UNIVERSIDADE FEDERAL DO ABC
ENGENHARIA DE INFORMAÇÃO

FABÍOLA ALINE TOLENTINO

CONSTRUÇÃO DE UM BLOCKCHAIN E IMPLEMENTAÇÃO DE UM SMART CONTRACT.

Monografia

SANTO ANDRÉ – SP
2018

Fabíola Aline Tolentino

CONSTRUÇÃO DE UM BLOCKCHAIN E IMPLEMENTAÇÃO DE UM SMART CONTRACT.

Monografia.

Monografia apresentada ao curso de graduação da Universidade Federal do ABC como requisito parcial para obtenção do grau de Engenheiro de Informação.

Orientador: Prof. Dr. Nunzio Marco Torrisi

SANTO ANDRÉ – SP
2018

Fabíola Aline Tolentino

Fabíola Aline Tolentino
BLOCKCHAIN - IMPLEMENTAÇÃO DE UM SMART CONTRACT. / Fabíola Aline Tolentino – Santo André, SP 2018.
Orientador: Prof. Dr. Nunzio Marco Torrisi.

Trabalho de Graduação – Universidade Federal do ABC – UFABC, 2018.

1. Paravra-chave1. 2. Palavra-chave2. 3. Palavra-chave3. I. Nunzio Marco Torrisi. II. Universidade Federal do ABC. III. Centro de Engenharia, Modelagem e Ciências Sociais Aplicadas. IV. Blockchain-Implementação de um Smart Contract.

Fabíola Aline Tolentino

CONSTRUÇÃO DE UM BLOCKCHAIN E IMPLEMENTAÇÃO DE UM SMART CONTRACT.

Essa monografia foi julgada e aprovada para obtenção de grau no curso de Engenharia de Informação da Universidade Federal do ABC.

Santo André – SP, 27 de Novembro de 2018

_____

Prof. Dr. Amaury Kruel Budri
Coordenador do Curso

BANCA EXAMINADORA

_____ _____

Nunzio Marco Torrisi João Marcelo Borovina

_____

Márcio Katsumi Oikawa

DEDICATÓRIA

À Deus e minha família, pelo apoio e paciência.
À todos meus professores de graduação
da Universidade Federal do ABC.
Aos meus colegas de curso e estudo.

AGRADECIMENTOS

À Universidade Federal do ABC. Ao orientador Prof. Dr. Nunzio Marco Torrisi, pelo pronto auxílio e acompanhamento do projeto. Aos professores do curso de graduação de Engenharia de Informação. A todos que direta ou indiretamente contribuíram para a realização desta monografia.

RESUMO

Este trabalho de graduação estuda a tecnologia blockchain e seu uso para o desenvolvimento e implementação de contratos inteligentes. O crescente uso de dispositivos móveis e o aumento significativo no número de diferentes tipos de transações reforçam a importância em encontrar tecnologias alternativas para garantir transparência, segurança, integridade e eficiência nas negociações. O blockchain construído em Python permite que transações sejam verificadas e registradas automaticamente a partir da rede descentralizada distribuída, sem intervenção humana ou qualquer outra entidade centralizadora. A informação passa a ser transparente e imutável. O contrato inteligente por sua vez, foi codificado e implementado utilizando o blockchain Ethereum, o objetivo principal foi entender o funcionamento da tecnologia, podendo aplicar os conceitos aqui apreendidos para diversas outras áreas.

Palavras Chave: blockchain, transações, Contratos Inteligentes.

ABSTRACT

This graduation work studies the blockchain technology and its use for development and deployment of smart contracts. The increasing use of mobile devices and the growth of transactions reinforce the importance in finding alternative technologies to guarantee transparency, security, integrity and efficiency in negotiations. The blockchain builded in Python language allows transactions to be verified and recorded automatically in the decentralized and distributed network without human interventions or any other centralizing entity. The information becomes transparent and immutable. The Smart Contract was encoded and implemented using Ethereum blockchain and the main goal was to understand how the technology worked and how could be used. The knowledge learned here can be extended to several areas and diverse applications.

Keywords: blockchain, transactions, Smart Contracts.

LISTA DE ABREVIATURAS E SIGLAS

IOT	*Internet of Things*
NFC	*Near Field Communications*
CAGR	*Compound annual growth*
WSGI	*Web Server Gateway Interface*
UTXO	*Unspent Transaction Output*

LISTA DE FIGURAS

LISTA DE TABELAS

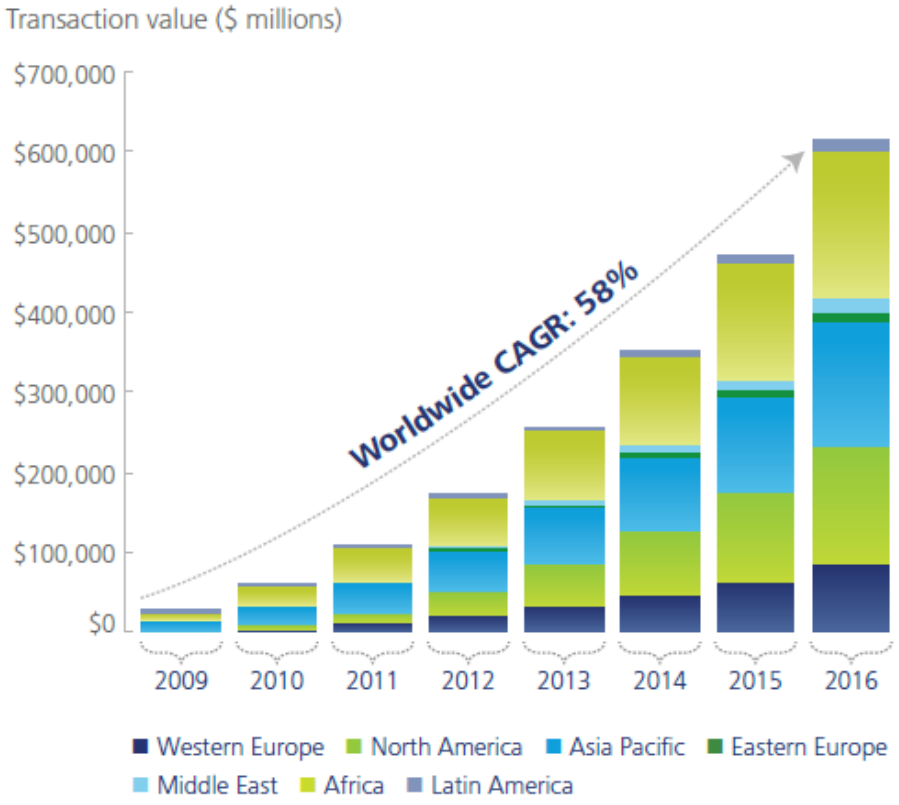
SUMÁRIO

1. Introdução

Devido aos avanços digitais e tecnológicos, o volume de transações ao longo da internet vem crescendo exponencialmente, tendo como principais impulsionadores o crescimento do comércio eletrônico, transações bancárias online, compras por aplicativo e aumento da mobilidade de pessoas ao redor do mundo. Industrias e diversos setores econômicos estão tendo que rever seus modelos de negócio.

A figura 1 retrata a expansão das transações via mobile para o mercado de consumidores globais por região de 2009 a 2016. [1]

Figura 1- Transações de pagamentos realizados via Mobile ao redor do Mundo de 2009 à 2016

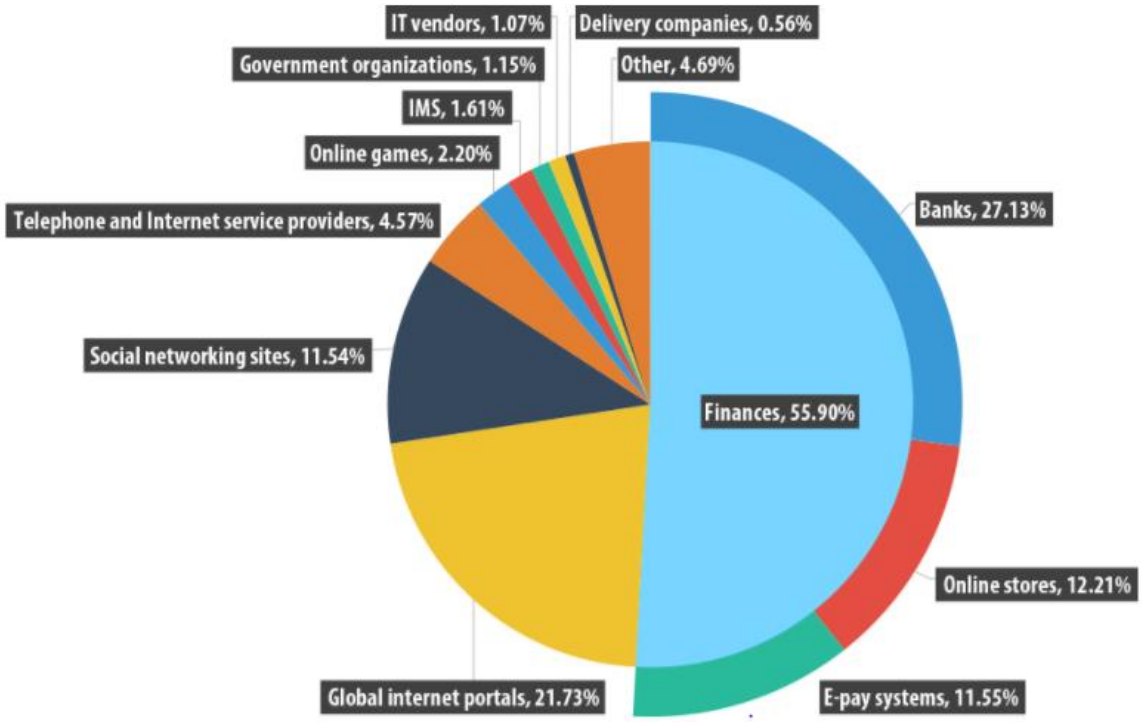


Source: Sandy Shen, *Forecast: Mobile payment, worldwide, 2009–2016*, Gartner, May 9, 2012; Deloitte analysis.

