**CENTRO UNIVERSITÁRIO DINÂMICA DAS CATARATAS**

**CURSO DE SISTEMAS DE INFORMAÇÃO**

**PROPOSTA DE UM MÉTODO DE TRANSFERÊNCIA DE POSSE DE BENS UTILIZANDO CONTRATOS INTELIGENTES**

**MARCIO ALEXSANDRO DE PAIVA JUNIOR**

**FOZ DO IGUAÇU – PR**

**2022**

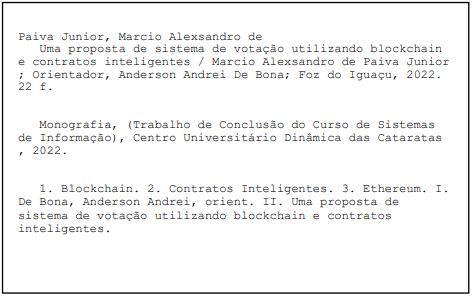
**MARCIO ALEXSANDRO DE PAIVA JUNIOR**

**PROPOSTA DE UM MÉTODO DE TRANSFERÊNCIA DE POSSE DE BENS UTILIZANDO CONTRATOS INTELIGENTES**

Trabalho de conclusão de curso apresentado à banca examinadora do curso de Sistemas de Informação do Centro Universitário Dinâmica das Cataratas – UDC para obtenção do grau de Bacharel em Sistemas de Informação, sob a orientação do Prof. Me. Adélio de Souza Conter

**FOZ DO IGUAÇU – PR**

**2022**

****

**TERMO DE APROVAÇÃO**

**MARCIO ALEXSANDRO DE PAIVA JUNIOR**

**PROPOSTA DE UM MÉTODO DE TRANSFERÊNCIA DE POSSE DE BENS UTILIZANDO CONTRATOS INTELIGENTES**

Trabalho de Conclusão de Curso apresentado ao Centro Universitário Dinâmica das Cataratas UDC, curso de Sistemas de Informação, para obtenção do grau de Bacharel de Sistemas de Informação.

Nota: \_\_\_\_\_\_

Aprovado pela banca examinadora formada por:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Orientador: Prof. Me. Adélio de Souza Conter

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Prof.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Prof.

**FOZ DO IGUAÇU – PR**

**2022**

**AGRADECIMENTOS**

PAIVA JUNIOR, Marcio Alexsandro de. **Proposta de um método de transferência de posse de bens utilizando contratos inteligentes.** Trabalho de Conclusão de Curso (Graduação em Sistemas de Informação) – Centro Universitário Dinâmica das Cataratas, 22p., 2022.

**RESUMO**

Com o passar do tempo, a computação evolui e a cada dia que passa, novas ferramentas tecnológicas surgem para auxiliar o ser humano nas tarefas do dia-a-dia. Uma das tecnologias que mais cresceu nos últimos anos e que atualmente detém grande destaque é a *blockchain*. Este recurso permite a transação de dados de pessoa para pessoa através de uma rede *peer-to-peer* e possibilita a criação de aplicações descentralizadas, os chamados contratos inteligentes. Contratos inteligentes, em contraste com os tradicionais, não têm cunho jurídico, sendo dessa forma um conjunto de promessas descritas em forma de programa de computador que se auto executam. Esta pesquisa teve como objetivo solucionar o problema gerado pela burocracia na transferência de posse de bens. Para isto foi implementado uma aplicação *web* que seja capaz de armazenar os dados de um imóvel em uma *blockchain* e posteriormente transferi-lo para outra pessoa. Os contratos inteligentes disponibilizam funções que são chamadas através de mensagens que podem ser enviadas de contas externas para contas de contratos e entre contas de contrato. Além disso, após compilar um contrato inteligente, o mesmo é migrado para uma *blockchain*, que por sua vez garante que todo o código implementado seja executado exatamente como ele foi programado para executar sem nenhum tipo de interferência externa à rede. Para realizar a fundamentação teórica foram consultados livros e artigos científicos coerentes com o assunto e para o desenvolvimento do protótipo foram utilizados como base as documentações das ferramentas usadas no processo. Os resultados gerados pelos testes com o protótipo demonstraram que é possível virtualizar imóveis através de *tokens* únicos gerados por contratos inteligentes, porém no que diz respeito a validade jurídica desta representação, ainda são necessário estudos. Conclui-se dessa forma que a tecnologia de *blockchain*, bem como os contratos inteligentes, possuem um enorme potencial para descentralizar atividades que necessitam de intermédio e, caso futuramente haja validade jurídica, os contratos inteligentes podem ser uma excelente ferramenta para executar acordos entre duas partes da forma menos burocrática possível.

**Palavras-chave:** *Blockchain*; Ethereum; Aplicações descentralizadas.

PAIVA JUNIOR, Marcio Alexsandro de. **Proposal of a method of transferring ownership of goods using smart contracts.** Undergraduate Final Work (Information Systems Graduation) – University Center Dynamic of Falls, 22p., 2022.

**ABSTRACT**

**Keywords:**

**LISTA DE FIGURAS**

Figura 1 – Fluxograma da metodologia de pesquisa;

Figura 2 – Comando para criação de um projeto React.js;

Figura 3 – Estrutura de pastas de um projeto React.js;

Figura 4 – Estrutura de pastas de um projeto NodeJS;

Figura 5 – Código fonte do arquivo “server.js”;

Figura 6 – Código fonte do arquivo “router.js”;

Figura 7 – Código fonte do arquivo “imoveisRoutes.js”;

Figura 8 – Estrutura de pastas de um projeto Truffle;

Figura 9 – Tela inicial do Ganache;

Figura 10 – Configuração inicial do *workspace* do Ganache;

Figura 11 – Tela principal do Ganache;

Figura 12 – Código fonte do arquivo “truffle-config.js”;

Figura 13 – Código fonte do contrato inteligente em Solidity;

Figura 14 – Código fonte do arquivo “2\_imoveis\_migration.js”;

Figura 15 – Captura de tela com os resultados da migração dos contratos;

Figura 16 – Código fonte do contexto Ethereum;

Figura 17 – Código fonte do arquivo “App.tsx”;

Figura 18 – Adicionar rede na MetaMask;

Figura 19 – Configurações de nova rede na MetaMask;

Figura 20 – Como encontrar a chave privada da conta utilizada na aplicação;

Figura 21 – Importar conta na MetaMask;

Figura 22 – Captura de tela do formulário de cadastro de imóvel;

Figura 23 – Confirmar transação;

Figura 24 – Bloco gerado pela transação;

Figura 25 – Informações do imóvel cadastrado na *blockchain*.

**LISTA DE ABREVIAÇÕES E SIGLAS**

B2B *Business to Business* / Empresa para Empresa;

SHA-256 *Secure Hashing Algorithm 256* / Algoritmo de Hash Seguro;

P2P *Peer-to-Peer* / Pessoa para pessoa;

PoW *Proof of Work* / Prova de Trabalho;

PoS *Proof of Stake* / Prova de Aposta;

FLP Fischer Lynch Patterson;

EVM Ethereum Virtual Machine / Máquinas Virtuais Ethereum;

DApps *Decentralized Applications* / Aplicações Descentralizadas;

EOA *Externally Owned Accounts* / Contas de Domínio Externo;

NFTs *Non-Fungible Tokens.*

**SUMÁRIO**

[**1.** **INTRODUÇÃO** 12](#_Toc117908180)

[1.1. PROBLEMA DE PESQUISA 13](#_Toc117908181)

[1.2. OBJETIVOS 13](#_Toc117908182)

[**1.2.1.** **Objetivo geral** 13](#_Toc117908183)

[**1.2.2.** **Objetivos específicos** 13](#_Toc117908184)

[1.3. JUSTIFICATIVA 13](#_Toc117908185)

[1.4. ESTRUTURA DO TRABALHO 14](#_Toc117908186)

[**2.** **FUNDAMENTAÇÃO TEÓRICA** 15](#_Toc117908187)

[2.1. BLOCKCHAIN 15](#_Toc117908188)

[**2.1.1.** **Bloco** 16](#_Toc117908189)

[**2.1.2.** **Transações** 17](#_Toc117908190)

[**2.1.3.** **Funções hash** 17](#_Toc117908191)

[**2.1.4.** **Elementos base de uma blockchain** 18](#_Toc117908192)

[**2.1.5.** **Protocolos de consenso** 19](#_Toc117908193)

[2.2. CONTRATOS 21](#_Toc117908194)

[**2.2.1.** **Contratos Tradicionais** 21](#_Toc117908195)

[*2.2.1.1.* *Condições de Validade do Contrato* 21](#_Toc117908196)

[*2.2.1.1.1.* *Requisitos Subjetivos* 22](#_Toc117908197)

[*2.2.1.1.2.* *Requisitos Objetivos* 23](#_Toc117908198)

[*2.2.1.1.3.* *Requisitos Formais* 23](#_Toc117908199)

[**2.2.2.** **Contratos Inteligentes** 23](#_Toc117908200)

[**2.2.3.** **Estudo de casos** 25](#_Toc117908201)

[**2.2.4.** **Plataformas** 27](#_Toc117908202)

[**2.2.5.** **Justificativa para a escolha da plataforma Ethereum** 27](#_Toc117908203)

[**2.2.6.** **Ethereum** 28](#_Toc117908205)

[2.2.6.1. Contas 29](#_Toc117908206)

[2.2.6.2. Mensagens e transações 30](#_Toc117908207)

[**2.2.7.** **Tokens e NFTs** 31](#_Toc117908208)

[2.2.7.1. Conceito 31](#_Toc117908209)

[2.2.7.2. Padrões de Token 32](#_Toc117908210)

[**2.2.8.** **Utilização de blockchain em cartório brasileiro** 33](#_Toc117908211)

[**3.** **METODOLOGIA** 34](#_Toc117908212)

[3.1. FORMULAÇÃO DA QUESTÃO DE PESQUISA 34](#_Toc117908213)

[3.2. ESTRATÉGIAS 35](#_Toc117908214)

[**3.2.1.** **Palavras-chave** 35](#_Toc117908215)

[**3.2.2.** **Ambientes** 35](#_Toc117908216)

[**3.2.3.** **Idioma** 35](#_Toc117908218)

[**3.2.4.** **Data de publicação** 35](#_Toc117908219)

[**3.2.5.** **Strings de busca** 36](#_Toc117908220)

[**3.2.6.** **Critérios de inclusão** 36](#_Toc117908221)

[**3.2.7.** **Critérios exclusão** 36](#_Toc117908222)

[**3.2.8.** **Metodologia apresentada em figura** 37](#_Toc117908223)

[**3.2.9.** **Introdução ao protótipo** 38](#_Toc117908224)

[**4.** **PROTÓTIPO** 39](#_Toc117908225)

[4.1. ESTRUTURA 39](#_Toc117908226)

[**4.1.1.** **Ideia proposta** 39](#_Toc117908227)

[**4.1.2.** **Ferramentas** 39](#_Toc117908228)

[4.2. DESENVOLVIMENTO 40](#_Toc117908229)

# **INTRODUÇÃO**

Na atualidade, novas tecnologias surgem a todo momento proporcionando novos tipos de aplicações que são criadas para facilitar as tarefas do dia-a-dia, do trabalho e dos estudos. Para compreender os contratos inteligentes, antes é necessário contextualizar as tecnologias que surgiram posteriormente e servem como base para o funcionamento dos mesmos. Satoshi Nakamoto (2008) apresentou a plataforma Bitcoin, que utilizava *blockchain* para registrar as transações.

*Blockchain,* como o próprio nome sugere, é uma corrente de blocos contendo registros que são ligados entre si através de criptografia. Cada bloco leva consigo os dados de transações, um registro de tempo (*timestamp*) e o código *hash* do bloco anterior, e uma vez que um bloco é inserido em uma *blockchain* é muito difícil de alterar ou deletar o mesmo (ZOU *et al.*, 2019).

Segundo Christidis e Devetsikiotis (2016), *blockchain* atrai interessados de diversas áreas, como por exemplo de finanças, saúde e governamental. O aumento do interesse nesta tecnologia se deve ao fato de que através da mesma, existe possibilidade de executar aplicações de forma descentralizada (que são os chamados contratos inteligentes) que antes só podiam rodar através de um intermediário confiável, desta maneira dispensando a necessidade de uma autoridade central para gerenciar o processo, porém obtendo o mesmo grau de segurança e confiabilidade.

Contratos inteligentes, apesar do nome remeter a área do direito, são códigos de computador executando funções pré-determinadas que geralmente rodam em uma rede distribuída, desta forma tornando possível a implementação de aplicações descentralizadas e abrindo uma vasta gama de possíveis interações e transações de pessoa para pessoa sem intermédio (CHRISTIDIS; DEVETSIKIOTIS, 2016).

Dado o contexto, contratos inteligentes podem ser aplicados para resolver uma série de problemas que normalmente dependem de uma entidade central que garanta confiabilidade em uma transação entre duas partes, porém gerando burocracia na hora de executar a dinâmica. Um exemplo disto é na hora de transferir a documentação de posse de um bem para outra pessoa, sendo necessário passar por uma série de departamentos para verificar a veracidade do documento, bem como o reconhecimento da assinatura do portador do mesmo. Sendo assim, este trabalho tem como objetivo demonstrar uma solução utilizando contratos inteligentes que permite repassar um bem de uma parte para outra sem precisar de uma terceira entidade para intermediar a troca.

