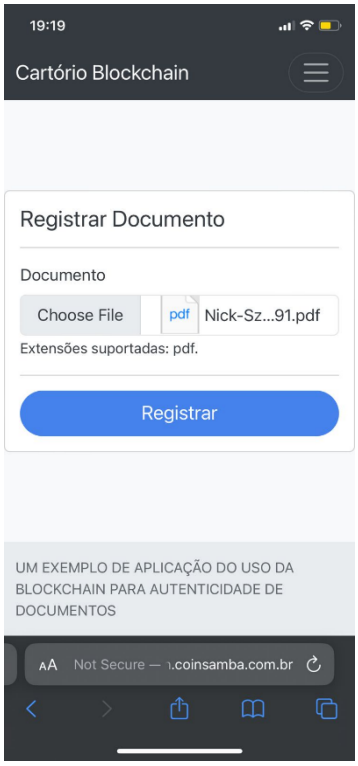
Após o usuário pressionar o seletor de arquivos que na Figura 13 está escrito como "*Choose File"*, em inglês pois o navegador utilizado estava neste idioma, é aberta a interface do sistema operacional para a seleção do arquivo PDF desejado como mostra a Figura 14.

Figura 14 - Interface do Seletor de Arquivos do Sistema

Fonte: Elaborado pelo autor.

Depois de selecionar um arquivo agora a tela exibe o nome do arquivo que o usuário escolheu como mostra a Figura 15.

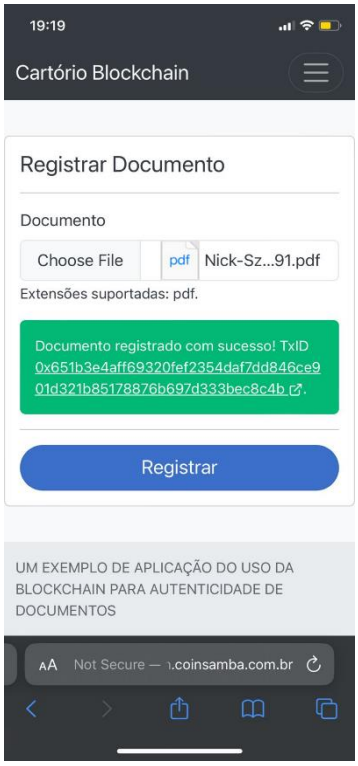
Figura 15 - Documento Selecionado



Fonte: Elaborado pelo autor.

Agora com o arquivo selecionado basta o usuário pressionar em “Registrar” que um alerta de sucesso será exibido contendo o comprovante da transação em que este arquivo foi registrado (Figura 16).

Figura 16 - Confirmação do Registro de um Novo Documento

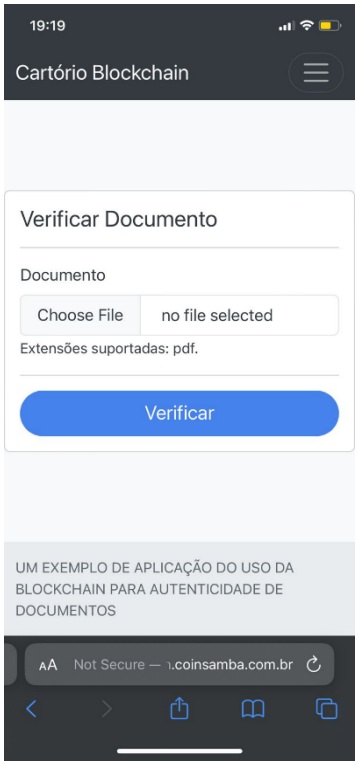


Fonte: Elaborado pelo autor.

5.2.5. *Tela de Verificação de Documento*

A Figura 17 mostra a tela de Verificação de Documento, onde um usuário mesmo sem autenticação consegue escolher um documento de seu dispositivo e verificar se este já foi registrado na plataforma.

Figura 17 - Tela de Verificação de Documento



Fonte: Elaborado pelo autor.