Ao longo de 6 anos o crescimento anual composto (CAGR) foi de 58% e, ao que tudo indica com temas como Indústria 4.0, Internet das Coisas e Cidades Inteligentes estima-se que em 2020 o número de objetos conectados chegará a 50 bilhões [2]. Como resultado, também crescem os níveis de complexidade, vulnerabilidade, eficiência, bem como as preocupações quanto ao custo destes sistema de transações.

Também cresce o número de ataques a esses diversos tipos de organização, conforme ilustra a figura 2. Tais organizações centralizam as informações de seus clientes em servidores vulneráveis a ataques. Bancos e instituições financeiras representam mais da metade dos ataques registrados no terceiro quadrimestre de 2016 de acordo com dados fornecidos Kaspersky Lab, empresa internacional de cibersegurança. [3].

Figura 2- Distribuição de organizações afetadas por ataques phising por categoria – Resultado do 3° trimestre de 2016.



Fonte: GUDKOVA, Darya; VERGELIS, Maria; DEMIDOVA, Nadezhda. Spam and phishing in Q3 2016. 2016. Disponível em: <https://securelist.com/spam-and-phishing-in-q3-2016/76570/>. Acesso em: 06 jul. 2018.

Este trabalho teve como objetivo entender os principais conceitos relacionados a tecnologia blockchain e utilizar tal conhecimento para simular uma rede distribuída blockchain. Além disso obteve-se conhecimentos relacionados a contratos inteligentes e implementou-se um sistema simples a fim de verificar o potencial uso do blockchain como uma das tecnologias capazes de equilibrar transparência, segurança, integridade e eficiência nas transações.

A seção dois deste trabalho, apresenta o estudo teórico realizado para entender o funcionamento da tecnologia do blockchain, através da aplicação dos conceitos aprendidos, a seção três mostrará os softwares utilizados para a construção do blockchain e o passo-a-passo do desenvolvimento realizado.

A seção quatro apresenta conceitos teóricos relacionados a contratos inteligentes e por fim, a seção cinco exibe detalhes da simulação criada a partir do blockchain pessoal da Ethereum, utilizando o Solidity, linguagem de programação voltada ao desenvolvimento de contratos inteligentes.

2. Blockchain

Moeda digital e pagamentos foram apenas a primeira aplicação do Blockchain. A tecnologia tem como propósito central rastrear qualquer tipo de transação direcionada a dois ou mais usuários, de uma maneira completamente descentralizada e distribuída globalmente. Portanto, a criptomoeda se tornou porta de entrada para o comércio de outros tipos de recursos. [4] A tabela 1 apresenta uma idéia das possíveis aplicações da tecnologia. [5]

Tabela 1- Aplicações do blockchain

Classe	**Exemplo**
Geral	Transação de custódia, Contratos vinculados, Transações entre partes
Transação Financeira	Capital social, Pravite equity, Títulos, Derivativos,crowdfunding.
Registro Público	Títulos de propriedade, registro de veículos, certificado de casamento.
Identificação	Licença de motorista, RG, passaportes, registro de voto.
Registro Privado	Contratos, apostas, assinaturas
Testemunho	Prova de seguro, prova de propriedade,
Chave para ativo físico	Casa, quarto de hotel, aluguel de carros, acesso a automóveis.
Ativo Intangível	Patentes, marcas registradas, direitos autorais, reservas.

Fonte:(Adaptação de Letra Capital Mega Master Blockchain List, apêndice 1[7])

Os conceitos, ideias e ferramentas usados atualmente para descrever o que conhecemos por blockchain surgiram através de Stuart Haber e W. Scott Stornetta em 1997 com a publicação do paper chamado "*How to Timestamp a Digital Document*". O paper apresenta procedimentos computacionais práticos para inserir nos registros data e hora de modo que seja inviável a alteração mesmo com a possibilidade de colisão dos dados, mantendo a privacidade dos mesmos.

Pela definição de Harber e Stornetta, o bloco foi definido como sendo um "cofre digital" que contém o dado de qualquer tipo. O serviço de *time-stamp* (TSS) é o responsável pela inserção de data e hora na entrada da informação e manutenção de uma cópia de segurança do arquivo,

em caso de ameaça à integridade do documento é feita uma comparação com a cópia armazenada pelo TSS. [6]

O bitcoin apresentado por Satoshi Nakamoto em 2008, usa como componente base a tecnologia do blockchain pautada nas ideias de Harber e Stornetta. O blockchain, de modo geral, representa uma lista de registros crescentes, em que os dados estão dispostos em blocos e estão vinculados usando criptografia. [7]

Cada bloco contém um hash criptográfico do hash anterior, podendo ser interpretado como uma espécie de "carimbo" com informações, tais como: a data e hora da inserção do registro à cadeia e dados específicos da transação representados por um código hash que não pode ser alterado. [8]

A confiabilidade e integridade do blockchain se baseia em diminuir as chances de haver dados fraudulentos e para tanto, o pilar usado pela tecnologia para manter esta confiabilidade é o *hashing*.

Uma função hash toma um valor de tamanho arbitrário como entrada e produz uma *string* de saída com tamanho fixo, análogo à uma impressão digital. Devido a propriedades adicionais, as funções hash criptográficas são adequadas para a verificação de integridade de uma mensagem como parte de uma assinatura digital.

Uma função hash H, toma como entrada uma mensagem de comprimento arbitrário e produz uma mensagem resumida com comprimento fixado. A mensagem original passa a ser compactada e se torna a entrada da função de verificação. [9].

De forma geral e simplista, para realizar verificações é necessário haver uma chave secreta, uma chave pública e duas funções: uma para produzir a assinatura e outra para retorno de um valor booleano, indicando a validação de uma assinatura dado uma mensagem. A assinatura tem como função provar a autenticidade, por parte do remetente e integridade da mensagem e é utilizada para provar o envio da mensagem por parte do remetente original. [9]

Se tornam possíveis dois tipos de falsificações. A tentativa de falsificação existencial, onde o atacante cria um par válido de mensagem e assinatura; e a tentativa de falsificação universal em que o atacante calcula uma assinatura válida dado uma mensagem e uma chave pública. [9]

O algoritmo hash é mapeado de modo a ser praticamente impossível encontrar uma função inversa, as formas possíveis de recriar a informação de entrada se dão através de tentativa exaustiva via testes para encontrar o valor correspondente ou por meio do método conhecido por *rainbow table*, tabela pré-computada usada para a inversão de funções hash na descoberta de senhas acima de um certo comprimento, para casos de caracteres limitados. [9]

Uma função hash criptográfica idealmente possui as seguintes características:

1. É determinística, a mesma mensagem sempre resulta no mesmo código hash.
2. O cálculo de valor de hash é rápido para qualquer mensagem
3. Se torna inviável gerar a mensagem a partir do valor do hash, com exceção da tentativa de todas as possibilidades
4. Uma mudança pequena na mensagem, produz uma mudança exponencial no hash obtido de modo a parecer descorrelacionado com o hash anterior.
5. Impossível encontrar duas mensagens diferentes dado o valor de hash.[9]

O algoritmo hash é voltado para aplicações que requerem alta segurança de informação em processos de autenticação, como no blockchain, em que os blocos são ligados através de códigos hash e apresentam um grupo de transação postado sequencialmente comparado à um livro de registros.

Os registros inseridos na cadeia de blocos não podem ser alterados, em caso de alteração de acordo com as propriedades da função hash, o link criptográfico irá mudar, e todos o blocos conectados à rede perderão sua combinação. Torna-se difícil para o invasor mudar todos os blocos da rede uma vez que blocos são adicionados e sincronizados ao longo da rede constantemente, checando a validade dos hashes. [10]

No blockchain, assim como em um sistema P2P distribuído todos os computadores da rede estão conectados e as informações da rede são compartilhadas em todos os computadores, também conhecidos como nós. A conexão se dá através de chaves criptográficas garantindo a confidencialidade das informações.

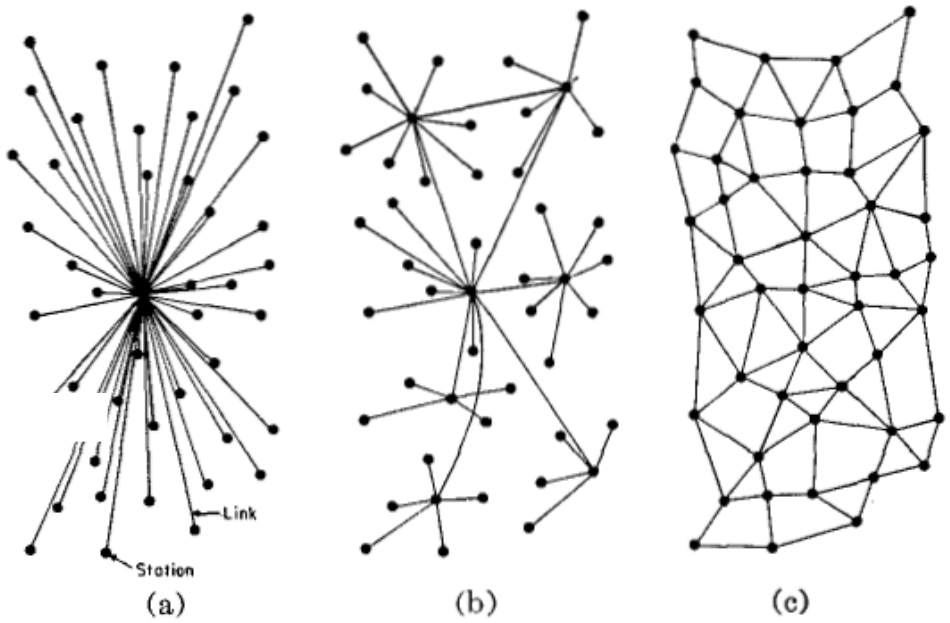
Cada nó representa parte dos recursos computacionais e pode compartilhar seu processamento, armazenamento e largura de banda com toda a rede. Se um simples nó da rede existir com cópia de todo o blockchain, todos os registros da rede permanecerão intactos, com possibilidade de reconstrução total da cadeia de blocos.