* 1. PROBLEMA DE PESQUISA

É possível simplificar o processo burocrático de transferência de posse de bens utilizando contratos inteligentes?

# OBJETIVOS

* + 1. **Objetivo geral**

Implementar, através de contratos inteligentes, uma solução para transferência de posse de bens de uma parte para outra sem intermédio.

* + 1. **Objetivos específicos**
* Pesquisar sobre *blockchain* e contratos inteligentes;
* Estudar a estrutura e o funcionamento de uma *blockchain*;
* Examinar os protocolos utilizados em *blockchains*;
* Observar a estrutura e o funcionamento de um contrato inteligente;
* Implementar um contrato inteligente e interface gráfica que demonstre o caso de uso.

# JUSTIFICATIVA

Cada dia que passa o ser humano se torna mais dependente de tecnologia para realizar as tarefas do dia-a-dia, seja no trabalho, em casa ou nos estudos. Novas aplicações surgem a todo momento para facilitar o jeito como interagimos com o mundo virtual. Diante desta crescente necessidade em conjunto com o alto nível de burocracia na execução de alguns processos, surgem os contratos inteligentes com a proposta de simplificar e descentralizar atividades e aplicações, e possibilitar que as mesmas possam ser realizadas de pessoa para pessoa sem intermédio.

Desta forma, este trabalho se justifica na ideia de solucionar o problema burocrático na transferência de posse de bens através de um método alternativo e permitir que esta ação possa ser efetuada entre duas partes sem uma entidade central confiável.

* 1. ESTRUTURA DO TRABALHO

O capítulo I tem como objetivo fazer uma introdução ao tema abordado, dando uma visão geral sobre quais serão as tecnologias utilizadas para o desenvolvimento da solução e quais os objetivos a serem alcançados para a resolução do problema de pesquisa.

O capítulo II trata de explicar os conceitos citados no capítulo I, deixando tudo especificado para seguir com o desenvolvimento da solução.

O capítulo III demonstra a metodologia utilizada para a busca do material de pesquisa como artigos e livros e explica como foi realizada a filtragem do mesmo.

O capítulo IV diz respeito à implementação da aplicação encarregada de demonstrar um método menos burocrático e descentralizado para transferência de posse de bens, proposta no capítulo I.

O capítulo V faz a conclusão do trabalho mostrando os resultados obtidos.

# **FUNDAMENTAÇÃO TEÓRICA**

# BLOCKCHAIN

*A* tecnologia *blockchain* foi idealizada em 2008 e surgiu para atender a demanda de transações de criptomoedas. É como se fosse um banco de dados, formado por blocos encadeados que tem como objetivo primário de sua criação registrar transferências de *bitcoins* (CARVALHO; ÁVILA, 2019). Nofer *et al.* (2017) definem *blockchain* como um conjunto de dados feito de blocos, onde cada bloco tem várias transações registradas carregando também a identificação do bloco anterior. Ribeiro e Mendizabal (2021) complementam dizendo que uma *blockchain* pode ser vista como um livro razão contábil, que fornece um histórico de todas as trocas de *bitcoins* realizadas ao longo do tempo.

Na prática, *blockchain* é um banco de dados público, distribuído em uma rede peer-to-peer contendo registros encadeados que são acessíveis para todos os nós participantes. Utiliza alguns conceitos, tais como *timestamp*, funções *hash*, assinatura digital e poder computacional para garantir uma criptografia forte o suficiente capaz de inibir qualquer tipo de adulteração nos blocos anexados à cadeia (LUCENA, 2016). Para Nakamoto (2008), a segurança total dos dados é garantida uma vez que a quantidade de nós cedendo poder computacional em prol da proteção da rede é maior que a quantidade de nós que atacam a mesma.

Segundo Greve *et al.* (2018) e Rebello *et al.* (2019) são características de uma *blockchain*:

* Descentralização: as aplicações rodam de maneira distribuída em redes *peer-to-peer*, utilizando protocolos de consenso entre os participantes, excluindo uma organização central que detenha os dados;
* Desintermediação: as transações acontecem pela rede compartilhada, eliminando a necessidade de intermédio por uma terceira entidade na troca de ativos;
* Disponibilidade e Auditabilidade: os dados são compartilhados para todos os nós da rede de forma segura, garantindo que qualquer usuário da rede possa verificar os dados a fim de encontrar erros ou fraudes;
* Imutabilidade e Irrefutabilidade: os blocos adicionados à cadeia são imutáveis garantindo que nenhuma transação possa ser alterada, recriada ou apagada, sendo ela irrefutável uma vez que confirmada, já que cada transação é assinada digitalmente;
* Privacidade e Anonimidade: os usuários têm sua identidade preservada sem que ninguém tenha acesso aos seus dados. Isto é garantido pois esta tecnologia utiliza chaves públicas e privadas que são armazenadas com criptografia em cada transação;
* Cooperação e Incentivos: uma *blockchain* oferece recompensas, geralmente em criptomoedas, aos participantes da rede.

*Blockchains*, segundo Lin e Liao (2017), podem ser divididos em três tipos:

* Público: onde qualquer usuário pode participar do processo de consenso para verificar e validar transações. Ribeiro e Mendizabal (2021) complementam esta afirmativa dizendo que é uma rede descentralizada e aberta para qualquer um que queira participar, tendo sua identidade preservada, sendo aplicável na grande maioria dos sistemas de moedas virtuais, tais como *Bitcoin* e *Ethereum*;
* Privado: quantidade restrita de nós participantes onde os dados são gerenciados por uma ou mais autoridades participantes (LIN; LIAO, 2017). Garante um maior controle sobre a identidade dos usuários descartando anonimato, além de ser centralizado. É utilizado, por exemplo, em redes empresariais, quando se tem a necessidade de identificar os nós e designar suas permissões (RIBEIRO; MENDIZABAL, 2021);
* Consórcio: É uma junção entre o tipo público e privado, garantindo a possibilidade acesso público, entretanto sendo parcialmente centralizado dando privilégios para um número reduzido de nós (RIBEIRO; MENDIZABAL, 2021). Em geral este modelo é utilizado através de parceria entre empresas, seguindo a dinâmica Business to Business (B2B) onde os dados podem ser públicos ou privados (LIN; LIAO, 2017).

# **Bloco**

Como dito anteriormente, um bloco carrega consigo múltiplas transações (NOFER *et al.*, 2017). Ribeiro e Mendizabal (2021) definem um bloco como um arquivo com várias transações registradas. Comparando uma *blockchain* com um livro razão, cada bloco representa uma página.

Nofer *et al.* (2017) afirmam que cada bloco pode ser validado pela rede de nós através de criptografia. Singh e Singh (2016) complementam dizendo que cada bloco é identificado por um código gerado por uma função chamada *hash,* normalmente a SHA-256 (*Secure Hash Function*), e além de carregarem sua identificação única, também são marcados pela identidade do bloco anterior. Levando em consideração estas informações, Nofer *et al.* (2017) definem que alguns atributos são fundamentais para reconhecer um bloco sendo eles o seu *hash* próprio, o *hash* do bloco anterior, um valor *timestamp* e um número inteiro e aleatório denominado *nonce* utilizado para verificação do *hash*.

# **Transações**

Uma transação no contexto de *blockchain* determina uma sequência de operações, levando consigo uma transferência de ativos ou um contrato inteligente (GREVE *et al.,* 2018).

O conceito *blockchain* surgiu para suprir as necessidades de transacionar dinheiro sem precisar de intermédio (NAKAMOTO, 2008). Ainda segundo Nakamoto (2008), dinheiro eletrônico é uma cadeia de assinaturas digitais em que uma pessoa envia uma determinada quantia para outra assinando digitalmente um *hash* gerado pelo *hash* da transação anterior e pela chave pública de quem está recebendo.

* + 1. **Funções Hash**

O termo função *hash* se refere a uma função computacional que comprime uma *string* de entrada em uma *string* de saída, chamada de *hash*, de tamanho fixo e caso utilizada seguindo alguns requerimentos, pode ser empregue em aplicações criptográficas tais como autenticação, assinatura digital e marcação de *timestamp* em arquivos digitais (SOBTI; GANESAN, 2012).

Pierro (2017) define *hash* como uma versão criptografada de uma *string*, sendo impossível de reverter o processo e obter o valor original. Loureiro (2012) reforça essa definição afirmando que funções *hash* são algoritmos de mão única, ou seja, deve ser capaz calcular a saída através de uma entrada, porém sem que seja possível calcular a entrada utilizando a saída. Além disso, qualquer alteração na *string* de entrada resulta em um *hash* totalmente diferente (RIBEIRO; MENDIZABAL, 2021).

* + 1. **Elementos Base de uma blockchain**

A tecnologia de *blockchain* idealizada por Satoshi Nakamoto, tem como objetivo principal realizar transações financeiras de pessoa para pessoa de forma segura, entretanto desde seu surgimento em 2008, diversas aplicações surgiram a partir desta ideia (MIERS *et al.*, 2019). Ainda segundo Miers *et al.* (2019), mesmo que os detalhes de uma *blockchain* variem de aplicação para aplicação, este tem como base dois elementos:

* Integridade: é garantida devido ao modo como uma *blockchain* é projetada, utilizando blocos encadeados contendo registros e metadados, onde cada um deles carrega o *hash* do bloco anterior e, para calcular o seu próprio identificador, junta todas essas informações gerando seu próprio *hash*. Caso ocorra qualquer alteração em um bloco, mesmo que mínima, todos os blocos subsequentes são alterados, sendo necessário muito poder computacional para completar a operação;
* Protocolos de consenso: garantem a consistência da cadeia através de algoritmos que validam bloco por bloco antes de anexá-los à uma blockchain.

Uma *blockchain* garante que o conteúdo e ordem dos blocos não possa ser alterado, entretanto sozinha não é capaz de validar os dados nele inseridos. Desta forma, fica a cargo da camada de aplicação realizar esta tarefa, definindo regras que assumam a veracidade dos dados inseridos nos blocos. Em plataformas de transferência de criptomoedas, duas regras básicas são respeitadas para que uma transação seja considerada válida:

* + As moedas que participam da troca devem estar em posse do dono e não ter sido gastas em uma transação anterior;
  + A transação deve conter a assinatura digital do dono, aprovando a transferência.

Estas duas regras simples garantem que não ocorra o problema de gasto duplo com criptomoedas, entretanto as regras são implementadas de acordo com as necessidades de cada transação (MIERS *et al.*, 2019).

Miers *et al.* (2019) afirmam que essas regras podem ser utilizadas também para aplicabilidades que envolvam transferência de ativos digitais, como a compra de uma arte ou propriedade intelectual, entretanto quando os ativos fazem parte do mundo real, é necessário que o mesmo seja virtualizado de forma que ele exista tanto no mundo físico quanto no digital. Exemplos de objetos que podem ser virtualizados são imóveis, através de sua matrícula e veículos utilizando seu chassi.

* + 1. **Protocolos de Consenso**

Protocolos de consenso existem bem antes da tecnologia de *blockchain*. É na verdade um problema antigo de computação distribuída a fim de garantir acordos de maneira democrática, dando a confiança do que é acordado e distribuído. Em qualquer contexto aplicado, inclusive em *blockchain*, sua função é certificar de que as regras estabelecidas entre usuários de uma rede sejam seguidas de forma rigorosa (MIERS *et al.,* 2019).

O protocolo de consenso é uma parte fundamental de uma *blockchain*. Segundo Rebello *et al.* (2020), este é um algoritmo utilizado a fim de garantir a evolução do sistema adicionando blocos válidos à corrente. Este processo garante que todos os usuários participantes da rede tenham a mesma visão ao aceitar um bloco válido e anexá-lo à cadeia. Cada nó participante da rede carrega consigo uma cópia da cadeia e cada dado que está inserido é verificado de forma individual a fim de validá-los de acordo com as regras propostas na camada de aplicação e após um bloco ser verificado por um participante, o mesmo é replicado aos restantes (MIERS *et al.*, 2019).

Os subcomponentes de um mecanismo de consenso são apontados por Tasca e Tessone (2019), sendo eles:

* Tipo da rede de consenso: centralizada, utilizada por *blockchains* privados; parcialmente centralizada, usada por *blockchains* consórcio; descentralizada, empregue em *blockchains* públicos;
* Comunicação entre os nós: participantes de uma rede devem trocar mensagens para que haja consenso sobre qual cadeia é a correta;
* Acordo de consenso: são um conjunto de regras sob as quais os nós de um sistema distribuído operam de forma individual para que a rede funcione de forma coerente;
* Consenso imutável e tolerante a falhas: mesmo que um ou mais participantes deixem a rede, seja de propósito ou por falha, a aplicação deve continuar rodando (MIERS *et al.*, 2019). Além disso, é necessário um mecanismo de consenso eficiente que garanta a existência de apenas uma cadeia original visível para todos os participantes da rede (TASCA; TESSONE, 2019).