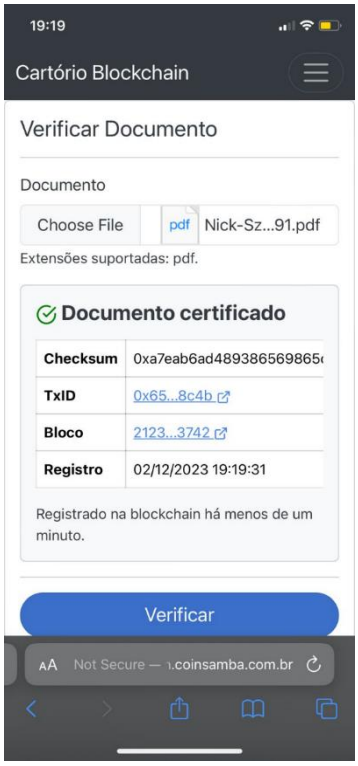
Após o usuário clicar no seletor de arquivos, o seu sistema operacional irá abrir o diálogo de seleção de arquivos como mostra a Figura 18.

Figura 18 - Interface do Seletor de Arquivos do Sistema

Fonte: Elaborado pelo autor.

A Figura 19 mostra que após selecionar um arquivo e clicar no botão "Verificar", no caso de o documento escolhido já ter sido registrado anteriormente pela aplicação, uma tabela é apresentada, contendo informações pertinentes ao arquivo. Estas informações incluem o *hash* (referido como *checksum*), o *hash* da transação (TxID), o bloco em que o documento foi registrado e, por fim, a data em que o arquivo foi efetivamente registrado no sistema.

Figura 19 - Informações de um Documento Registrado



Fonte: Elaborado pelo autor.

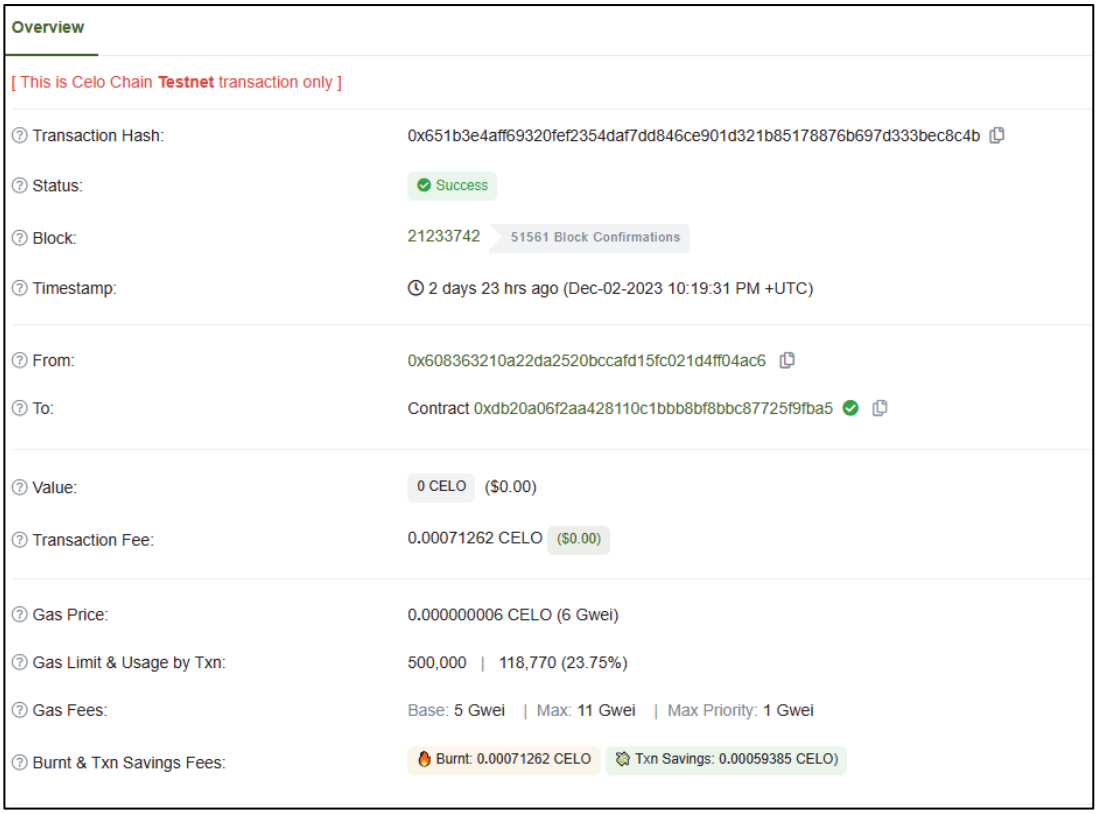
5.3. Custos

Para obter os custos da solução proposta foi enviado o contrato para a *Testnet* da *blockchain* Celo, que é um ambiente semelhante ao da rede normal, com o único benefício que não é preciso efetuar a compra do ativo para os testes.