Esta espécie de livro-razão elimina a necessidade de uma autoridade central controlando as informações armazenadas, este banco de dados se torna imutável e está de acordo às regras estabelecidas entre os participantes da rede.

Os dados ficam propensos a ataques cibernéticos, mas um atacante deve atacar toda a rede simultaneamente, e não apenas um único elemento para obter sucesso. O consenso da maioria ganha e conforme a rede aumenta, aumenta também a dificuldade de um ataque bem-sucedido. [11]

A figura 3 ilustra de maneira intuitiva as principais diferenças entre os tipos de redes existentes.

Figura 3-Esquema para diferenciação entre os tipos de rede existentes. (a)Redes centralizadas, (b) Redes descentralizadas e (c) Redes Distribuídas.



Fonte: BUTERIN, Vitalik. The Meaning of Decentralization. 2017. Disponível em: <https://medium.com/@VitalikButerin/the-meaning-of-decentralization-a0c92b76a274>. Acesso em: 15 ago. 2016.

Uma rede centralizada possui todo o controle em um único ponto, em uma rede descentralizada o processamento das transações ocorre em diferentes pontos, já uma rede distribuída mostra que nenhum ponto tem controle sobre os demais e se torna útil pois passa a ser:

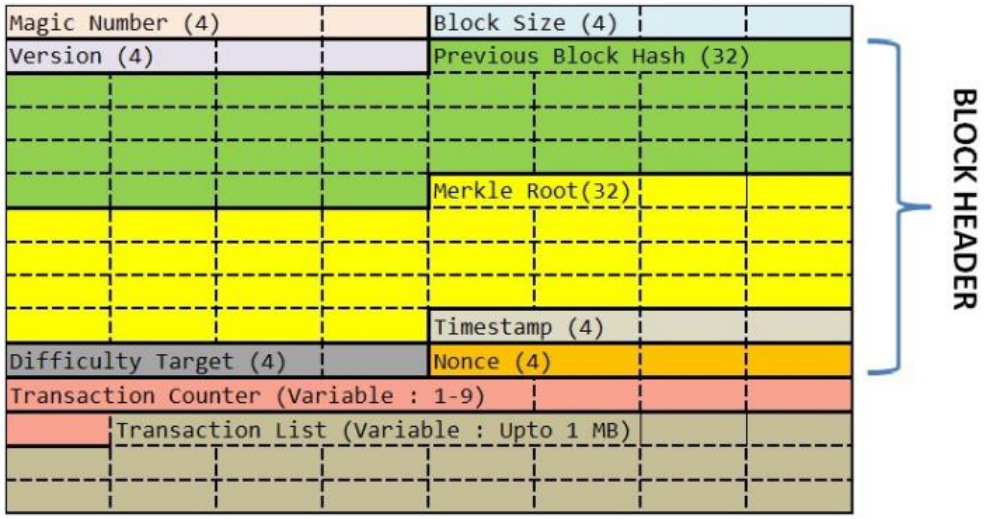
- Resistente a falhas – componentes são separados diminuindo a probabilidade de falha.
- Resistente a ataques – não existem pontos centrais sensíveis que possam ser atacados,
- Resistente a conspirações – torna-se difícil para os participantes beneficiarem alguns em função de outros como acontece hoje em dia por parte de governos e grandes coorporações. [12]

2.1 Estrutura de um bloco e infraestrutura da rede

Para entender a estrutura de um bloco, foi estudada a estrutura de um bloco partindo do Bitcoin, por ser a primeira implementação prática testada em tempo real enquanto as demais implementações de blockchain se encontram ainda em fase de prova de conceito.

Cada bloco no bitcoin possui a mesma estrutura, o cabeçalho de bloco é constituído principalmente do: número da versão do bloco, um timestamp, hash usado no bloco anterior, hash da Merkle Root, o nonce e o hash de destino. O esquema da composição do bloco é mostrado na figura 4.

Figura 4-Esboço ilustrativo da estrutura de um bloco.



Fonte: [13]Vaidya, K. **Bitcoin's implementation of Blockchain**, 2016 Disponível em: <https://medium.com/all-things-ledger/bitcoins-implementation-of-blockchain-2be713f662c2> Acessado em: 25 de Agosto de 2018.

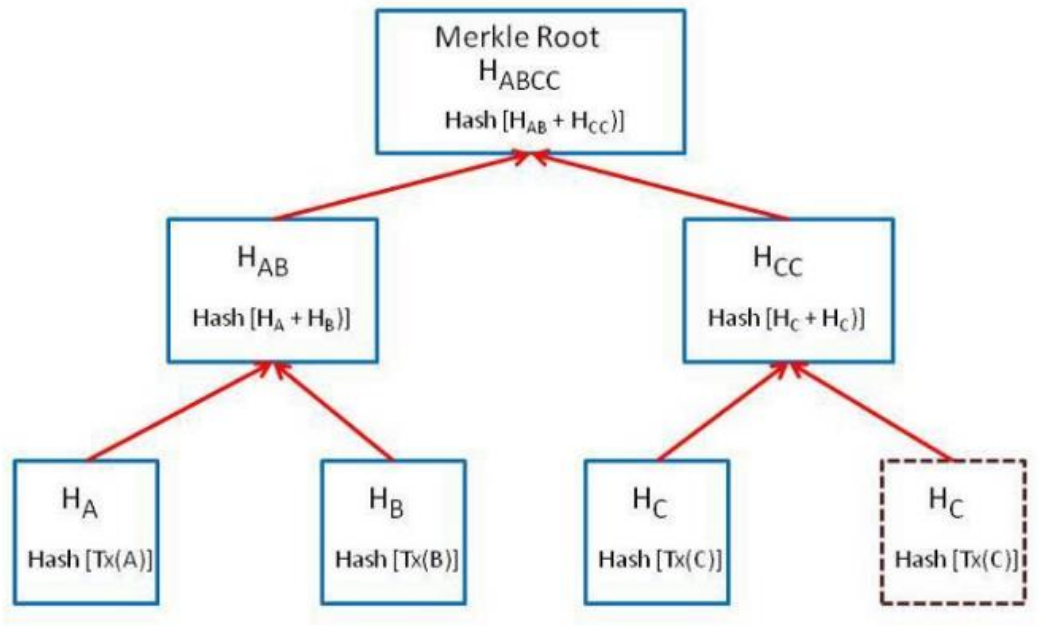
Cada célula individualmente representa 1 byte e, portanto, um campo de 4 bytes ocupa 4 células. O cabeçalho é formado desde o campo de versão (version) até o campo nonce com um total de 80 bytes.

Abaixo o detalhamento dos campos:

- Magic Number (Número Mágico): campo identificador do blockchain na rede, que indica o início de um bloco da rede.
- Block size (Tamanho do bloco): indica o tamanho fixo de um bloco 2MB
- Versão (version) cada nó deve implementar a mesma versão.
- Previous block hash: o identificador único do bloco anterior adicionado ao blockchain. É calculado levanto em conta todos os campos do cabeçalho e aplicando a função criptográfica (SHA-256).
- Merkle Root: formato da estrutura das transações, lista dos hash criptográficos de todas as transações
- Timestamp: registra data e hora da inserção de registro.
- Transaction Counter: contabiliza transações adicionadas ao bloco
- Transaction List: armazena os identificadores únicos de cada transação no bloco, cada transação possui sua própria estrutura. [13]

Cada bloco contém um resumo das transações e a partir do momento que o bloco passa a fazer parte da rede, as informações se tornam imutáveis. As transações são listadas como merkle tree conforme esquematizado na figura 5.

Figura 5– Exemplo de uma Árvore de Dispoersão para 4 transações.



Fonte: Vaidya, K. **Bitcoin's implementation of Blockchain**, 2016 Disponível em: <https://medium.com/all-things-ledger/bitcoins-implementation-of-blockchain-2be713f662c2> Acessado em: 25 de Agosto de 2018.

A raiz da árvore se encontra no nó mais elevado, os nós abaixo são chamados de nós-folhas, cada nó contém o hash criptográfico da transação. As transações A, B e C representam os nós-folhas da árvore.

A árvore de dispersão não possui a lista de todas as transações, apenas armazena uma espécie de impressão digital de todas as transações auxiliando na identificação das mesmas. Faz parte da infraestrutura do blockchain, composta por nós que armazenam, distribuem e mantém os dados dos blocos. [13]

O blockchain pode ser executado em um único nó, porém o armazenado seria centralizado em um único dispositivo que estaria vulnerável a falhas de energia, hackers ou travamentos sistêmicos. Quanto maior o número de nós completos, melhor a resiliência contra catástrofes. [13]

Os nós na rede podem estar online ou offline. Os nós on-line recebem, salvam e transmitem os últimos blocos de transações para outros na rede. Quando um nó offline voltar a ficar on-line, precisará ser sincronizado com restante do blockchain e irá baixar todos os blocos que foram adicionados à rede desde que o nó ficou offline. [13]

2.2 O processo de Mineração

Os indivíduos ou companhias que processam blocos são chamados de mineradores e o processo para adição de um novo bloco envolve o descobrimento do código hash, que corresponde a solução de um problema matemático estabelecido conforme regras da rede para a criação do bloco.