Desta forma, existem alguns algoritmos de consenso em destaque no estado atual da tecnologia de *blockchain*, sendo eles:

* *Proof of Work* (PoW): Os nós (chamados de mineradores) competem entre si utilizando poder computacional para encontrar a melhor criptografia para um bloco (MIERS *et al.*, 2019);
* *Proof of Stake* (PoS): uma alternativa ao PoW que consome menos recursos, tais como eletricidade e processamento. Este algoritmo prioriza a participação na rede onde cada usuário que deseja participar do consenso deve ceder uma parte de seu capital como promessa de que é um nó honesto. Quanto maior este valor, maiores as chances de este participante realizar um consenso (MIERS *et al.,* 2019).

Devido a descentralização de uma *blockchain*, cada usuário pode ter uma visão diferente do estado atual da cadeia. O protocolo de consenso faz com que os participantes da rede estejam de acordo com a ordem dos blocos inseridos, garantindo que todos enxerguem a mesma versão da *blockchain* e, no contexto da troca de ativos, evita o problema de gasto duplo sem necessidade de uma organização intermediadora para verificar possíveis fraudes (MIERS *et al.* 2019). Rebello *et al.* (2020) complementam dizendo que o objetivo primário dos protocolos de consenso é dispor consistência, garantindo que a cadeia seja idêntica para todos os nós participantes da rede, e vivacidade uma vez que todas as rodadas do consenso adicionam novos blocos.

Segundo Fischer *et al.* (1985) apud Rebello *et al.* (2020), um dos grandes desafios da dinâmica de consenso para sistemas distribuídos é o chamado resultado FLP (nome dado em homenagem aos autores Fischer, Lynch e Patterson, formuladores do teorema). Este resultado traz como prova que o consenso não é determinístico em redes assíncronas, como por exemplo a *internet*. Neste caso, a aposta é investir em redes de comunicação síncronas, fazendo com que os protocolos foquem apenas em consistência enquanto que o sistema de comunicação garante a entrega das mensagens, assim atendendo a propriedade de vivacidade.

Os protocolos de consenso são divididos em dois tipos: determinístico e probabilístico. Os determinísticos focam na consistência deixando de lado a vivacidade evitando assim bifurcações, entretanto compromete o sistema a possíveis travamentos caso o mesmo utilize comunicação assíncrona. Já os protocolos probabilísticos tem enfoque na vivacidade em troca da consistência, fazendo com que funcionem em redes assíncronas, como a *internet*, porém estão sujeitos a bifurcações (REBELLO *et al.* 2020).

* 1. CONTRATOS

# **Contratos Tradicionais**

Um contrato é, em sua natureza, um negócio jurídico composto por no mínimo duas partes sendo, desta forma, bilateral ou plurilateral se diferindo do conceito unilateral, uma vez que sua força provém das vontades de apenas um indivíduo. Assim, contratos bilaterais e plurilaterais abrangem os interesses de duas ou mais partes envolvidas (GONÇALVES, 2019).

Aquino (2021) conceitua contrato como um mecanismo jurídico de relações interpessoais, que compete aos interesses entre as partes que o compõem, levando em consideração as leis do código civil e os contextos situacionais em que o contrato foi firmado. Em outras palavras, é um vínculo criado entre duas ou mais entidades, sejam elas físicas ou jurídicas, composto por regras que visam proteger os interesses de ambas as partes.

# *Condições de Validade do Contrato*

Segundo Gonçalves (2019), para que um contrato gere efeitos possibilitando aquisição, modificação ou extinção de direitos, deve atender a alguns requisitos obrigatórios, que caso preenchidos executam as ações previamente estipuladas e em caso contrário o contrato se torna inválido. Aquino (2021) complementa afirmando que as condições podem ser de duas naturezas:

* Geral: presente em todos os contratos;
* Especial: referente a cláusulas específicas para cada contrato, tais como consentimento entre duas partes e vontades acordadas.

Gonçalves (2019) e Aquino (2021) apontam os três requisitos que devem ser preenchidos por um contrato. São eles: Requisitos subjetivos, requisitos objetivos e requisitos formais.

# *Requisitos Subjetivos*

Diniz, citada por Aquino (2021) separa os requisitos subjetivos em quatro partes, sendo elas: a existência de duas ou mais pessoas no exercício do contrato; capacidade das partes em cumprir o contrato; aptidão específica para contratar; consentimento entre ambas as partes. Já Gonçalves (2019) divide em três partes sendo: dois ou mais interesses em cumprir o contrato e capacidade genérica dos contratantes; aptidão específica para contratar e consentimento entre ambas as partes. Desta forma, Aquino (2021) analisa os requisitos subjetivos repartindo-os em quatro partes, que são: as partes do contrato; capacidade genérica; legitimidade; consentimento.

As partes do contrato devem ser compostas por duas ou mais entidades, sendo pessoas físicas ou jurídicas, caracterizando um contrato bilateral ou unilateral respectivamente (AQUINO, 2021).

A capacidade genérica diz respeito à aptidão entre as partes do contrato em cumprir com seus direitos e obrigações estipulados tanto no código civil quanto nas próprias cláusulas do contrato (GONÇALVES, 2019).

A legitimidade é de suma importância no âmbito contratual pois ela filtra a capacidade genérica de tal forma que mesmo que uma ou mais partes tenham disposição para cumprir com o contrato, em certos casos, a lei limita o poder atuação de quem o solicita, como no caso de venda de bens de ascendente (pai) para descendente (filho), salvo exceção caso os outros descendentes consentirem com o ato (AQUINO, 2021).

O consentimento deve ser de livre e espontânea vontade entre ambas as partes podendo ter sua validade alterada ou anulada em caso de defeitos no contrato, tais como erro, coação, estado de perigo, lesão e fraude (GONÇALVES, 2019). Aquino (2021) complementa afirmando que não há firmação de contrato quando apenas uma das partes está interessada.

# *Requisitos Objetivos*

Tanto Aquino (2021) quanto Gonçalves (2019) afirmam que os requisitos objetivos dizem respeito ao objeto do contrato, que deve ser lícito, possível, determinado ou determinável, podendo ser de natureza móvel, imóvel, material ou imaterial.

O objeto lícito é aquele que não apresenta nenhuma característica que se oponha a lei, moral ou bons costumes. O objeto imediato é aquele que se refere a operação contratual, e o objeto mediato diz respeito aos bens ou vantagens geradas pelo contrato (GONÇALVES, 2019).

Ainda sobre o objeto, ele deve ser possível, seja de forma física ou jurídica e em caso de impossibilidade o contrato é anulado. Impossibilidade física diz respeito a leis físicas ou naturais que impossibilitam a execução do contrato. Já a impossibilidade jurídica aponta problemas legislativos que negam a execução de um contrato (AQUINO, 2021).

Gonçalves (2019) aponta também outra característica que especifica a determinação do objeto. Isto é, ele deve ser determinado ou determinável, devendo existir parcialmente ou totalmente no ato da execução do contrato.

# *Requisitos Formais*

Aquino (2021) descreve requisitos formais como a forma em que a vontade do contratante é reconhecida. Gonçalves (2019) complementa dizendo que no direito brasileiro, em regra, a forma é livre, ficando a cargo das partes do contrato decidir como o mesmo é firmado, podendo ser particular ou público, de forma verbal ou não em caso da necessidade de maior segurança e seriedade do contrato garantido por lei de forma escrita.

# **Contratos Inteligentes**

O termo contrato inteligente parece, à primeira vista, remeter a contratos jurídicos, entretanto não é desta maneira que este recurso se comporta (ANTONOPOULOS; WOOD, 2018). Lin e Liao (2017) descrevem contrato inteligente como um contrato digital que controla ativos digitais do usuário, descrevendo e garantindo os direitos e obrigações do mesmo de forma automática através de computação.

Segundo Nick Szabo, apud Antonopoulos e Wood (2018), a definição de contrato inteligente é um conjunto de promessas especificadas digitalmente em que as partes envolvidas as cumpram. Em outras palavras, é um contrato (não necessariamente jurídico ou legal) capaz de se executar por si só. Antonopoulos e Wood (2018) utilizam o termo contrato inteligente para se referir a programas de computador imutáveis rodando em uma *blockchain*.

Carvalho e Ávila (2019) consideram o sistema de máquinas de vendas automática como o pontapé inicial dessa tecnologia, se tratando de uma execução automática da seguinte regra: uma pessoa depositava uma quantia em dinheiro e a máquina devolvia um item. Esta regra levou Szabo a pensar em uma forma de que um contrato se executasse por conta própria sem necessidade de intermédio.

O objetivo de Nick Szabo no desenvolvimento da ideia de contratos inteligentes, tem como base o fato de existir possibilidade de ambiguidade em contratos gerando desentendimento entre as partes (DIVINO, 2018). Naquela época, Nick não encontrou recursos computacionais suficientes para implementação de sua ideia, deixando-a adormecida por muito tempo (ABIJAUDE *et al.*, 2021). Abijaude *et al.* (2021) também afirmam que após o surgimento e o crescimento da plataforma Bitcoin, em 2013 Vitalik Buterin idealizou uma nova, chamada de Ethereum, ao qual utilizava uma nova criptomoeda chamada ether, entretanto com uma diferença para a primeira de que além de transacionar ativos, também executa códigos de programação chamados de contratos inteligentes que nada mais são do que aplicações descentralizadas.

Desta forma, Antonopoulos e Wood (2018) definem as seguintes características para um contrato inteligente:

* Programa de computador: contratos inteligentes são apenas aplicações escritas através de código computacional. Desta forma, não tem caráter legal neste contexto;
* Imutável: contratos inteligentes rodam em *blockchain*, uma vez implantados na rede, não podem ser alterados, diferente dos *softwares* tradicionais. A única forma de alterar um código é gerando um novo contrato e aplicando ele a cadeia;
* Determinista: o resultado gerado pela execução de um contrato inteligente é o mesmo para qualquer usuário que rode o mesmo, de acordo com o contexto da transação e o estado atual do *blockchain* *Ethereum* no presente momento;
* Contexto da Máquina Virtual *Ethereum* (*Ethereum Virtual Machine ou* EVM): contratos inteligentes operam em um contexto extremamente limitado, de forma que os mesmos podem acessar seu próprio estado, o contexto da transação que os chamou e algumas informações sobre os blocos mais recentes adicionados à cadeia;
* Computador mundial descentralizado: cada nó da rede *Ethereum* executa um EVM de forma local, entretanto todos operam no mesmo estado inicial e gerando o mesmo estado final. Desta forma pode-se dizer que o sistema todo atua como único formando um computador mundial descentralizado, operando sobre uma estrutura *blockchain*.

A imutabilidade proporcionada por uma *blockchain* faz com que contratos inteligentes sejam uma excelente escolha para aplicações que necessitem de consistência na execução e segurança de que o código escrito vai de fato ser executado sem nenhuma interferência externa a rede, seja proposital ou não.

Shuai Wang *et al*. (2018) apontam outras três características de um contrato inteligente, sendo elas:

* Autonomia: os contratos inteligentes se auto executam, de forma que os agentes iniciais não precisem atuar durante a execução do código de contrato;
* Auto suficiência: os recursos utilizados por um contrato inteligente são gerenciados pelo mesmo. Assim sendo, ele consegue controlar seu poder de processamento e armazenamento de acordo com a demanda;
* Descentralização: contratos inteligentes não existem em um único servidor controlado, eles são distribuídos e executados nos nós da rede de uma *blockchain*.