Foi efetuado o teste do registro de um arquivo PDF, cada passo foi capturado de acordo com as Figuras 12 até a Figura 15 e explicado detalhadamente no tópico 5.2.4.

Após o registro de um documento, o aplicativo web exibe um link que é o *hash* da transação em que o documento foi inserido na *blockchain*, ao acessar este link é exibido uma página como ilustrado na Figura 20.

Figura 20 - Transação resultante do registro de um documento na *blockchain* Celo.



Fonte: CeloScan (https://alfajores.celoscan.io/tx/0x651b3e4aff69320fef2354daf7dd846ce901d321b85178876b697d333bec8c4b).

Ao analisar com detalhes essa transação é possível verificar que a transação custou 0,00071262 CELO para ser executada.

A Figura 21 mostra a conversão da quantidade de CELO consumida anteriormente para reais.

Figura 21 - Conversão da quantidade CELO utilizada para reais.

Fonte: CoinMarketCap. Acesso em 05/12/2023.

O valor em reais em negrito no topo da imagem está a cotação da moeda CELO no dia do teste, bem abaixo, ao final da imagem, está a conversão da quantidade que foi utilizada (0,00071262 CELO) para o equivalente em reais (0,0020 BRL).

5.4. Limitações

É crucial ressaltar as limitações identificadas durante o desenvolvimento deste trabalho. A principal restrição encontrada reside na impossibilidade de armazenar efetivamente um documento na *blockchain* Celo. Diante desse obstáculo, a decisão foi tomada de armazenar apenas o *hash* SHA-256 calculado do arquivo enviado pelo usuário.

Esta limitação emerge devido à desaconselhada prática de armazenar arquivos diretamente na *blockchain*. É imperativo destacar que existem tecnologias mais adequadas para essa finalidade. Além disso, considerando a natureza pública da *blockchain*, é importante reconhecer que qualquer indivíduo com conhecimentos avançados poderia obter acesso ao documento completo, o que motivou a escolha de armazenar apenas o *hash* do arquivo. Essa abordagem preserva a integridade dos dados, mitigando potenciais riscos associados à exposição indiscriminada de informações sensíveis na *blockchain* pública.

5.5. Discussão

Todos os requisitos funcionais propostos foram atendidos, e o teste pôde ser concluído com sucesso.

Ao avaliar os resultados obtidos, confirmou-se a expectativa de que a tecnologia *blockchain* seria capaz de garantir a autenticidade de documentos e proporcionar custos mais baixos em comparação com os cartórios. Surpreendentemente, no entanto, os resultados revelaram uma redução de custos de aproximadamente 3 mil vezes, conforme detalhado na Tabela 15, que apresenta os valores das taxas de transação pagas, como evidenciado na Figura 20. Esses valores foram convertidos com base na cotação vigente no momento da redação, conforme ilustrado na Figura 21. No entanto é necessário um estudo mais aprofundado com um número maior de dados, já que o exemplo foi feito utilizando um único registro.

Tabela 15 - Resultado de custos entre *blockchain* e um cartório

Tipo	**Cartório**	***Blockchain***
Custo aproximado	R$ 6,67*	R$ 0,002

Fonte: Elaborado pelo autor.

* esse custo é mais bem explicado na seção 2.2.

6. CONCLUSÕES E TRABALHOS FUTUROS

Em síntese, o estudo buscou investigar a aplicabilidade da tecnologia *blockchain* como uma alternativa viável para garantir a autenticidade de documentos. A elaboração e implementação bem-sucedidas de uma aplicação prática na *blockchain* Celo destacaram a acessibilidade dessa tecnologia, permitindo que usuários sem conhecimento técnico desfrutem de seus benefícios.