O valor do campo hash pode ser manipulado de acordo com a mudança no campo *nonce* ilustrado na figura 4, campo presente no cabeçalho do bloco, o minerador deve ser o primeiro a encontrar a sequência para este campo específico. O *nonce* é anexado ao conteúdo de hash do bloco e reescrito. Quando o hash atender aos requisitos estabelecidos para ser solução do problema, o bloco será adicionado ao blockchain.[14]

Determinar o caractere a ser usado como *nonce* requer uma quantidade significativa de tentativa e erro, pois tal campo se trata de uma sequência de caracteres aleatória. Um minerador deve obter o *nonce*, anexá-lo ao hash do cabeçalho do bloco atual, refazer o valor do campo e compará-lo ao cídigo hash alvo.

É altamente improvável que se descubra com sucesso o *nonce* na primeira tentativa. Quanto maior a dificuldade, mais tempo levará para gerar uma solução. O processo de busca de soluções para acertar o campo *nonce* recebe o nome de prova de trabalho. [14]

A dificuldade para geração de um novo bloco é mantida da mesma forma em toda a rede, o que significa que todos os mineradores têm a mesma chance de descobrir o hash correto. Normalmente, as redes de criptomoeda estabelecem um número-alvo de blocos que desejam processar durante um período específico e ajustam periodicamente a dificuldade para garantir que essa meta seja atingida. [14]

Todos os mineradores são necessariamente nós completos da rede, pois para a criação de um novo bloco se torna necessário obter todo o histórico de informações prévias da rede, porém um nó nem sempre é um minerador pois não necessariamente precisa ser criado um novo bloco para obter esta caracterização, tendo função muito similar à um pequeno servidor.[15]

Um nó completo da rede valida cada bloco e cada transação existente nos blocos. Esta validação é feita através das regras de consenso definidas no blockchain. [15]

Define-se como consenso a “Concordância ou unanimidade de opiniões, raciocínios, crenças, sentimentos etc. em um grupo de pessoas; decisão, opinião, deliberação comum à maioria ou a todos os membros de uma comunidade.” [16].

No contexto de sistemas distribuídos todos os nós estão ordenados em sequência, contendo um nó “líder” se comunicando com os demais denominados nós de backup. Os nós honestos devem chegar a um acordo sobre o estado do sistema, vencendo a decisão da maioria.

Para garantir a comunicação efetiva, assim que uma mensagem é recebida o modelo deve provar de qual nó veio a informação e verificar se a mensagem foi modificada durante a transmissão. [19]O mecanismo de consenso no blockchain busca resolver os seguintes desafios:

Desafio 1: Proteger a rede contra eventuais ataques: adição de um bloco malicioso ao fim da cadeia de algum bloco por exemplo.

Resolução adotada: após adição de um novo bloco, o minerador receberá uma recompensa pelo trabalho feito além disso obterá taxas associadas a transações inseridas no bloco específico, existem incentivos financeiros para cada novo bloco. Caso tenha conteúdo malicioso na inserção de um bloco, o minerador ele não receberá por isto. [19]

Além disso, todos os nós da rede executam um checklist antes de adicionar um novo bloco, itens como por exemplo árvore de dispersão, o hash do bloco anterior, *timestamp*, conferência do conteúdo nas transações entre outros. Caso um dos itens não esteja de acordo o bloco passa a ser rejeitado na rede. [19]

Desafio 2: Cadeias competitivas: dois nós podem ser minerados ao mesmo tempo.

Resolução adotada: Caso dois blocos sejam adicionados ao mesmo tempo, espera-se até que um terceiro novo bloco seja adicionado, após isto é verificado o tamanho da cadeia, e a maior permanecerá e substituirá a anterior.

A rede com maior poder computacional para a checagem de hash por segundo tende a ter maiores chances em resolver o desafio criptográfico primeiro. No blockchain os que tiverem mais de 51% de poder computacional entrando em consenso tem uma maior probabilidade em adicionar o bloco.[19]

Característica importante para todos os sistemas descentralizados, está relacionada ao funcionamento correto de uma rede distribuída e sua habilidade em atingir consenso suficiente, mesmo que haja falha sistêmica e componentes (nós) propagando conteúdo malicioso.

Sempre que surge uma nova transação os nós podem ignorá-la ou inclui-la em seu “livro-cópia”. Em caso de falhas para garantir que o consenso foi alcançado se torna importante estabelecer um protocolo de segurança resistente a falhas.

A base para tal protocolo foi publicado pela primeira vez em 1975, e em 1978 recebeu o nome o problema dos dois generais que tinham o mesmo inimigo em comum, e portanto, precisavam juntar forças para vencer a batalha, atacando o inimigo ao mesmo tempo.

Entre ambos os generais havia o campo de batalha e a mensagem contendo a hora do ataque, deveria chegar ao exército aliado, mas nenhum dos dois saberia se um ou outro teria concordado com o plano no momento certo. Para tal problema foi provado não haver solução. [17]

Em 1982 surgiu uma versão generalizada do problema anterior denominada Problema dos Generais Bizantinos que descreve o mesmo cenário, porém aumenta o número de generais e insere o fato de que um dos generais pode ser o traidora. O teorema dita que para qualquer m, sendo m o número de generais, o algoritmo atinge consenso se houver mais de 3m generais e no máximo m traidor. [17]

Para o problema dos generais Bizantinos, enquanto 2/3 da rede for honesto, o consenso é alcançado, ou seja, a maioria ganha. A solução probabilística para o problema dos Generais Bizantinos surgiu com o conceito de Prova de Trabalho o paper de Satoshi Nakamoto. [7].

No contexto do problema dos generais bizantinos o conceito de prova de trabalho se aplica da seguinte forma: a mensagem original a ser enviada pode ser anexada a um *nonce*, uma função hash pode ser aplicada à mensagem e ao *nonce*, os exércitos poderiam entrar em um acordo para o compartilhamento da mensagem sob uma determinada condição e os dois exércitos manteriam suas tentativas até que atingissem o resultado esperado.

Caso a mensagem fosse pega pelo inimigo as propriedades do hash fariam com que a mensagem mudasse drasticamente e por outro lado se os generais notassem que a mensagem não chegou de acordo com a condição estipulada previamente eles poderiam abortar o ataque.

A ideia por trás da prova de trabalho é fazer com que o processo para encontrar o *nonce* na descoberta do hash alvo se torne trabalhoso, desta forma, a adição de um novo bloco requer alta quantidade de energia e uso computacional. Enquanto os problemas são projetados para serem complexos, o processo de verificação para adição do bloco deve ser rápido afim de diminuir o tempo de latência da rede. [20]

2.3 Transações no blockchain

O envio de determinado recurso no blockchain, ocorre via transação a partir de uma carteira de aplicação, aqui carteira está relacionada ao meio onde os objetos de envio são armazenados. A transação é transmitida pela aplicação e enquanto um novo bloco não for minerado, a transação aguardará no chamado “conjunto de transações não confirmadas”, lugar onde se encontram todas as transações ainda não processadas. [21]

Os mineradores da rede ou ‘nós’ selecionam as transações para a formação dos blocos, verificando a elegibilidade de execução de cada uma, no caso de bitcoin confirmando se há saldo suficiente na carteira dos remetentes; se houver, a transação será considerada válida e poderá ser adicionada ao bloco. [22]

Cada transação pode ou não ter uma taxa associada, no bitcoin as carteiras utilizam o conceito de taxa dinâmica para as transações, onde as taxas dependem das condições da rede em um determinado período e do tamanho das transações. Os usuários competem para terem suas transações adicionadas aos blocos. [23]

Cada bloco pode conter um tamanho fixo e portanto, o número de transações a serem incluídas no bloco é limitado. Durante o tempo de congestionamento quando vários usuários estão trocando recursos na rede os mineradores podem priorizar as transações com maiores taxas de incentivo para que os beneficiem. O tamanho das transações influencia, pois, transações menores são mais fáceis de validar enquanto transações maiores requerem mais espaço no bloco e maior trabalho computacional envolvido. [21]

Após a seleção de transações no bloco os mineradores precisarão de uma assinatura para adição do bloco na rede, ou seja, para que o bloco seja reconhecido via mecanismo de consenso. Cada vez que um novo bloco é adicionado à rede é necessária uma confirmação do bloco anterior, pois quanto mais confirmações, menores as chances de alteração nos dados armazenados.

Os primeiros mineradores que encontram a assinatura elegível para o bloco a compartilham para todos os outros nós, a solução passará a ser verificada se corresponder ao problema endereçado para a criação do bloco, a assinatura será validada pela maioria da rede e o bloco é adicionado ao blockchain. [23]