# **Estudo de casos**

Quem conceituou os contratos inteligentes foi Nick Szabo, em 1994, que dizia que os termos de um contrato poderiam ser executados através de protocolos de transação computacionais, dando a ideia de que um acordo entre duas partes pudesse ser de fato realizado apenas entre as mesmas, mesmo que sem confiança entre ambas (MOHANTA; PANDA; JENA, 2018). Desta forma, segundo Mohanta, Panda e Jena (2018) existem diversos em que os contratos inteligentes podem ser aplicados, como por exemplo:

* Cadeia de suprimento: uma cadeia de suprimentos é composta por diferentes níveis de transação, cada um com suas próprias condições, como por exemplo, processamento da comida, transporte e venda. Utilizando os contratos inteligentes se torna possível ter um registro mais transparente de cada uma das etapas deste longo processo da mesma forma em que dispensa o controle por uma entidade central;
* Internet das Coisas: é uma área em constante evolução a qual ainda são realizadas pesquisas para utilização em conjunto com contratos inteligentes. Uma vez que haja a possibilidade de integração, os dispositivos IoT se tornam autônomos;
* *Healthcare*: com os padrões de vida do ser humano avançando dia após dia, surgem novos dispositivos capazes de monitorar sinais vitais que capturam os mesmos e de forma muito rápida e prática mostram esses dados ao usuário em tempo real. Utilizando contratos inteligentes, se abrem muitas possibilidades de uso deste dado de forma que se mantenha a privacidade de quem os detenha. Como por exemplo, um contrato inteligente pode ser utilizado para leitura destes dados e indicar possíveis causas em tempo real;
* Finanças: tradicionalmente, as instituições financeiras são utilizadas como intermédio para movimentar dinheiro. Porém utilizando contratos inteligentes esta necessidade pode ser descartada uma vez que *blockchains* possibilitam transações de pessoa para pessoa, como no caso da plataforma Bitcoin.Apesar desde ideia necessitar de amadurecimento, à medida que novas pesquisas são realizadas, é um caso de uso interessante;
* Imobiliária: a transferência de posse de bens dentro do sistema imobiliário tradicional envolve muita burocracia e etapas antes de realmente ocorrer a execução do acordo entre duas partes. Utilizando contratos inteligentes, é possível que toda a documentação seja transferida de forma automática e totalmente descentralizada de pessoa para pessoa, uma vez que o objeto em questão esteja com os documentos virtualizados e inseridos em uma *blockchain*.
  + 1. **Plataformas**

De acordo com Zheng *et al.* (2020), são alguns exemplos:

* Hyperledger Fabric*:* se difere da Ethereum pois os contratos inteligentes rodam em contêineres *Docker* ao invés de máquinas virtuais e além disso esta plataforma adota o modelo de *blockchain* privado, onde todos os participantes da rede devem ser identificados e autorizados a participar. Suporta linguagens de programação de alto nível tais como Java e Go;
* Corda: plataforma focada em moedas digitais, sendo como um livro razão distribuído que processa e registra as transferências de ativos, semelhante à *Bitcoin*. Também suporta linguagens de programação de alto nível como Java e Kotlin, desta forma rodando a aplicação em Java Virtual Machine (JMS), além de ser Turing incompleto e adotando o modelo de *blockchain* privado;
* Stellar*:* também focada em moedas digitais, dando suporte a diversas linguagens de programação como por exemplo JavaScript, Python, Go e PHP. Assim como a plataforma Corda, também é Turingincompleto, rodando códigos em Docker, reduzindo sobrecarga, desta forma possibilitando transações mais rápidas e baratas e adotando o modelo privado de *blockchain*;
* Ethereum: é uma plataforma de *blockchain* pública capaz de rodar contratos inteligentes, desenvolvidos através da linguagem de programação solidity, em máquinas virtuais Turing completas chamadas de *Ethereum Virtual Machines*. Utiliza a criptomoeda chamada Ether tanto para transações de pessoa para pessoa quanto para rodar os contratos inteligentes.
  + 1. **Justificativa para a escolha da plataforma Ethereum**

Conforme citado na seção anterior, Ethereum, Hyperledger Fabric, Corda e Stellar têm atuações distintas. Em primeiro lugar, a escolha da Ethereum se dá pelo fato de a plataforma ser Turing completo, possibilitando a criação de contratos de alto nível com a utilização de laços de repetições e condições. Além disso, conforme citam Valenta e Sandner, dentre as quatro plataformas apontadas, a única que utiliza o estilo de *blockchain* público é a Ethereum, que possibilita o desenvolvimento de aplicações descentralizadas, facilitando transações de pessoa para pessoa. Hyperledger Fabric e Corda atendem casos de uso mais específicos. A primeira tem foco em setores como o bancário, saúde e cadeia de suprimentos, enquanto a segunda foca em serviços financeiros e industriais (VALENTA; SANDNER, 2017).

# **Ethereum**

Ethereum é uma plataforma *blockchain* de código aberto capaz de rodar aplicações chamadas de contratos inteligentes através de máquinas virtuais descentralizadas. Também utiliza um sistema monetário para realizar as transações, moeda que é chamada de *ether*, entretanto apesar das semelhanças com *bitcoin*, suas finalidades se diferem (LIN; LIAO, 2017). Segundo Karatas (2018), o que diferencia a plataforma Ethereumdas outras é que se tem a possibilidade de desenvolver e executar contratos inteligentes e aplicações descentralizadas (*DApps*), podendo serem escritos em linguagens de programação próprias, sendo a mais famosa *Solidity*, onde o código é compilado, convertido em bytecode e enviado para o *blockchain* Ethereum.

Em 2013, Vitalik Buterin, um Russo-Canadense de 18 anos, idealizou uma rede baseada em *blockchain*, que tivesse como função armazenar códigos de computador, executando funções que alcançariam áreas muito além do *Bitcoin* (RIBEIRO; MENDIZABAL, 2021). Karatas (2018), assim como Ribeiro e Mendizabal (2021), citam que em 2014, os investimentos na plataforma Ethereum chegaram a cerca de US$ 18 milhões em 42 dias pela oferta inicial de moedas proposta pelos idealizadores da rede. Desta forma, em 2015 Gavin Wood, Jeffrey Wilcke e Vitalik Buterin lançam o projeto.

Segundo Buterin (2014), a plataforma Ethereum une os conceitos de programação, cripto-ativos e capitalização de mercado, permitindo que programadores criem aplicações baseadas em consenso que garantam escalabilidade, padronização, integridade de recursos, facilidade de desenvolvimento e interoperabilidade, características essas proporcionadas através de uma *blockchain* construída sob uma linguagem de programação Turing-completo, possibilitando que qualquer usuário escreva contratos inteligentes e aplicações descentralizadas com suas próprias regras de propriedade, formatos de transação e funções de mudança de estado. Uma vez compilados, os contratos inteligentes são convertidos em *bytecode* e migrados para a blockchain (HEGEDUS, 2019).

Segundo Antonopoulos e Wood (2018), a Ethereumé considerado Turing-completo uma vez que é capaz executar programas armazenados em uma *Ethereum Virtual Machine* (EVM), enquanto lê e escreve dados na memória, entretanto isto é um problema se tratando de uma rede aberta como uma *blockchain*.

Sistemas Turing-completo podem rodar programas que apresentem *loops* infinitos, em outras palavras, um sistema que roda infinitamente através de um laço de repetição, seja por erro de programação acidental ou de propósito. Desta forma, a Ethereum não pode prever quando um contrato inteligente vai terminar sua execução e acerca disso, os nós da rede devem validar todas as transações e rodar todos os contratos inteligentes que são solicitados. Isso pode ocasionar um travamento da rede, já que os nós podem ficar presos a um contrato que está rodando infinitamente (ANTONOPOULOS; WOOD, 2018). Para resolver este problema, Buterin (2014) escreveu em seu artigo o conceito de *gas*, uma espécie de combustível utilizado para executar contratos.

# Contas

Na plataforma Ethereum, o estado do *blockchain* é feito de objetos chamados contas, cada uma contendo um endereço formado por 20 bytes e as mudanças de estado sendo transações de valores entre as contas (BUTERIN, 2014). Existem dois tipos de contas que são: Contas de Domínio Externo (*Externally Owned Accounts* ou EOA), esta que se assemelha a uma conta bancária de pessoa física, na qual existe um saldo de ether que está disponível para realizar transações que são controladas pela chave privada da conta; e Contas de Contrato, que são mais complexas do que contas jurídicas já que além de possuírem um saldo de ether, também possuem o código, chamado de contrato inteligente, associado a ela, que é executado sempre que acontece uma solicitação de mensagem ou quando uma transação é realizada (RIBEIRO; MENDIZABAL, 2021).

Buterin (2014) afirma que uma conta Ethereum é composta por 4 atributos:

* Um *nonce*, um número que garante que cada transação só pode ser processada uma vez;
* O saldo atual de ether;
* O código do contrato inteligente (para Contas de Contrato);
* O armazenamento (vazio por padrão).

A diferença entre os dois tipos de conta é que uma conta externa é controlada por chave privada e não tem contrato inteligente vinculado a ela, desta forma possibilitando que as contas troquem mensagem criando e assinando transações, enquanto que uma conta de contrato é controlada pelo seu código de contrato que é executado sempre que a conta recebe uma mensagem, permitindo leitura e escrita no seu armazenamento interno próprio (BUTERIN, 2014). Um fato curioso é que sem as contas de contrato, a rede Ethereumseria praticamente igual a rede *Bitcoin*, onde só seria possível usuários trocarem ether entre si. Desta forma, a diferença entre as duas plataformas é justamente a possibilidade de rodar códigos computacionais imutáveis de alto nível através de máquinas virtuais (RIBEIRO; MENDIZABAL, 2021).

* + - 1. Mensagens e Transações

As mensagens na rede Ethereum apresentam três características: Primeiro, podem ser enviadas tanto por contas externas quanto por contas de contratos; Segundo, as mensagens devem conter dados; Terceiro, se o destinatário de uma mensagem for uma conta de contrato, a chamada pode retornar resposta, abrangendo o conceito de funções (BUTERIN, 2014). Ainda segundo Buterin (2014), o termo transação na rede Ethereum se refere a pacotes de dados assinados, contendo mensagens que são enviadas por contas externas à cadeia. Os componentes de uma mensagem são o destinatário, a assinatura do remetente, a quantidade de ether, os dados a serem enviados e dois valores chamados *STARTGAS e GASPRICE.*

O *gas* funciona como um combustível para rodar contratos inteligentes, apresentando sua própria taxa de câmbio em relação ao ether*.* O *gas* é utilizado para pagar o processamento das chamadas de um contrato, desta forma sendo isolado da cotação do ether no mundo real (ABIJAUDE *et al.*, 2021). Sempre que alguém fazer uma requisição, seja criação de um contrato, solicitação de uma mensagem, acesso aos dados de contas e execução de funções na EVM, é cobrado um imposto chamado de *fee*, que é calculado pela equação *fee = gasUsed \* gasPrice*, na qual o *gasPrice* é o preço proposto a pagar pela transação e o *gasUsed* é um número inteiro representando a quantidade exata de *gas* usado que deve ser menor que o *STARTGAS* (RIBEIRO; MENDIZABAL, 2021).

O *gas* resolve o problema gerado pela completude de Turing em um sistema público e distribuído como uma *blockchain*. Cada instrução tem um valor determinado de *gas* máximo a ser utilizado, desta forma a EVM termina a execução de um contrato inteligente caso o *gas* usado ultrapasse o limite permitido (ANTONOPOULOS; WOOD, 2018). Caso o cálculo de *gas* utilizado na transação exceda a limitação imposta, todas as mudanças são revertidas ao estado inicial em que ocorreu a chamada, exceto o pagamento do imposto *fee* que fica retido e, se a transação ainda tiver uma porção de *gas* restante, ele é reembolsado ao remetente (BUTERIN, 2014). As transações prioritárias são as que têm um *GASPRICE* maior (RIBEIRO; MENDIZABAL, 2021).

# **Tokens e NFTs**

# Conceito

NFTs, sigla para *Non-Fungible Tokens*, são ativos digitais criados através de contratos inteligentes e armazenados em uma *blockchain* (MORINGIELLO; ODINET, 2022). Segundo Kugler (2021), NFTs são *tokens* criptográficos criados na plataforma Ethereum e vendidos, assim como na plataforma Bitcoin mas com uma diferença, a criptomoeda bitcoin é fungível, isto é, pode ser trocada por outros ativos fungíveis tais como a criptomoeda ethereum ou outras, bem como moedas fiduciárias como real ou dólar. Já *tokens* não fungíveis não podem ser trocados por outros *tokens* de mesma natureza. Em outras palavras, *tokens* fungíveis podem ser trocados por outros *tokens* fungíveis, enquanto NFTs não podem ser trocadas por outras, apenas compradas.

A documentação da plataforma Ethereum (2022), define *token* como sendo uma representação virtual de qualquer coisa dentro da mesma, sendo alguns exemplos:

* Pontos de reputação em plataformas *online*;
* Habilidades de um personagem em um jogo;
* Bilhetes de loteria;
* Ativos financeiros como por exemplo as ações de uma empresa;
* Criptomoedas.

Ainda segundo a documentação da plataforma Ethereum (2022), os *tokens* são tratados em diferentes padrões, sendo eles:

* ERC-20;
* ERC-721;
* ERC-777;
* ERC-1155;
* ERC-4626.

# Padrões de Token

Segundo a documentação da Ethereum (2020), os padrões são definidos como:

* ERC-20: Padrão utilizado para determinar *tokens* únicos, porém com propriedades iguais, servindo como base para implementação de criptomoedas onde o valor é sempre o mesmo;
* ERC-721: Padrão utilizado para NFTs, possibilitando a criação de *tokens* únicos, mas com valores diferentes dentro de um contrato inteligente, utilizando como chaves primárias o endereço do contrato e o id do *token*;
* ERC-777: É um modelo aprimorado do ERC-20, possibilitando chamadas de funções sempre que um *token* é enviado ou recebido pelo contrato. Como por exemplo, a transferência de um *token* ERC-777 é simplificada a apenas uma chamada de contrato, enquanto no padrão ERC-20 são necessárias duas etapas: aprovação, para confirmar a validade do *token* e transferência;
* ERC-1155: Junta ambos os padrões ERC-20 e ERC-721 possibilitando que um *token* único realize as mesmas funções presentes nos dois modelos ao mesmo tempo;
* ERC-4626: Este padrão disponibiliza uma série de funções permitindo que *tokens* ERC-20 sejam utilizados como fundos de investimento, dando a possibilidade de criação de aplicações de finança descentralizada com depósito, saque e leitura de saldo.