Ao percorrer a trajetória desde as primeiras concepções da *blockchain* até a criação e teste da aplicação proposta, os objetivos específicos delineados foram integralmente cumpridos. Os resultados obtidos evidenciaram a eficácia da solução ao demonstrar que a utilização da *blockchain* pode, de fato, ser utilizada para registro de documentos. Contudo, é crucial ressaltar que algumas limitações, como a inviabilidade de armazenamento integral de arquivos na *blockchain* escolhida, tal limitação pôde ser contornada através da adoção de estratégias como o armazenamento do *hash* dos arquivos.

Para aprimorar ainda mais esse campo de estudo, sugere-se uma análise mais aprofundada do potencial impacto da tecnologia *blockchain* não apenas nos cartórios, mas também em outros órgãos públicos. Investigar de forma mais detalhada os custos e benefícios associados à implementação dessa tecnologia proporcionaria uma visão mais abrangente de sua viabilidade econômica. Além disso, a realização de estudos de caso em uma escala mais ampla permitiria avaliar a eficácia da solução em diferentes contextos, contribuindo para uma compreensão mais completa de suas implicações práticas.

Em última análise, o estudo não apenas cumpriu seus objetivos iniciais, mas também abriu caminho para futuras investigações e aprimoramentos, destacando a relevância contínua da tecnologia *blockchain* no cenário contemporâneo.

REFERÊNCIAS

ANDONI, M. *et al.* Blockchain technology in the energy sector: A systematic review of challenges and opportunities. **Renewable and Sustainable Energy Reviews**, n. 100, Fevereiro 2019., p. 143-174 Disponivel em: https://www.sciencedirect.com/science/article/pii/S1364032118307184. Acesso em: 30 Março 2023.

ANTONOPOULOS, A. M. **Mastering Bitcoin:** Unlock Digital Crypto-Currencies. [S.l.]: O'Reilly Media Inc, 2014.

BACK, A. Hashcash - A Denial of Service Counter-Measure, 1 Agosto 2002. Disponivel em: http://www.hashcash.org/papers/hashcash.pdf. Acesso em: 27 Maio 2023.

BASHIR, I. **Mastering Blockchain:** A deep dive into distributed ledgers, consensus protocols, smart contracts, DApps, cryptocurrencies, Ethereum, and more. 3ª. ed. [S.l.]: Packt Publishing, 2020.

BHARATHAN, V. Blockchain Was Born 20 Years Before Bitcoin. **Forbes**, 1 Junho 2020. Disponivel em: https://www.forbes.com/sites/vipinbharathan/2020/06/01/the-blockchain-was-born-20-years-before-bitcoin/?sh=351bc84d5d71. Acesso em: 15 Junho 2023.

BUTERIN, V. Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform., 2014. Disponivel em: https://ethereum.org/content/whitepaper/whitepaper-pdf/Ethereum_Whitepaper_-_Buterin_2014.pdf. Acesso em: 20 Novembro 2023.

CHAINALYSIS. **Mapping the Universe of 460 Million Bitcoin Addresses**. [S.l.]. 2018.

COINTELEGRAPH. O que são contratos inteligentes? Guia para iniciantes. **Cointelegraph**. Disponivel em: https://br.cointelegraph.com/learn/what-are-smart-contracts-a-beginners-guide-to-automated-agreements. Acesso em: 01 dez. 2023.

FORNI, A. A.; VAN DER MEULEN, R. **Gartner Identifies Three Megatrends That Will Drive Digital Business Into the Next Decade**. Stamford. 2017.

GURGACZ, A. Projeto de Lei n° 2876, de 2020, 2020. Disponivel em: https://www25.senado.leg.br/web/atividade/materias/-/materia/142112. Acesso em: 02 Dezembro 2023.

HABER, S.; STORNETTA, W. S. How to time-stamp a digital document. **Journal of Cryptology**, Janeiro 1991., p. 99-111 Disponivel em: https://rdcu.be/dss3R. Acesso em: 22 Março 2023.

LACERDA, D. P. *et al.* Design Science Research: método de pesquisa para a engenharia de produção. **Gestão & Produção**, São Carlos, 20, 2013., p. 741-761

NAKAMOTO, S. Bitcoin: A peer-to-peer electronic cash system., 2008. Disponivel em: http://www.bitcoin.org/bitcoin.pdf. Acesso em: 22 Março 2023.

NARAYANAN, A. *et al.* **Bitcoin and Cryptocurrency Technologies**. [S.l.]: Princeton University Press, 2016.