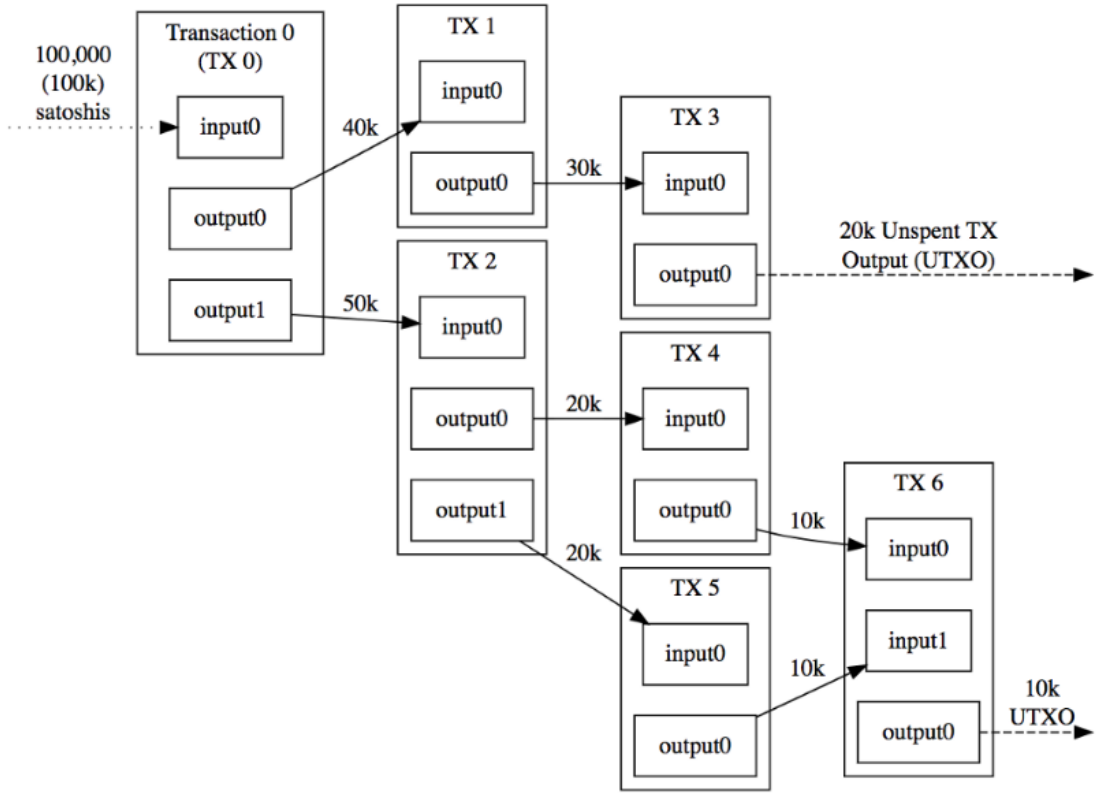
Quando um nó aceita um novo bloco de transações, ele salva e armazena os dados no fim da "fila" no bloco. Os nós: verificam se o bloco de transações é válido, se válido, salvam, armazenam blocos de transações, contendo o histórico de transações no blockchain, e transmitem este histórico para outros nós visando o si ncronismo da rede. Caso o bloco não seja [21]

A cada adição de um novo bloco, os mineradores recomeçam o processo de combinação de transações para a criação de um novo bloco, pois transações já inseridas pelo bloco anterior serão consideradas inválidas, tornando o bloco inválido. [22]

Para exemplificar o que ocorre com as transações, este trabalho considera o que ocorre no bitcoin em que uma quantidade específica de recurso sai de uma carteira para o blockchain assim que um novo bloco é minerado.

Neste trabalho, o conceito de carteira não está relacionado à uma quantidade de dinheiro específica e sim a chamadas saídas de transação não gasta (UTXO). Todas as transações que ainda não foram gastas são mantidas em um nó totalmente sincronizado, que é o conjunto de transações não confirmadas. A figura 6 apresenta como a saída gerada por transações anteriores gera saída de transações futuras. [24]

Figura 6– Esquema do sistema de transações usados no Bitcoin.



Fonte: (Sun, Flora., 2018)[6]

A carteira de um usuário acompanha a lista de transações não gastas associadas a todos os endereços que o usuário possui, o saldo é dado com base nas transações não gastas. As transações não gastas podem ser associadas à conta e a quantidade de contas registra o total de recurso que o usuário possui. [24]

Quando se deseja utilizar esse recurso, utiliza-se as UTXO existentes, a cada uso as transações somem, pois são consumidas, ou seja, "gastas", elas também podem ser somas ao saldo total se houver envio por parte de outro usuário da rede. [24]

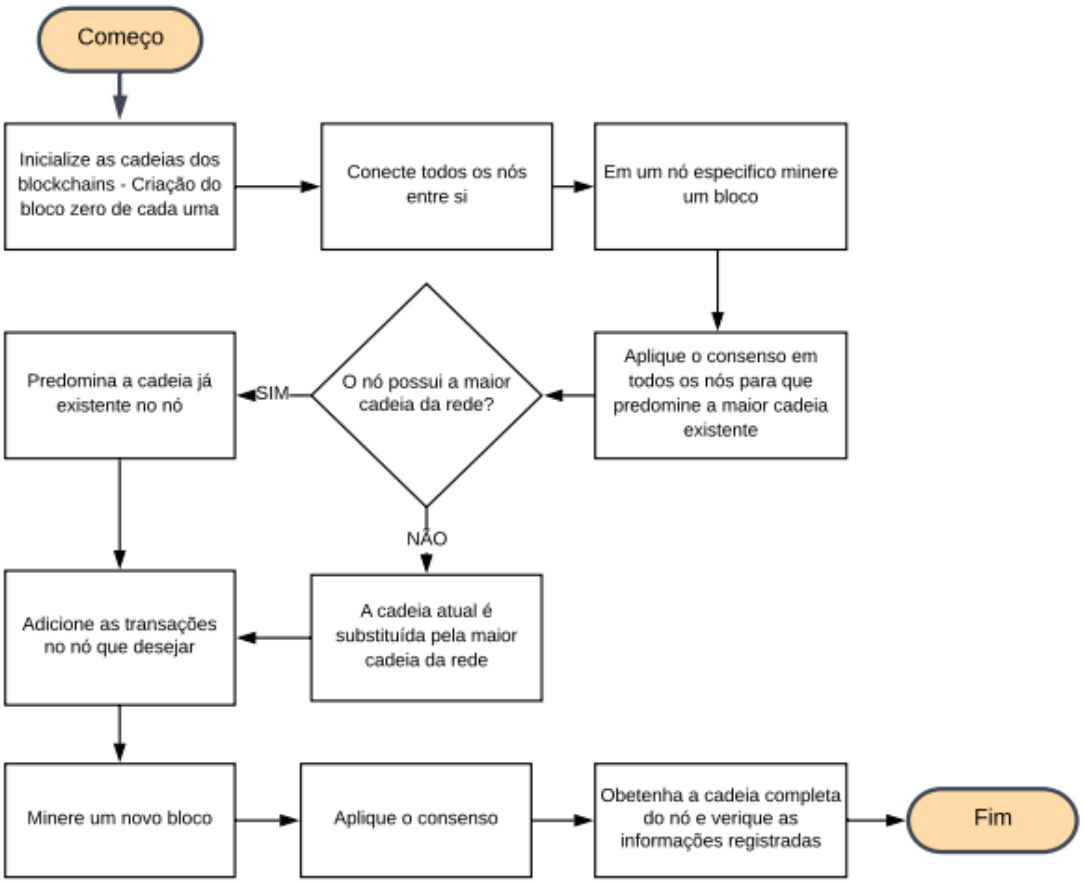
Técnicas criptográficas avançadas garantem a privacidade e integridade das mensagens nas transações. O autor da transação é o responsável pela Chave privada, identificador único, utilizado para assinar cada transação que um usuário envia ao blockchain. Tal assinatura confirma a legitimidade da transação por parte do usuário e impede sua alteração por parte de terceiros. Chave pública: pode ser compartilhado entre outros, a chave privada ainda será necessária. [25]

A mensagem a ser enviada combinada com a chave primária gera a assinatura única para o dono da chave. No blockchain existe a função de verificação que toma como entrada a mensagem, assinatura e a chave pública e confirma se a mensagem realmente foi assinada pela chave privada. [25]

3. Simulação

O diagrama da figura 7 apresenta uma representação visual dos passos seguidos durante o processo de simulação e teste do blockchain desenvolvido.

Figura 7- Digrama de atividades do Blockchain desenvolvido.



O blockchain foi construído utilizando a linguagem de programação Python através da plataforma livre e distribuída, Anaconda, voltada ao processamento de dados em larga escala. Ela simplifica o gerenciamento dos pacotes através de seu gerenciador de ambientes virtuais, o Anaconda Navigator, que possui mais de mil pacotes e elimina a necessidade de instalação das bibliotecas que são usadas independentemente. [26]

Para construção do blockchain foi utilizado o micro-framework Flask baseado na biblioteca WerkZeug, voltada ao desenvolvimento de aplicações denominada Interface de Porta de Entrada do Servidor (WSGI) interface para servidores web que encaminham aplicações web ou frameworks escritos na linguagem de programação Python. A biblioteca possui ainda implementação básica do padrão para interceptar requests e lidar com response, controle de cache, cookies, status HTTP e roteamento de URLs . [27]

Para o teste da aplicação foi utilizado o Postman, uma aplicação web que permite enviar e receber dados via requisição HTTP a partir de sua interface. O serviço é construído de acordo com o modelo REST (Representational State Transfer) que define as restrições a serem utilizadas na construção de uma aplicação web. O postman possibilitou controle dos dados trafegados [28].

Não está no escopo deste trabalho o desenvolvimento de interface gráfica para o teste do serviço.

3.1 Construção do Blockchain

A primeira parte consistiu-se na construção da arquitetura do blockchain, funções para obtenção do estado do blockchain e a mineração de um novo bloco na rede.