# **Utilização de blockchain em cartório brasileiro**

Segundo o site oficial do Colégio Notarial do Brasil em maio de 2020 o Colégio Notarial do Brasil (CNB) criou a Notarchain, uma *blockchain* focada em atos notariais eletrônicos utilizando como argumento a transparência e segurança gerada pela tecnologia *blockchain*. Segundo a presidente do Colégio Notarial do Brasil Giselle Oliveira de Barros, a Notarchain utiliza o modelo privado de *blockchain*, onde diferente de *blockchains* públicas como a Bitcoin, os participantes da rede devem ser autorizados a entrar na mesma (NOTARIADO, 2021).

Conforme citado por Giselle Oliveira de Barros, um exemplo de uso desta plataforma é reconhecimento de firma totalmente *online* realizado através do módulo e-Notariado, presente no site do CNB. Desta forma, esta dinâmica burocrática que convencionalmente necessita da presença física entre um comprador e um vendedor para assinar o ato, é simplificada através de toda a segurança e transparência proporcionada por uma *blockchain* (NOTARIADO, 2021).

1. **METODOLOGIA**

A metodologia tem como objetivo descrever técnicas de pesquisa utilizadas para coletar e processar informações a fim de resolver um problema. Desta forma, a metodologia diz respeito à aplicação de técnicas utilizadas para comprovar a construção de determinado argumento (PROVDANOV; FREITAS, 2013). Durante a formação acadêmica, novas habilidades são desenvolvidas de forma natural através de informações obtidas tanto em sala de aula quanto em pesquisas, então a metodologia científica trata de estabelecer padrões a serem seguidos para que se possa filtrar as ideias e adquirir conhecimento (ESTRELA, 2018).

O primeiro passo para a realização deste trabalho foi buscar artigos científicos relevantes para o contexto. O artigo escrito por Nakamoto (2008) serve como uma excelente base para o entendimento do assunto e acerca disso foi possível obter o devido direcionamento do que se deve pesquisar. Após conseguir fontes de pesquisa confiáveis, foi necessário realizar uma leitura buscando os principais pontos a serem desenvolvidos, como por exemplo *blockchain*, contratos inteligentes, *Bitcoin*, mineração, protocolos de consenso e *Ethereum*. Após ter um embasamento teórico inicial, os parágrafos começaram a ser redigidos, argumentando entre os autores escolhidos e os utilizando para complementar a fala um do outro.

Após completar o embasamento teórico, foram apresentados estudos de uso de contratos inteligentes e em seguida começou-se a produção do protótipo utilizando a plataforma *Ethereum* em conjunto com a linguagem de programação *Solidity*.

Por fim foram realizados testes utilizando o protótipo, demonstrando suas funcionalidades e após isto procedeu-se com a análise dos resultados dos testes

# FORMULAÇÃO DA QUESTÃO DE PESQUISA

Para realização da pesquisa, são levantadas algumas questões que têm como objetivo buscar conteúdo pertinentes ao tema. Desta forma, são feitas algumas perguntas como: O que é *blockchain*? Quando surgiu esta tecnologia? Como funciona uma *blockchain*? Quais são os componentes de uma blockchain? O que são protocolos de consenso? O que são funções *hash*? O que são contratos inteligentes e para que servem? Quais as linguagens de programação para contratos inteligentes? O que é a plataforma Ethereum? O que é solidity e como utilizá-la para implementar um contrato inteligente?

* 1. ESTRATÉGIAS PARA BUSCA E SELEÇÃO DE ESTUDOS PRIMÁRIOS
     1. **Palavras-chave**

Palavras-chave são termos utilizados para facilitar a busca sobre os assuntos tratados. Sendo assim, são definidas as seguintes palavras-chave: *Blockchain*;criptografia; protocolos de consenso; funções *hash;* bitcoin; criptomoedas; *smart contracts* / contratos inteligentes; ethereum, solidity.

* + 1. **Ambientes de buscas e bases de artigo**

Foram utilizadas as seguintes bases científicas na busca por conteúdo relevante para a pesquisa: *Institute of Electrical and Electronico Engineers* (IEEE); *International Journal of Information Technology* (IJIT); *International Journal of Network Security* (IJNS); Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC); *Ledger Journal*, Sociedade Brasileira de Computação; *International Journal of Computer Science Issues* (IJCSI); *Business and Information Systems Engineering*; Revista Em Tempo; Revista Interface Tecnológica e Google Acadêmico.

# **Idioma**

Este trabalho utilizou como base dois idiomas, sendo eles o português pois a pesquisa é feita em território brasileiro e inglês pelo fato de grande parte do conteúdo relevante pesquisado ser apresentado na língua inglesa.

* + 1. **Data de publicação**

Últimos 10 anos, considerando a evolução acelerada da tecnologia e estudos publicados, para obter uma quantidade razoável de conteúdo atualizado e de qualidade, com exceção de clássicos necessários para o entendimento do assunto.

* + 1. **Strings de busca**

*Strings* de busca são utilizadas para pesquisar as referências de forma filtrada para a produção do trabalho, sendo elas:

* *Blockchain* AND Criptografia;
* *Blockchain* AND Protocolos de consenso;
* *Blockchain* AND Criptomoedas;
* *Blockchain* AND Bitcoin;
* *Blockchain* AND Funções *hash*;
* *Blockchain* AND Contratos inteligentes;
* Contratos inteligentes AND Ethereum;
* Contratos inteligentes AND Solidity;
* Protocolos de consenso AND Bitcoin;
* Protocolos de consenso AND Ethereum;
  + 1. **Critérios de inclusão**

Critérios de exclusão, assim como *strings* de busca servem para filtrar o conteúdo com a finalidade de buscar informações de qualidade. Neste caso, se tem os seguintes critérios para este trabalho: Estudos sobre o funcionamento de uma *blockchain*; Estudos sobre criptomoedas; Artigos que abordem o funcionamento de uma *blockchain*; Conteúdos sobre contratos inteligentes; documentações das ferramentas utilizadas para criação do protótipo e *date range* de no máximo 10 anos com exceção de alguns clássicos necessários para o entendimento do assunto.

* + 1. **Critérios de exclusão**

Também são utilizados como filtro, porém com a finalidade de excluir os conteúdos não relevantes para a pesquisa. Para este trabalho são definidos os seguintes critérios de exclusão: Artigos que não correspondam com o foco da pesquisa; Artigos datados de mais de 10 anos; Artigos que não estejam em português ou inglês.

* + 1. **Metodologia apresentada em figura**

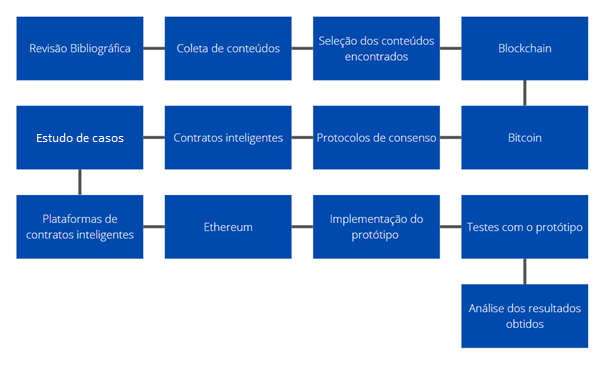


Figura 1: Fluxograma da metodologia de pesquisa. Fonte: (DO AUTOR)

Este trabalho foi dividido em algumas etapas necessárias para o desenvolvimento. Começando pela revisão bibliográfica, onde são definidas todas as regras para buscar conteúdos atualizados e de qualidade. Em seguida começa o processo de coletar fontes científicas como por exemplo artigos e livros. Após a coleta, é feito uma análise a respeito da procedência, relevância e qualidade do que foi obtido. Então o trabalho começa a ser produzido utilizando o que sobrou da análise, seguindo pelos seguintes tópicos:

* *Blockchain*;
* Protocolos de consenso;
* Contratos;
* Estudo de casos;
* Plataformas de contratos inteligentes;
* Ethereum;
* *Tokens* e NFTs.

Após concluir a pesquisa de embasamento teórico, são demonstrados casos de uso de contratos inteligentes. Posteriormente inicia-se a implementação do protótipo de votação e, ao concluir, realizam-se os testes. Por fim é demonstrado uma análise comparativa de resultados obtidos ao longo da pesquisa.

# **Introdução ao protótipo**

O protótipo utiliza a posse de bens imobiliários como modelo para a implementação. O padrão de *token* escolhido é o ERC-721 tendo em vista que é necessário transformar um imóvel real em um *token* único capaz de ser transferido de pessoa para pessoa sem intermédio. Para o desenvolvimento, assume-se que os dados informados no formulário de cadastro de um imóvel são válidos a fim de manter o foco no funcionamento do contrato inteligente na transferência da propriedade de uma pessoa para outra.

# **PROTÓTIPO**

# ESTRUTURA

# **Ideia proposta**

A ideia a ser implementada é uma aplicação *web* que utilize contratos inteligentes para registrar propriedades em uma *blockchain* a fim de descentralizar o processo burocrático de transferência de posse de bens. Dentro dessa aplicação é possível registrar, consultar e transferir imóveis.

# **Ferramentas**

Existem diversas ferramentas fundamentais para o desenvolvimento do protótipo que incluem linguagens de programação, *frameworks*, *plugins*, editor de texto e bibliotecas de códigos.

A ferramenta fundamental para a implementação da aplicação é o Visual Studio Code, um editor de texto que oferece suporte a diversas linguagens de programação, enquanto disponibiliza uma vasta gama de extensões para facilitar a tarefa de programar.

As linguagens de programação escolhidas para implementar o modelo são o TypeScript, JavaScript e Solidity. Ambas são muito próximas em sua sintaxe. JavaScript foi escolhido pois é uma linguagem de programação nativa para navegadores facilitando o desenvolvimento da aplicação *web* (MOZILLA, 2022). TypeScript utiliza JavaScript como base, porém com algumas mudanças, sendo a mais notável delas a utilização de tipos de dados para declarar variáveis e constantes a fim de facilitar a detecção de erros no código, enquanto que o JavaScript não utiliza tipagem (TYPESCRIPTLANG, 2022). Solidity é a linguagem de programação nativa para contratos inteligentes na rede Ethereum (ETHEREUM, 2022).

O protótipo é dividido em três partes, sendo elas: *Client* (interface); *Server* (servidor); Contrato Inteligente. Para construção do *client* foi utilizado React.js com TypeScript. Segundo a própria documentação do React.js, é uma biblioteca que usa como base a linguagem JavaScript para criação de interfaces. O *server* utiliza NodeJS como base para seu desenvolvimento, este que segundo sua própria documentação, é um ambiente de tempo de execução de código aberto e plataforma cruzada agregando a linguagem de programação JavaScript para criação das aplicações. O contrato inteligente utiliza a linguagem de programação Solidity para implementação, além também do *framework* Truffle para integração com a *blockchain*.

Segundo a documentação do Truffle, este é um conjunto de ferramentas utilizado para o desenvolvimento e integração de contratos inteligentes com uma *blockchain*, sendo utilizado para compilar e migrar os contratos para dentro da mesma. Será utilizado Ganache para realização dos testes com a aplicação, este que é uma *blockchain* local disponibilizado pelo Truffle, que fornece várias ferramentas de desenvolvimento, tais como *logs*, controle de mineração e exploração dos blocos e transações.

Para armazenar os dados dos *tokens* gerados pelo contrato inteligente, será utilizado Pinata, que é um serviço de gerenciamento de NFTs que possibilita a hospedagem, armazenamento e compartilhamento de arquivos de uma *blockchain* (PINATA, 2022).

# DESENVOLVIMENTO

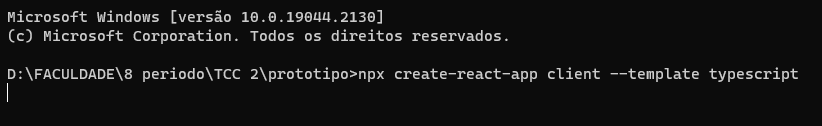
O primeiro passo é criar o *client*. Para isso utiliza-se o seguinte comando e sintaxe: npx create-react-app “nome da aplicação” –template typescript. A figura 2 demonstra o código escrito no *prompt* de comando.

Figura 2: Comando para criação de um projeto React.js. Fonte: (DO AUTOR, 2022).

Como o foco da pesquisa são os contratos inteligentes, não há necessidade de demonstrar toda a estrutura de código, porém ao final da implementação da interface gráfica, a estrutura de pastas é apresentada conforme figura 3.

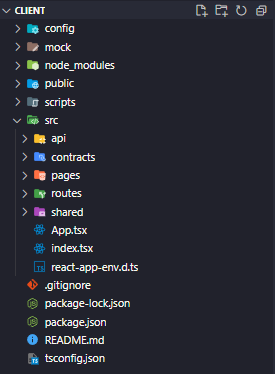


Figura 3: Estrutura de pastas de um projeto React.js. Fonte: (DO AUTOR, 2022).

Explicando de forma rápida a função de cada pasta, em “api” fica a configuração da biblioteca Axios utilizada para realizar *requests* HTTP para o *server*. Em “*contracts*” estão armazenadas as instâncias geradas pela migração dos contratos através do Truffle, para utilizá-los no *client*. Na pasta “*pages*” estão os códigos das telas da aplicação. Em “*routes*” é onde fica o componente utilizado para criar as rotas que redirecionam para as páginas da aplicação. E por fim “*shared*”é a pasta responsável por guardar todos os códigos que são compartilhados dentro da aplicação, incluindo componentes, contextos, *controllers*, váriaveis de ambiente, *hooks*, interfaces de implementação, *layouts, models*, temas e utilitários.