O POPULAR. Trio é preso suspeito de furtar processo e mandados de prisão do Tribunal de Justiça de Goiás. **O Popular**, Goiânia, Março. 2023. Disponivel em: https://opopular.com.br/cidades/trio-e-preso-suspeito-de-furtar-processo-e-mandados-de-prisao-do-tribunal-de-justica-de-goias-1.3007874. Acesso em: 20 Julho 2023.

SZABO, N. Smart Contracts, 1994. Disponivel em: https://www.fon.hum.uva.nl/rob/Courses/InformationInSpeech/CDROM/Literature/LOTwinterschool2006/szabo.best.vwh.net/smart.contracts.html. Acesso em: 30 Março 2023.

SZABO, N. Smart Contracts: Building Blocks for Digital Free Markets. **Extropy**, 8, n. 1, 1996., p. 50-53, 61-63 Disponivel em: https://archive.org/details/extropy-16/page/50/mode/2up.

TANENBAUM, A. ; WETHERALL, D. J. **Computer Networks**. 5ª. ed. [S.l.]: Pearson, 2010.

WAZLAWICK, R. **Metodologia de Pesquisa para Ciência da Computação**. 2ª. ed. Rio de Janeiro: [s.n.], 2014.

7. APÊNDICE A – CONTRATO DO REGISTRO DE DOCUMENTOS NA LINGUAGEM SOLIDITY

```
1.  // SPDX-License-Identifier: MIT
2.  pragma solidity ^0.8.21;
3.
4.  import {AccessControl} from "@openzeppelin/contracts/access/AccessControl.sol";
5.
6.  /**
7.   * @title FileHashRegistry
8.   * @dev A contract to register information about files using their hashes.
9.   */
10. contract FileHashRegistry is AccessControl {
11.     /**
12.      * @dev Throws an error when a file is already registered.
13.      */
14.     error FileAlreadyRegistered();
15.
16.     /**
17.      * @dev Struct representing a file entry.
18.      * @param fileName The name of the file.
19.      * @param fileSize The size of the file in bytes.
20.      * @param timestamp The timestamp at which the file was registered.
21.      * @param blockNumber The block number in which the file was registered.
22.      */
23.     struct FileEntry {
24.         bytes32 fileName;
25.         uint256 fileSize;
26.         uint256 timestamp;
27.         uint256 blockNumber;
28.     }
29.
30.     /**
31.      * @dev A counter that keeps track of the number of registered files.
32.      */
33.     uint256 public registered;
34.
35.     /**
36.      * @dev Mapping from file hashes to file entries.
37.      */
38.     mapping(bytes32 => FileEntry) public fileEntries;
39.
40.     /**
41.      * @dev Constructor for the contract.
42.      * Grants the DEFAULT_ADMIN_ROLE to the deployer.
43.      */
44.     constructor() {
45.         _grantRole(DEFAULT_ADMIN_ROLE, msg.sender);
46.     }
47.
48.     /**
49.      * @dev Registers a file hash to the contract.
50.      * @param fileHash The hash of the file to be registered.
51.      * @param fileName The name of the file.
52.      * @param fileSize The size of the file in bytes.
53.      */
54.     function registerFileHash(
55.         bytes32 fileHash,
56.         bytes32 fileName,
57.         uint256 fileSize
```

```
58.     ) external onlyRole(DEFAULT_ADMIN_ROLE) {
59.         if (fileEntries[fileHash].blockNumber != 0) {
60.             revert FileAlreadyRegistered();
61.         }
62.
63.         fileEntries[fileHash] = FileEntry({
64.             fileName: fileName,
65.             fileSize: fileSize,
66.             timestamp: block.timestamp,
67.             blockNumber: block.number
68.         });
69.
70.         registered++;
71.     }
72.
73.     /**
74.      * @dev Returns the file entry for the given file hash.
75.      * @param fileHash The hash of the file entry to be returned.
76.      * @return The file entry for the given file hash.
77.      */
78.     function getFileEntryByHash(bytes32 fileHash)
79.         external
80.         view
81.         returns (
82.             bytes32,
83.             uint256,
84.             uint256,
85.             uint256
86.         )
87.     {
88.         FileEntry memory file = fileEntries[fileHash];
89.
90.         return (file.fileName, file.fileSize, file.timestamp, file.blockNumber);
91.     }
92. }
```