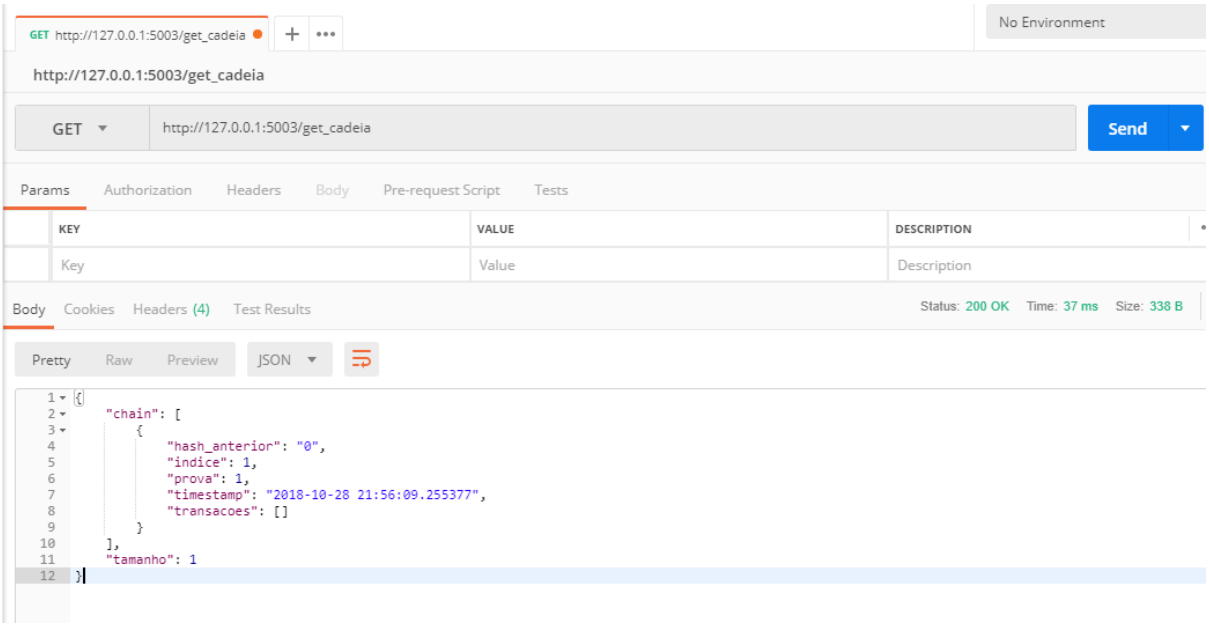
Inicialmente as bibliotecas utilizadas para construção do blockchain foram:

1. A **datetime**, dado que cada bloco tem seu prórpio ***timestamp***, tempo e data exata em que o bloco foi criado
2. A **hashlib** para trabalhar com funções hash dos blocos
3. A biblioteca **json** foi utilizada para codificar os blocos antes de inserir a função hash.
4. A biblioteca **flask** para a criação do objeto, por objeto entende-se a própria a aplicação web.
5. A biblioteca **jsonify** utilizada para retornar as mensagens no Postman a cada interação do blockchain.

O Blockchain teve como proposta ser uma rede descentralizada com outros blocos encadeados, a construção foi baseada em classes, o que facilita a inclusão de propriedades, funções, ferramentas e métodos.

A inicialização da classe **Blockchain** incluiu a inicialização da cadeia com os blocos que a serem criados na rede, de início uma lista vazia que chama a função **cria_bloco** para a criação do bloco gênesis 0 e dos demais blocos, cada bloco irá conter um valor arbitrário para **prova,** valor escolhido 1 e o **hash anterior** que funciona como link para os outros blocos da cadeia. A figura 8 mostra a criação do bloco gênesis através da requisição GET, get_cadeia.

Figura 8- Criação do bloco Gênesis no Blockchain



A função **cria_bloco,** foi aplicada logo em seguida a chamada da função para minerar um novo bloco **minera_bloco**, a função para mineração de um novo bloco obtém a prova de trabalho (proof of work) usado para resolver o problema, dado o algoritmo escolhido. Se encontrada a prova de trabalho, significa que o bloco foi minerado e que está função de criação de bloco pode ser chamada.

A função **minera_bloco** não teve argumentos, os objetos usados vieram a partir do blockchain criado, após obtenção da prova de trabalho do bloco anterior o método cria_bloco foi chamado, fornecendo como variável de resposta toda a informação do bloco. Foi retornada a mensagem e o status HTTP 200 significando que a execução ocorreu sem erros.

A função cria_bloco tem como argumentos ela mesma, uma vez que irá utilizar as variáveis dela mesma como objeto, o segundo argumento é a variável que contém o resultado da função minera_bloco denominada **prova** e o terceiro argumento sendo o link entre o bloco anterior e o novo bloco criado, o **hash_anterior.**

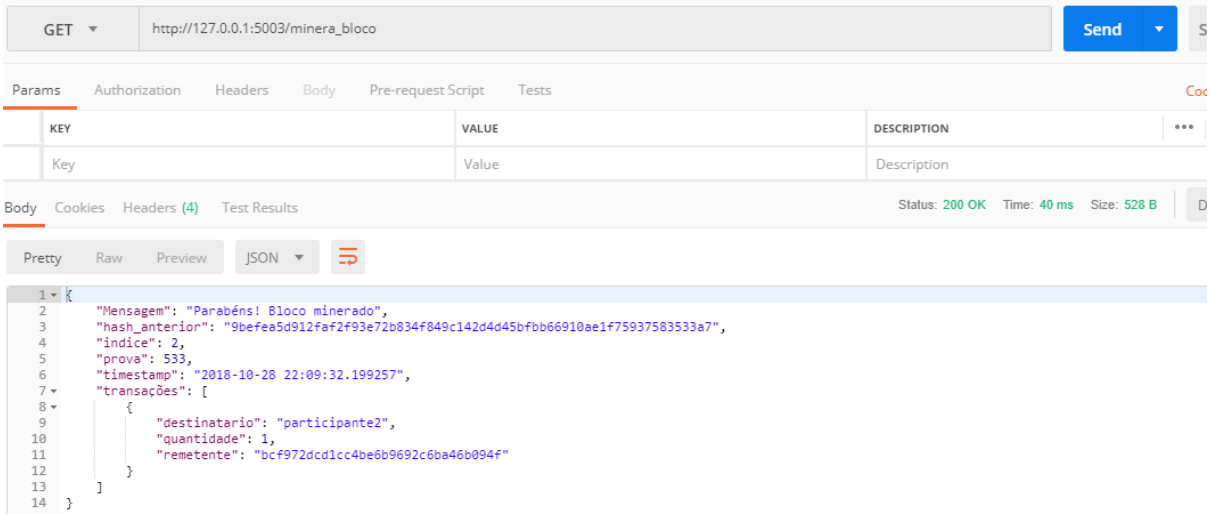
Na função cria_bloco foi definido que cada novo bloco da rede apresentaria um dicionário com cinco elementos: **índice** chave do bloco, ***timestamp*** contendo data e hora exatos em que um bloco é minerado, a **prova** , o **hash_anterior** e as **transações** que cada bloco contém.

A função **prova_de_trabalho** definiu o problema a ser resolvido pelos mineradores para encontrar a prova necessária na geração de um novo bloco. Seguindo a orientação teórica a prova de trabalho deve ser complexa de resolver, mas facilmente verificável.

A resolução do problema utiliza uma repetição contando as interações até encontrar a solução ótima. Foi utilizada a função criptográfica **hash 256** combinada com a função ***hexdigest*** para o retorno da solução, uma string de 64 caracteres começando com quatro zeros à esquerda, quanto mais zeros, mais complexo a mineração

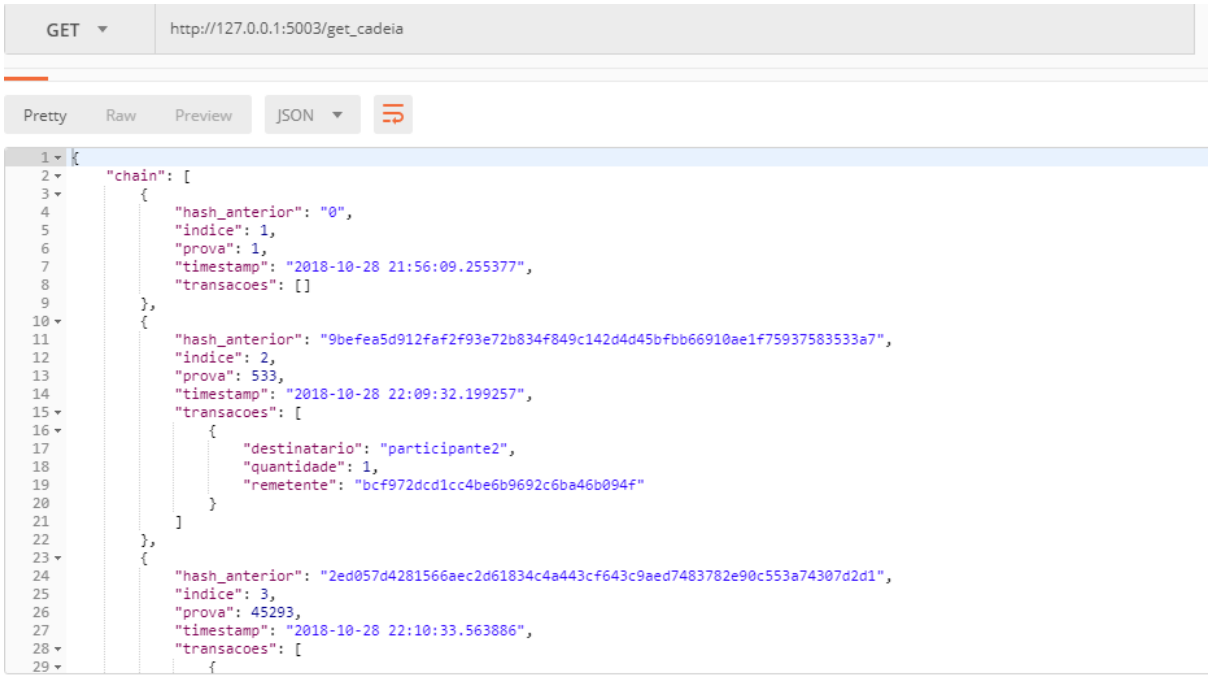
A figura 9 apresenta o retorno da requisição GET ao minerar um novo bloco.

Figura 9- Mineração de um novo bloco no Blockchain



A segunda requisição GET foi usada para mostrar todos os blocos do blockchain, a função **get_cadeia** contém em sua resposta todo o blockchain no formato json. Conforme mostra a figura 10.