O segundo passo é criar o *server*, utilizando o comando “npm init”. Vale ressaltar que tanto o comando para criação do *server* quanto para a criação do *client* são disponibilizados pelo NodeJS, sendo então necessário tê-lo instalado previamente.

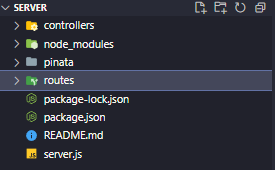
A figura 4 apresenta a seguinte estrutura de pasta após o desenvolvimento do *server*:

Figura 4: Estrutura de pastas de um projeto NodeJS. Fonte: (DO AUTOR, 2022).

O *server* é composto de três componentes principais para receber requisições do *client* e devolver uma resposta ao mesmo, sendo eles: *server*; *routes* e *controllers*. As requisições chegam para o *server* através do arquivo “server.js”, que por sua vez passa uma *request* e *response* como parâmetros para o *router* através da função “router(req, res)”, presente no arquivo “router.js”, dentro da pasta “*routes*”. As figuras 5 e 6 demonstram o código presente no arquivos “server.js” e “router.js”, respectivamente.

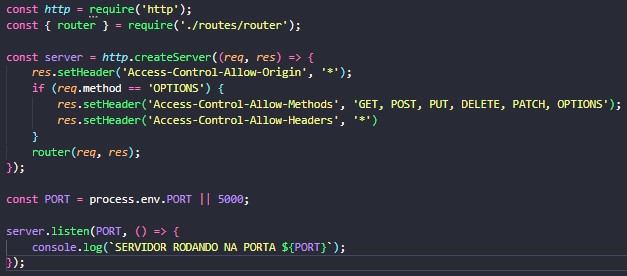
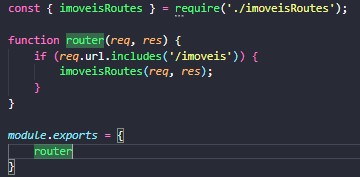
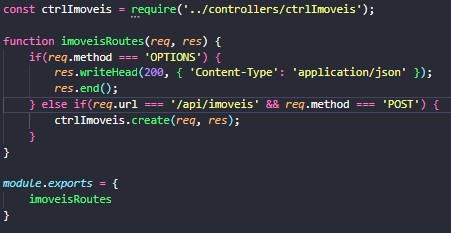
Figura 5: Código fonte do arquivo “server.js”. Fonte: (DO AUTOR, 2022).

Figura 6: Código fonte do arquivo “router.js”. Fonte: (DO AUTOR, 2022).

A função “router(req, res)” verifica qual é o objeto alvo da rota através da função “includes()”, que recebe uma string como parâmetro e verifica se o mesmo está presente na variável que chamou o método e em caso positivo, invoca a função que redireciona para as rotas solicitadas, neste caso sendo “imoveisRoutes(req, res)”. O arquivo “imoveisRoutes.js”, tem como tarefa verificar o método solicitado pela *request* do *client*, bem como a url de destino e redirecionar para o método adequado presente na *controller*. A figura 7 demonstra o código presente em “imoveisRoutes.js”.

Figura 7: Código fonte do arquivo “imoveisRoutes.js”. Fonte: (DO AUTOR, 2022).

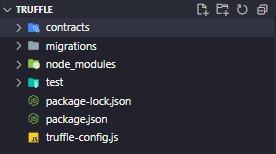
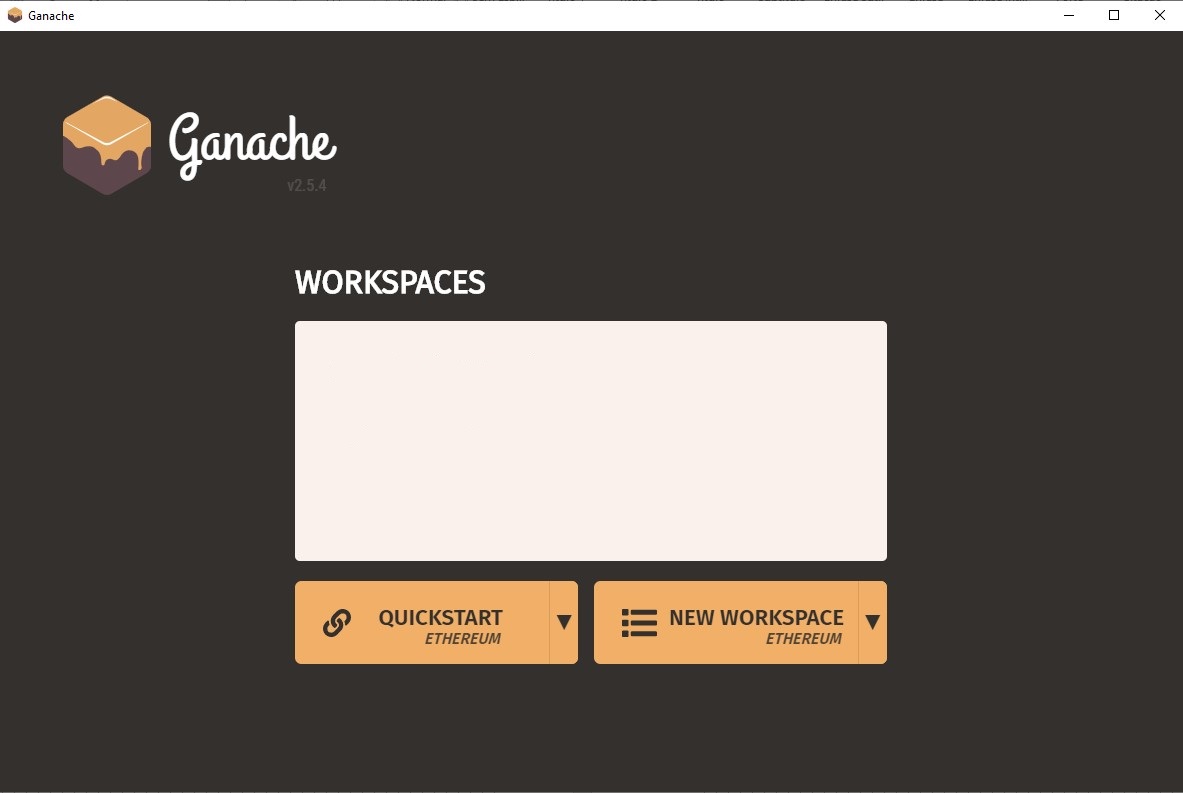
O terceiro passo é criar uma aplicação Truffle, utilizando o comando “truffle init”. Após realizar executar o mesmo, a seguinte estrutura de pastas é gerada, conforme demonstra a figura 8.

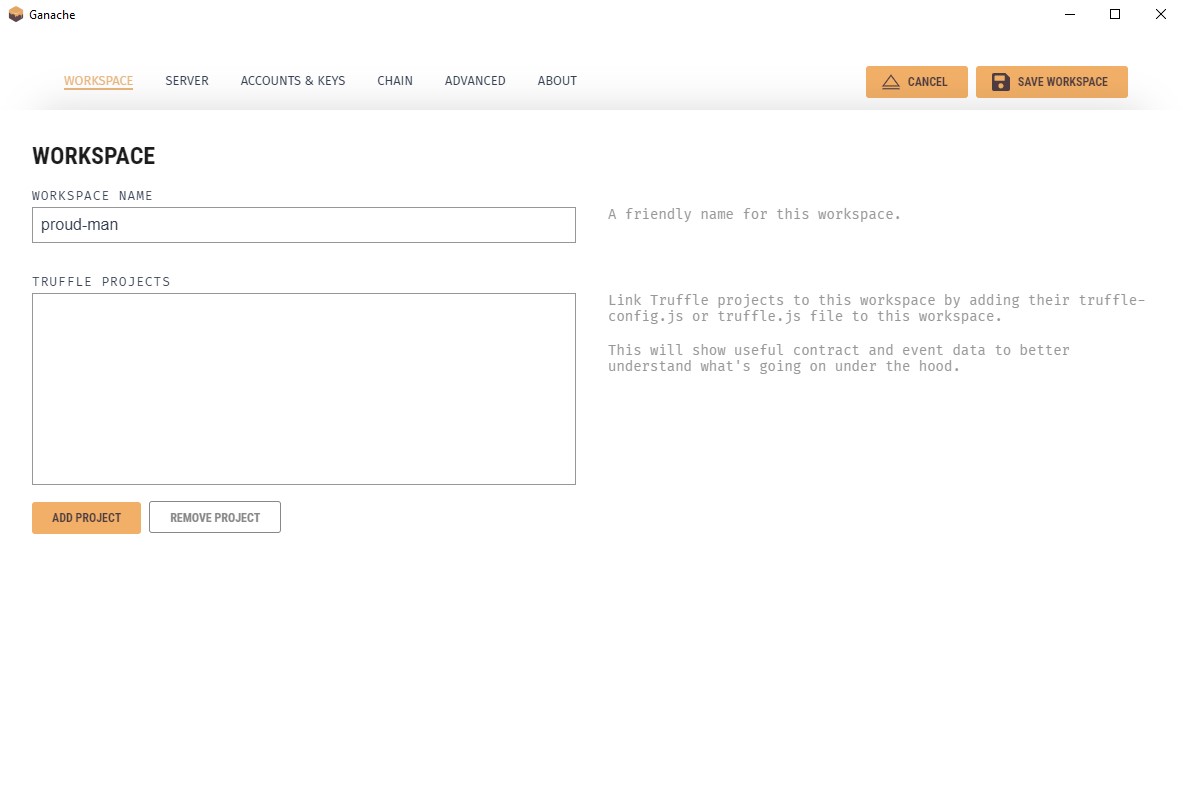
Figura 8: Estrutura de pastas de um projeto Truffle. Fonte: (DO AUTOR, 2022).

A pasta “*contracts*” armazena os contratos inteligentes desenvolvidos. Na pasta “*migrations*” é onde ficam os arquivos escritos em JavaScript que têm como função realizar a migração dos contratos para a *blockchain*. Em “test” é onde são implementados os testes unitários e por fim o arquivo “truffle-config.js” guarda as configurações necessárias para conectar a aplicação Truffle com a *blockchain*.

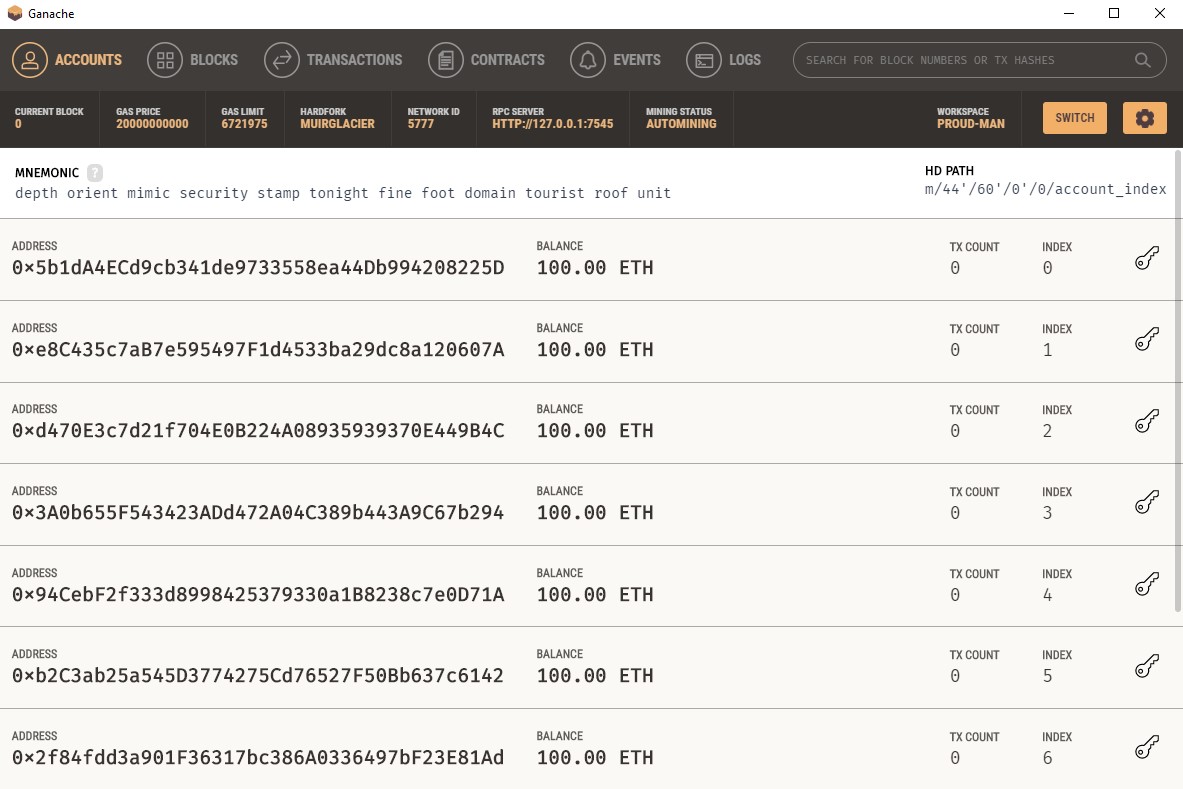
Antes de mexer nas configurações da aplicação, é necessário criar uma *blockchain* local. Para isso é utilizado o Ganache que deve ser baixado através do site do Truffle e instalado previamente. Após inicializar o Ganache, o usuário é apresentado para a seguinte tela, conforme mostra a figura 9:

Figura 9: Tela inicial do Ganache. Fonte: (DO AUTOR, 2022).

Em seguida, basta clicar no botão “*NEW WORKSPACE*”e uma nova interface é mostrada ao usuário, conforme apresentado na figura 10.

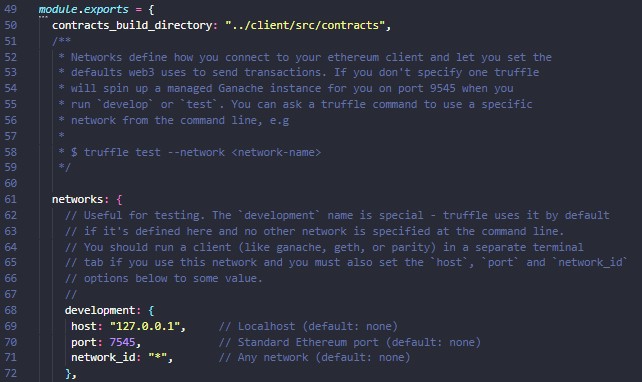
Figura 10: Configuração inicial do *workspace* do Ganache. Fonte: (DO AUTOR, 2022).

Na primeira caixa de texto, abaixo de “*WORKSPACE NAME*”, deve-se inserir o nome do projeto e, após isso, é necessário clicar no botão “*ADD PROJECT*” e, na janela que abrir, apontar o caminho do arquivo “truffle-config.js”, gerado quando foi criado a aplicação Truffle. Após concluir esses passos, basta clicar em “*SAVE WORKSPACE*” e a próxima tela é apresentada ao usuário conforme demonstra a figura 11.

Figura 11: Tela principal do Ganache. Fonte: (DO AUTOR, 2022).

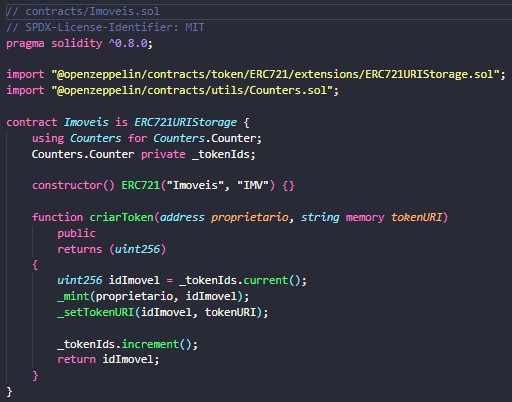
Nesta interface, são apresentadas diversas informações, tais como as contas que fazem parte da *blockchain* local, os blocos minerados, as transações realizadas ma rede, os contratos inteligentes migrados, os eventos e os *logs*, além também de algumas informações necessárias para configurar o ambiente do Truffle. Para realizar com sucesso a integração do Truffle com Ganache, é utilizado o IP disponibilizado na sexta coluna abaixo dos botões principais, em “RPC *Server*”. Neste caso o endereço é “127.0.0.1:7545”.

Para finalizar a configuração, basta abrir o arquivo “truffle-config.js” e alterar algumas linhas. A primeira alteração a se fazer é, logo após a linha 49 onde se abre um bloco de código abaixo de “module.exports = {”, inserir a seguinte linha: “contracts\_build\_directory: “../client/src/contracts”. Desta forma, todos os contratos compilados e migrados têm suas instâncias armazenadas dentro do módulo *client* e pode ser utilizado fora da aplicação Truffle. Em seguida selecionar as linhas 68 até 72 e retirar o comentário através do comando no teclado “CTRL+K+U” e, após isso, preencher a variável *host* e *port* de acordo com o endereço gerado pelo Ganache. A figura 12 demonstra como fica o resultado do processo desses últimos passos.

Figura 12: Código fonte do arquivo “truffle-config.js”. Fonte: (DO AUTOR, 2022).

Após finalizada a configuração inicial do projeto, a próxima tarefa é implementar o contrato inteligente. Para isto será utilizada a biblioteca OpenZeppelin, que disponibiliza uma série de padrões de *tokens* prontos para utilização, incluindo o ERC-721, que será utilizado para o desenvolvimento do contrato. Para instalar o pacote, é utilizado o comando via *prompt* “npm install @openzeppelin/contracts”, conforme demonstra a documentação da OpenZeppelin.

Depois de instalada, basta criar um novo arquivo dentro das pasta “*contracts*” chamado de “Imoveis.sol”. A extensão “.sol” é utilizada pera criar arquivos que utilizem a linguagem de programação Solidity. A documentação da OpenZeppelin disponibiliza o código do contrato inteligente, sendo necessário algumas alterações de nomenclatura para fazer sentido ao contexto do protótipo. A figura 13 demonstra o contrato inteligente implementado.

Figura 13: Código fonte do contrato inteligente em Solidity. Fonte: Adaptado de (OPENZEPPELIN, 2022).

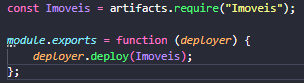
Em seguida, é necessário criar o arquivo responsável por migrar o contrato para a *blockchain*. Deve ser criado dentro da pasta “*migrations*”. O arquivo é nomeado como “2\_imoveis\_migration.js” e apresenta o seguinte código, de acordo com a figura 14.

Figura 14: Código fonte do arquivo “2\_imoveis\_migration.js”. Fonte: (DO AUTOR, 2022).

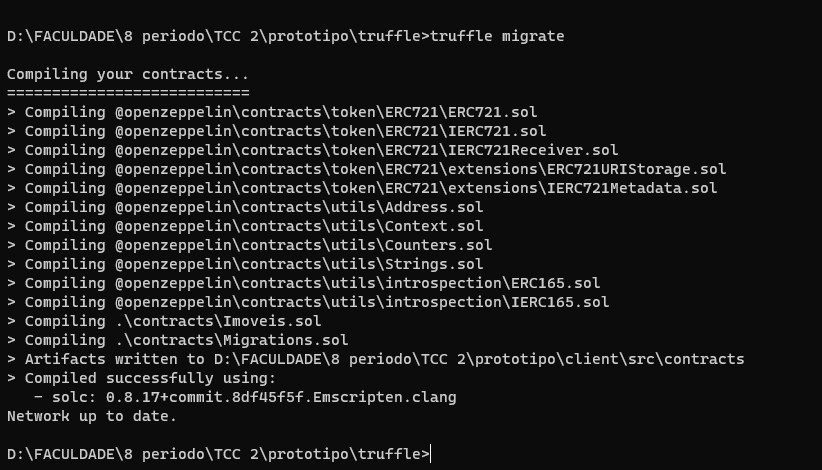
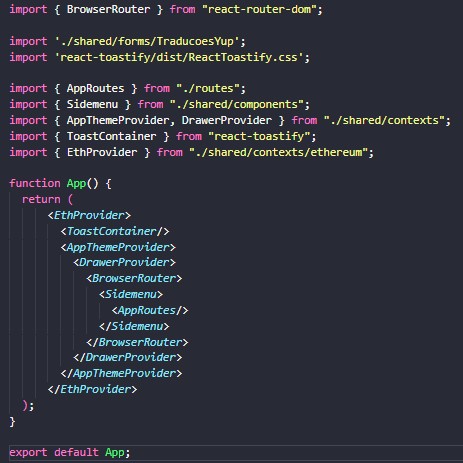
Para realizar o processo de migração dos contratos, basta abrir um *prompt* de comando, apontar para a pasta do Truffle e executar o seguinte código: “truffle migrate”. Este comando gera o seguinte resultado apresentado pela figura 15:

Figura 15: Fonte: Captura de tela com os resultados da migração dos contratos. Fonte: (DO AUTOR, 2022).

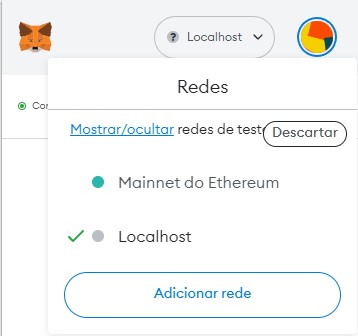
Depois de migrar o contrato para a *blockchain*, é necessário apontar a instância do mesmo dentro do contexto de Ethereum, presente na pasta “*shared*/*contexts*/ethereum/EthProvider.jsx”. A figura 16 demonstra o código final para esse arquivo.

Figura 16: Código fonte do contexto Ethereum. Fonte: (DO AUTOR, 2022).

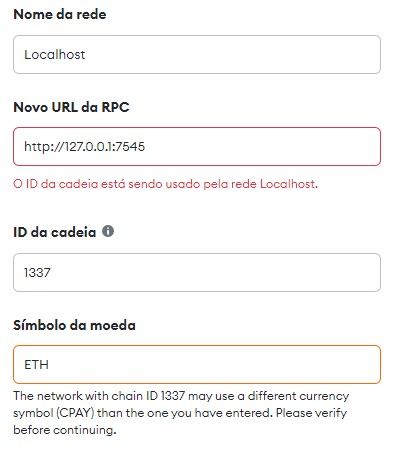
Este arquivo é importado e declarado dentro do arquivo “App.tsx”, na pasta raíz do módulo *client*, fazendo com que o contexto possa ser compartilhado em qualquer parte do código da aplicação. A figura 17 apresenta a estrutura do código em “App.tsx”.

Figura 17: Código fonte do arquivo “App.tsx”. Fonte: (DO AUTOR, 2022).

Antes de inicializar o protótipo, é necessário configurar o *plugin* MetaMask. Este *plugin* é o responsável por conectar o navegador de internet com o código da aplicação e com a *blockchain*. Basta instalar o *plugin* no navegador e criar uma conta. Após isso, abrir as extensões do navegador, selecionar MetaMask e logar com a conta criada. Depois de conectar,precisa-se configurar a rede local do Ganache, clicando no botão ao lado do ícone da raposa e em seguida “Adicionar rede”, conforme demonstra a figura 18.

Figura 18: Adicionar rede na MetaMask. Fonte: (DO AUTOR, 2022).

O formulário que aparecer na tela deve ser preenchido com os dados referentes a rede Ganache, criada previamente, conforme apresentado na figura 19.

Figura 19: Configurações de nova rede na MetaMask. Fonte: (AUTOR, 2022).

Em sequência, é necessário importar para a MetaMask uma conta de testes disponibilizada pelo Ganache. Para isso, basta selecionar qualquer conta, clicar em “*show keys*”, e copiar a chave privada da conta, conforme demonstra a figura 20.



Figura 20: Como encontrar a chave privada da conta utilizada na aplicação. Fonte: (DO AUTOR, 2022)

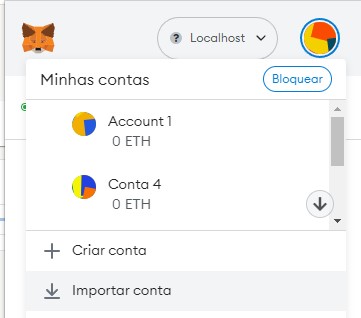
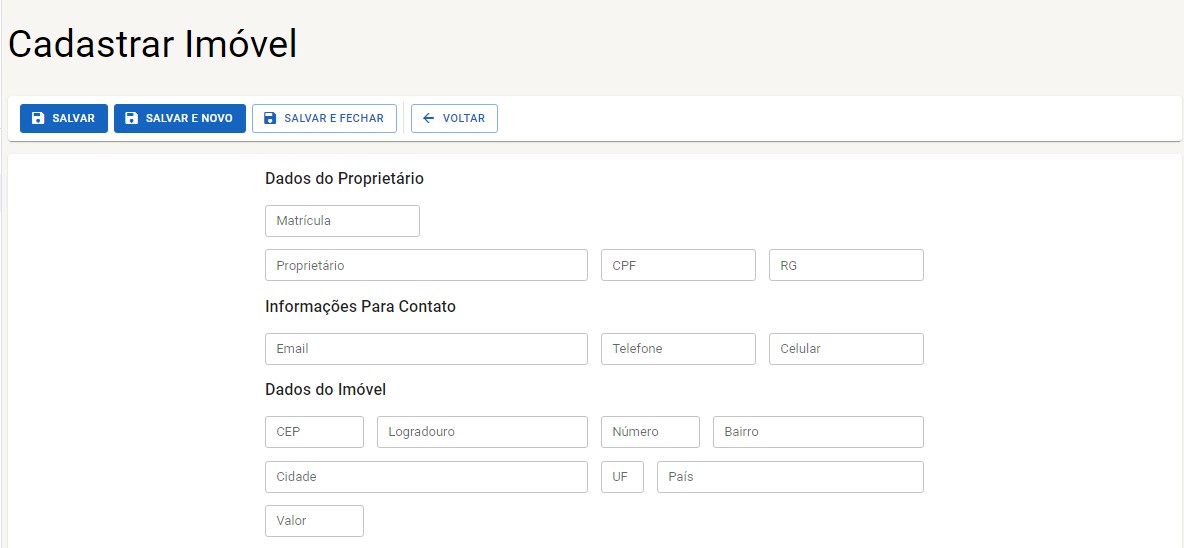
Depois de copiar a chave privada, o usuário deve abrir a MetaMask, clicar no botão circular ao lado do botão de redes e clicar em importar conta, da forma como exemplifica a figura 21.