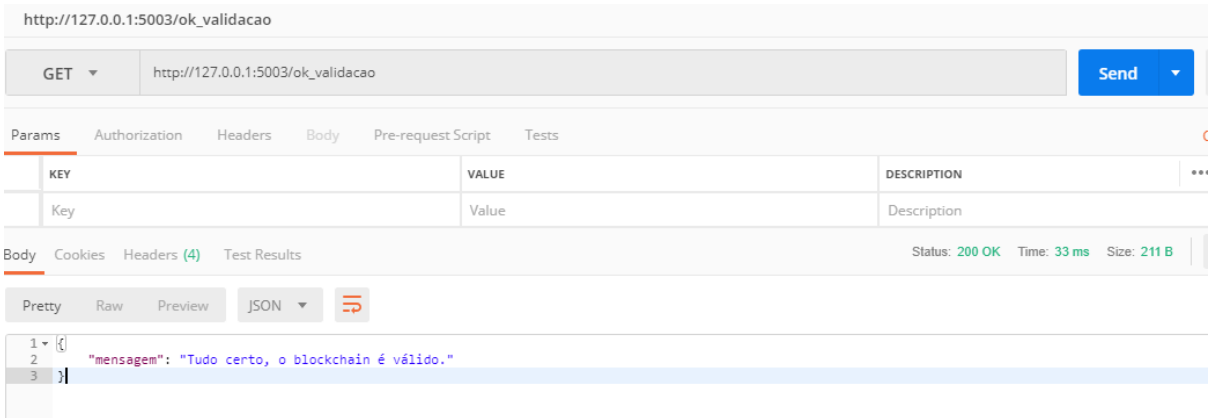
Figura 10- Visão geral do blockchain criado.



A variável **operacao_hash** contém a operação do problema a ser resolvido, a condição é que o problema seja não simétrico, uma vez que são utilizados dois tipos de chave distintos. Para este propósito a operação escolhida foi simples, podendo se estender a algo mais complexo. A operação completa foi colocada dentro de uma *string* e a função encode codificou no formato esperado pela função hash 256.

O método criado para validação do Blockchain verificou a prova de trabalho e o hash anterior, assegurando que o hash de cada bloco é igual ao hash do bloco anterior, conforme mostra a figura 11.

Figura 11-Validação do Blockchain



Para a validação foi necessário implementar a função **hash**, que retorna o hash criptográfico de cada bloco. A função toma um bloco como entrada e retorna o hash 256 do bloco, têm como

argumentos ela mesma e o bloco da rede. Para isto, foi feito do dicionário de cada bloco uma string, utilizando a biblioteca json e a função dumps para a criação do variável bloco_codificado.

```
def hash(self, block):
    bloco_codificado = json.dumps(block, sort_keys = True).encode()
    return hashlib.sha256(bloco_codificado).hexdigest()
```

A função de validação do Blockchain (**ok_validacao**) contém ela mesma como argumento e a cadeia de blocos que será verificada a cada interação.

Para simulação do envio de transações na rede blockchain, foram criadas outras consoles no Phyton com o intuito de criar um ambiente próximo ao descentralizado, ou seja, diferentes computadores conectados em diferentes servidores. Foram simulados três consoles, um para cada nó criado na rede. Conforme ilustram as figuras 12, 13 e 14.

Figura 12- Nó 1, rodando na porta 5001.

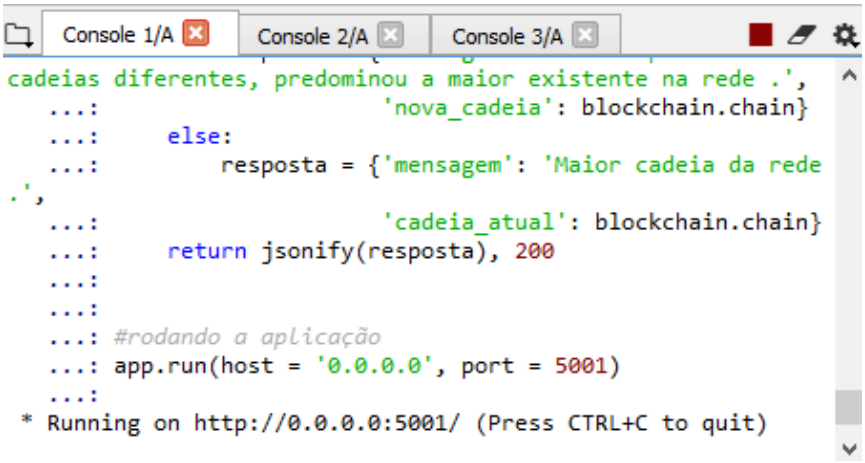


Figura 13-Nó 2 rodando na porta 5002

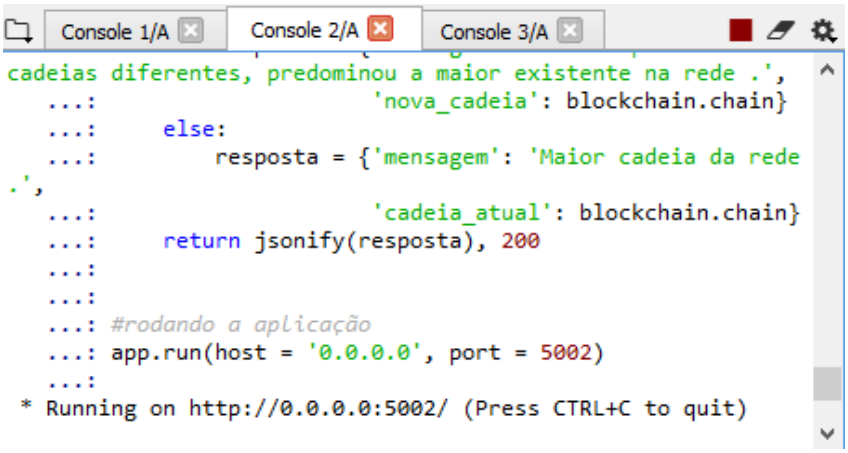
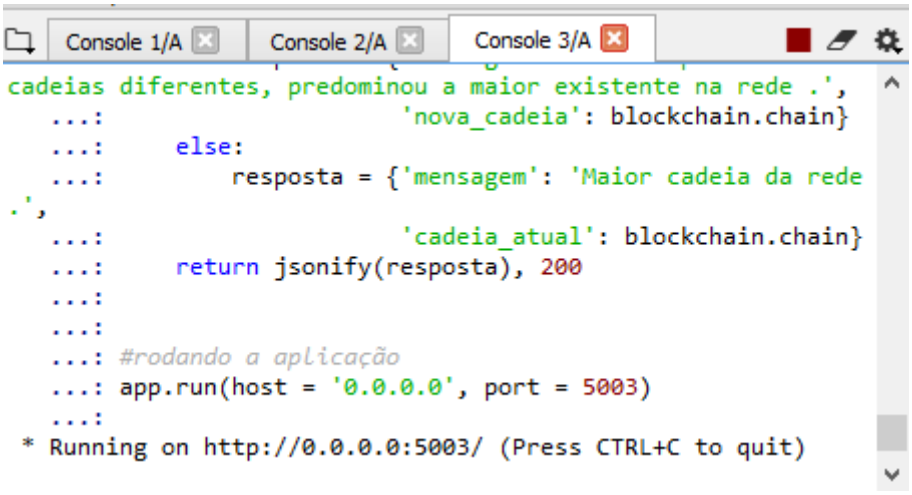
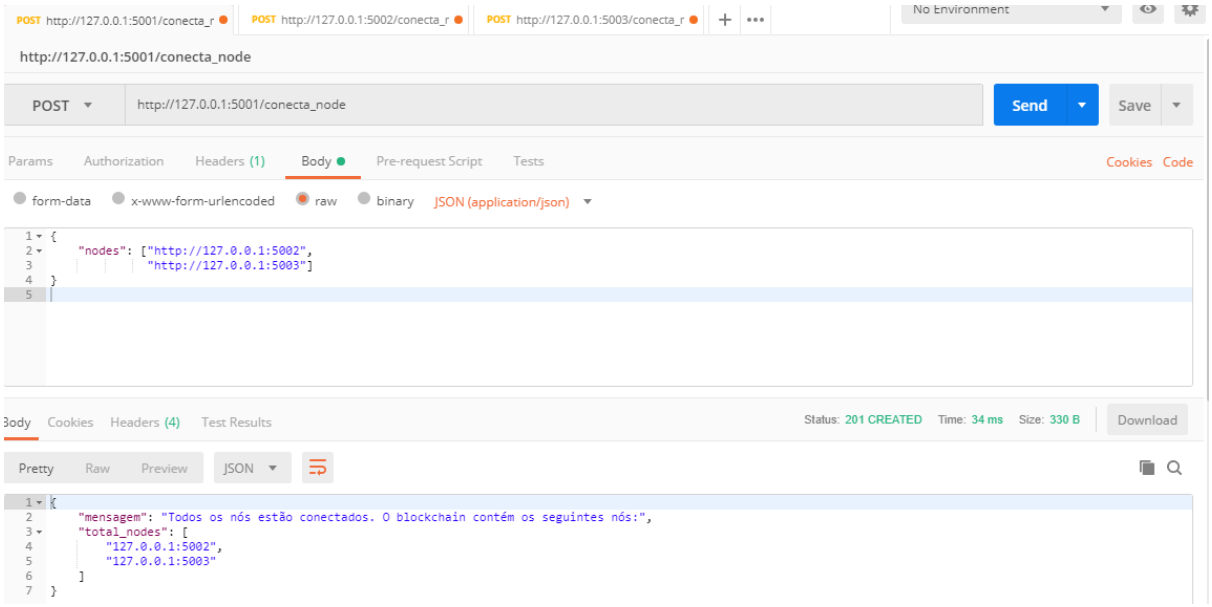


Figura 14- Nó 3 rodando na porta 5003



A adição de transações ao blockchain criado permite que a venda ou troca de qualquer recurso ocorra através de transações seguras adicionadas aos blocos através do processo de mineração. A conexão dos nós, denominados participantes da rede, se deu através da chamada de requisição POST apresentado na figura 15.

Figura 15 - Resultado da conexão entre nós da rede através da requisição POST.