Figura 21: Importar conta na MetaMask. Fonte: (DO AUTOR, 2022)

Na tela que aparecer, o usuário deve colar a chave privada e clicar em importar.

Após completar todas as etapas, o protótipo está pronto para criar *tokens*. Para inicializar a aplicação, basta abrir um *prompt* de comando apontando para a pasta *client* e executar o seguinte comando: “npm start”. Em seguida, abrir outro *prompt* de comando apontando para a pasta server e rodar o comando “node server”.

Logo quando a aplicação inicializar, o MetaMask vai pedir para autenticar, basta entrar com o *login* e senha. Em seguida clicar na aba “Imóveis”, na barra de navegação no lado esquerdo da tela e clicar no botão “NOVO”. O usuário será redirecionado para a tela de cadastro de imóvel, demonstrada na figura 22.

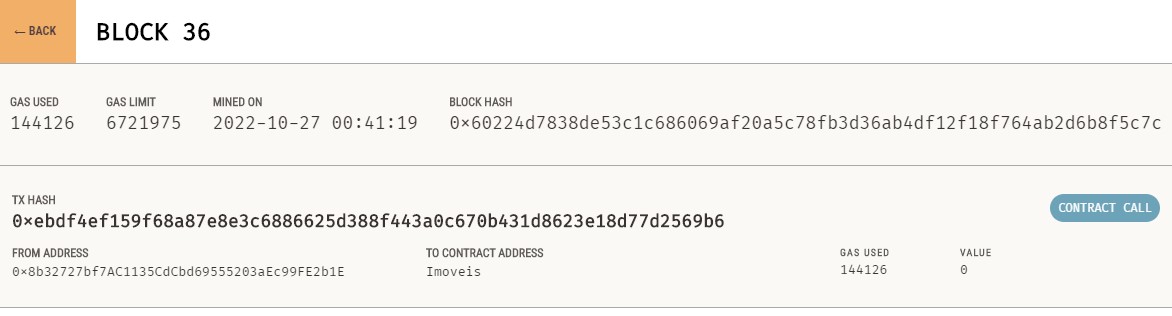
Figura 22: Captura de tela do formulário de cadastro de imóvel. Fonte: (DO AUTOR, 2022).

Após preencher os dados, deve clicar no botão “SALVAR” e então a MetaMask vai solicitar a confirmação da transação. Após confirmar, o *token* é criado e armazenado na *blockchain*. A figura 23 mostra a notificação de confirmação da transação pela MetaMask bem como o valor gasto em ETH pela transação.



Figura 23: Confirmar transação. Fonte: (DO AUTOR, 2022).

Ao confirmar a transação, é possível consultar o bloco minerado no Ganache, onde são apresentadas algumas informações, tais como horário da criação, endereço que chamou o contrato e o nome do contrato chamado, como demonstra a figura 24.

Figura 24: Bloco gerado pela transação. Fonte: (DO AUTOR, 2022).

Utilizando a função “tokenURI(tokenId)”, é possível recuperar o tokenURI que aponta para um JSON onde estão armazenados os dados do *token* e, utilizando a biblioteca Axios para realizar a requisição, é possível verificar os metadados do *token*, conforme exemplo na figura 25.

Figura 25: Informações do imóvel cadastrado na blockchain. Fonte: (DO AUTOR, 2022).

**REFERÊNCIAS BIBLIOGRÁFICAS**

ABIJAUDE, J. *et al.* Blockchain, Contratos Inteligentes, SistemasWeb: Teoria e Prática. **Tópicos Especiais em Sistemas de Informação:** Minicursos SBSI 2021. p.1–30, 2021.

ANTONOPOULOS, A. M.; WOOD, G. **Mastering Ethereum :** building smart contracts and DApps. 1. ed. Califórnia, 2018.

BUTERIN, V. **Ethereum White Paper:** A Next generation smart contract and decentralized application platform. p. 1–36, jan. 2014. Disponível em: https://blockchainlab.com/pdf/Ethereum\_white\_paper-a\_next\_generation\_smart\_contract\_and\_decentralized\_application\_platform-vitalik-buterin.pdf. Acesso em: 9/3/2022.

CARVALHO, C. A. de; ÁVILA, L. V. A Tecnologia Blockchain Aplicada aos Contratos Inteligentes. **Revista Em tempo**, v. 18, n. 01, p. 156–176, 2019. Disponível em: https://revista.univem.edu.br/emtempo/article/view/3210.

DIVINO, S. B. S. Smart Contracts: Conceitos, Limitações, Aplicabilidades e Desafios. **Revista Jurídica Luso-Brasileira**. p. 2771–2808, nov. 2020. Disponível em: https://www.cidp.pt/revistas/rjlb/2018/6/2018\_06\_2771\_2808.pdf. Acesso em: 23/5/2022.

ESTRELA, C. **Metodologia Científica**. 3. ed. São Paulo: Artes Médicas, 2018.

GOLOSOVA, J.; ROMANOVS, A. The advantages and disadvantages of the blockchain technology. **2018 IEEE 6th Workshop on Advances in Information, Electronic and Electrical Engineering**. p. 1-6, dez. 2018.

GREVE, F. *et al.* Blockchain e a Revolução do Consenso sob Demanda. **Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC)**, v. 1, n. 1, p. 52, maio. 2018. Disponível em: http://143.54.25.88/index.php/sbrcminicursos/article/view/1770. Acesso em: 22/5/2022.

HEGEDŰS, P. Towards Analyzing the Complexity Landscape of Solidity Based Ethereum Smart Contracts. **2018 IEEE/ACM 1st International Workshop on Emerging Trends in Software Engineering for Blockchain (WETSEB)**, v. 7, n. 1, p. 6, jan. 2019.

KARATAŞ, E. Developing Ethereum Blockchain-Based Document Verification Smart Contract for Moodle Learning Management System. **International Journal of Informatics Technologies**, v. 11, n. 4, p. 399-406, out. 2018.

KING, S.; NADAL, S. PPCoin: Peer-to-Peer Crypto-Currency with Proof-of-Stake. 2012.

LIN, I. C.; LIAO, T. C. A Survey of Blockchain Security Issues and Challenges. **International Journal of Network Security**, v. 19, n. 5, p. 653–659, set. 2017.

LOUREIRO, J. M. G. **Implementação de funções de hash e cifras de chave pública baseadas em retículos**. Dissertação (Mestrado em Matemática e Computação) - Escola de Ciências, Universidade do Minho. Braga, out. 2012.

MIERS, C. *et al*. Análise de mecanismos para consenso distribuído aplicados a Blockchain. **XIX Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais - SBSeg 2019**. p. 91–139, 2019. Disponível em: https://sol.sbc.org.br/livros/index.php/sbc/catalog/download/85/374/638-1?inline=1. Acesso em: 16/5/2022.

NAKAMOTO, S. Bitcoin: A Peer-to-Peer Electronic Cash System. 2008. Disponível em: https://bitcoin.org/bitcoin.pdf. Acesso em: 15/2/2022.

NOFER, M. *et al*. Blockchain. **Business and Information Systems Engineering**, v. 59, n. 3, p. 183–187, mar. 2017.

PIERRO, M. Di. What Is the Blockchain. **Computing in Science and Engineering**, v. 19, n. 5, p. 92–95, 2017.

PRODANOV, C. C.; FREITAS, E. C. de. **Metodologia do trabalho científico:** métodos e técnicas da pesquisa e do trabalho acadêmico. 2. ed. Novo Hamburgo: Feevale, 2013.

REBELLO, G. A. F. *et al*. Segurança e Desempenho de Protocolos de Consenso Baseados em Prova para Corrente de Blocos. **Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais (SBSeg)**. Petrópolis, p. 341–354, 2020. SBC. Disponível em: https://sol.sbc.org.br/index.php/sbseg/article/view/19248. Acesso em: 17/5/2022.

REBELLO, G. A. F.; CAMILO, G.; SILVA, L.; *et al.* Correntes de Blocos: Algoritmos de Consenso e Implementação na Plataforma Hyperledger Fabric. **Jornada de Atualização em Informática 2019**, p. 93–148, jul. 2019.

RIBEIRO, L.; MENDIZABAL, O. **Introdução à Blockchain e Contratos Inteligentes :** Apostila para Iniciante. 2021.

SINGH, S.; SINGH, N. Blockchain: Future of financial and cyber security. **Proceedings of the 2016 2nd International Conference on Contemporary Computing and Informatics, IC3I 2016**, p. 463–467, dez. 2016.

SOBTI, R.; GANESAM, G. Cryptographic Hash functions: A review. **IJCSI International Journal of Computer Science Issues**, v. 9, n. 2, p. 461–479, mar. 2012. Disponível em: https://www.researchgate.net/publication/267422045.

TASCA, P.; TESSONE, C. J. A Taxonomy of Blockchain Technologies: Principles of Identification and Classification. **Ledger**, v. 4, fev. 2019. Disponível em: https://www.ledgerjournal.org/ojs/ledger/article/view/140. Acesso em: 22/5/2022.

LUCENA, A. U. De. Estudo de arquiteturas dos blockchain de *Bitcoin* e *Ethereum*. **IX Encontro de Alunos e Docentes do DCA/FEEC/UNICAMP (EADCA)**. Campinas, set. 2016.

AQUINO, L. G. De. **Teoria Geral dos Contratos**. Belo Horizonte: Editora Expert, 2021.

ROBERTO, C. G. **Direito Civil Brasileiro:** Contratos e Atos Unilaterais. 16o ed. São Paulo: Saraiva Educação, 2019.

ZOU, W. *et al*. Smart Contract Development: Challenges and Opportunities. **IEEE Transactions on Software Engineering**, v. 47, n. 10, p. 2084–2106, 2021. Institute of Electrical and Electronics Engineers Inc.

CHRISTIDIS, K.; DEVETSIKIOTIS, M. Blockchains and Smart Contracts for the Internet of Things. **IEEE Access**, v. 4, p. 2292–2303, 2016. Institute of Electrical and Electronics Engineers Inc.

GONÇALVES, C. R. **Direito civil brasileiro**: contratos e atos unilaterais. 16ed. São Paulo: Saraiva Educação, 2019.

AQUINO, L. G. De. **Teoria Geral dos Contratos**. Belo Horizonte: Editora Expert, 2021.

WANG, S. *et al*. An Overview of Smart Contract: Architecture, Applications, and Future Trends. **2018 IEEE Intelligent Vehicles Symposium (IV)**, p. 108–113, jun. 2018. Institute of Electrical and Electronico Engineers.

MOHANTA, B. K.; PANDA, S. S.; JENA, D. An Overview of Smart Contract and Use Cases in Blockchain Technology. **2018 9th International Conference on Computing, Communication and Networking Technologies (ICCCNT)**, p. 1–4, 2018. Institute of Electrical and Electronico Engineers.

ZHENG, Z. *et al*. An overview on smart contracts: Challenges, advances and platforms. **Future Generation Computer Systems**, v. 105, p. 475–491, 2020. Elsevier B.V.

VALENTA, M.; SANDNER, P. Comparison of Ethereum, Hyperledger Fabric and Corda. **FSBC Working Paper**, p. 8, jun. 2017.

KARATAŞ, E. Developing Ethereum Blockchain-Based Document Verification Smart Contract for Moodle Learning Management System. **Journal of Information Technologies**, v. 11, n. 4, p. 399–406, out. 2018.

MORINGIELLO, J. M.; ODINET, C. K. Blockchain Real Estate and NFTs. **SSRN**, p. 1–43, 2022.

KUGLER, L. Non-fungible tokens and the future of art. **Communications of the ACM**, v. 64, n. 9, p. 19–20, set. 2021.

Ethereum Development Documentation. Disponível em: <https://ethereum.org/pt/developers/docs/>. Acesso em: 20/10/2022.

Bloomberg destaca Reconhecimento de Firma feito pelo e-Notariado – **Colégio Notarial**. Disponível em: https://www.notariado.org.br/bloomberg-destaca-reconhecimento-de-firma-feito-pelo-e-notariado/. Acesso em: 20/10/2022.

Ethereum development documentation. Disponível em: https://ethereum.org/pt/developers/docs/. Acesso em: 30/10/2022.

TypeScript: JavaScript With Syntax For Types. Disponível em: https://www.typescriptlang.org/. Acesso em: 30/10/2022.

Truffle suit documentarion. Disponível em: https://trufflesuite.com/docs/. Acesso em: 30/10/2022.

Pinata docs. Disponível em: https://docs.pinata.cloud/. Acesso em: 30/10/2022.

OpenZeppelin docs. Disponível em: https://docs.openzeppelin.com/. Acesso em: 30/10/2022.