O objeto **node** foi criado e inicializado como vazio, não precisou ser uma lista pois, para os nós da rede, a ordem não importa. A função uuid4 foi usada para gerar um endereço único para os nós. Um endereço exemplo de um nó é o http://127.0.0.1:5000/ encontrado na página de

documentação do flask [27]. As diferentes portas para os diferentes nós da rede foram identificadas como sendo:

http://127.0.0.1:5001/ para o participante 1,

http://127.0.0.1:5002/ para o participante 2,

http://127.0.0.1:5003/ para o participante .

O método **adiciona_node** adiciona o nó através do método add, tomando o endereço do mesmo ao conjunto de nós da rede e retornando o endereço fragmentado do nó. A função urlparse foi importada da biblioteca parse e possibilitou a criação de uma chamada genérica para cada nó da rede.

```
In [13]: node=url_parsed.netloc

In [14]: node
Out[14]: '127.0.0.1:5001'

In [15]: f'http://{node}/get_chain'
Out[15]: 'http://127.0.0.1:5001/get_chain'
```

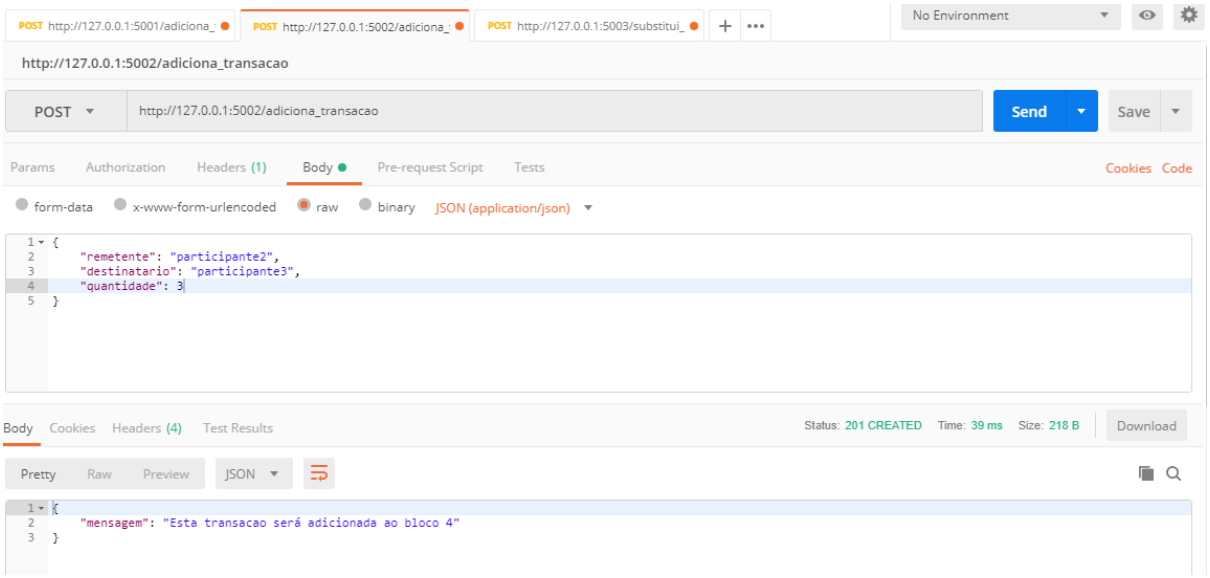
Foi introduzida uma nova lista, separada no método inicial, dado que as transações não nascem nos blocos, elas são anexadas em uma lista e que um bloco é liberado estas transações são inseridas após a mineração. [21]

Foi adicionada ao dicionário a chave transações que lista todas as transações de determinado bloco. Para a transação, a requisição HTTP chamada foi o POST, que diferentemente do GET irá criar a transação no formato json com os atributos do destinatário, remetente e quantidade e de acordo com a resposta desta requisição, a função jsonify irá adicionar a transação no novo bloco minerado.

Para o formato das transações foi criado o método **adiciona_trasacoes,** responsável por adicionar transações e por anexá-las a lista de transações existentes na rede. Os argumentos recebidos pelo método são: remetente, destinatário e quantidade enviada.

Além disso, o método adiciona_transacao verifica se os atributos chaves da transação estão de acordo com o esperado e caso algum esteja faltando, exibe o erro ao usuário através do código HTTP 400. Caso contrário, o programa busca pelo index do bloco para adicionar a transação ao novo bloco minerado na rede.

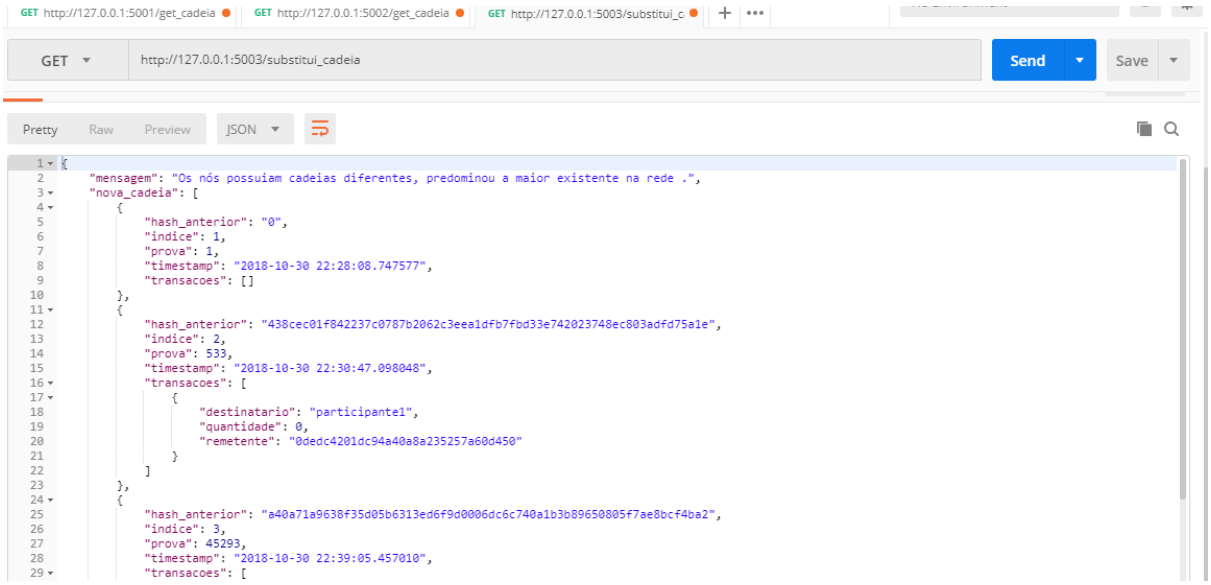
Figura 16-Ilustração do envio de uma transação da porta 5002(participante2) para a porta 5003(participante3).



A figura 16 mostra o envio de uma transação de um nó para outro, a transação foi criada no formato json e mostrada como resposta à requisição.

Para aplicação do consenso foi criada o método **substitui_chain**, responsável por substituir cadeias menores da rede pela maior existente dentro de cada nó. O método foi configurado para ser uma chamada específica dentro de cada nó e resultado é mostrado na figura 17.

Figura 17- Aplicação do mecanismo de consenso através da chamada do método substitui_cadeia



Foi criada uma variável denominada **nodes** contendo todo o conjunto de nós da rede. Para encontrar a maior cadeia da rede utilizou-se um loop interagindo com todos os nós da rede. A

variável **maior_cadeia** armazena a maior cadeia da rede e o método **get_chain** criado retorna a cadeia e o comprimento da mesma na tela.

Para obter a resposta do método get_chain no postman foi utilizada a biblioteca request . A função contendo a requisição get recebe como argumento o endereço do nó incluindo a porta e a requisição GET. Com a chamada do método é feita uma checagem da rede através do status do código HTTP. Para obter o tamanho foi utilizada a função json, dado que o dicionário do bloco está no formato json.

4. Contratos inteligentes

O aprendizado obtido na seção 3 mostra que o blockchain pode ser usado para registro de diferentes tipos de ativos, incluindo áreas econômicas, financeiras, ativos físicos e intangíveis. Desta maneira uma propriedade inteligente pode ser controlada via blockchain utilizando contratos sujeitos às leis existentes, os contratos inteligentes.

O conceito de propriedade inteligente se baseia na ideia de transacionar e controlar todos os tipos de ativos no modelo blockchain. Qualquer propriedade pode ser registrada como ativo digital através de uma transação no blockchain e esta propriedade pode ser controlada por qualquer pessoa que tenha em mãos a chave-privada referente ao registro. O dono do ativo pode vendê-lo transferindo sua chave privada para outra parte.

No geral, um contrato descreve um conjunto de direitos e obrigações para as partes envolvidas, que são usados para entre outras coisas encorajar relacionamentos de longo termo. [29]. Por outro lado, uma licença é utilizada para conceder permissão às partes (licenciado) de realizar atividades com produtos ou propriedades que de outra maneira seria ilegal, por exemplo um licenciante pode conceder permissão à cópia e distribuição de um software para um licenciado.

Dentre todas as definições, um contrato é um acordo legalmente válido entre duas partes. Perante à lei um contrato é considerado válido se contiver elementos tais como: oferta e aceitação, a intenção entre duas partes de criar relações de ligação, intenção de se pagar a promessa feita, capacidade legal de ação entre duas partes, consenso genuíno e legalidade no acordo realizado. [30]

Contratos escritos protegem as partes e garante que serão irrevogáveis em sua execução, quando há uma quebra de contrato ou baixa performance em sua execução as partes podem recorrer a cortes ou terceiros para garantir sua execução. Entretendo, tecnologia como o blockchain sugerem que alguns contratos agora passam a se tornar "inteligentes".

O termo contrato inteligentes ou smart contract, surgiu pela primeira vez em 1994 por um criptógrafo famoso conhecido como Szabo, estes contratos automatizados basicamente funcionam como qualquer outro programa de computador baseado em declarações if-then.[31, 32]

Segundo Idelberger[33] um contrato inteligente é um programa de computador que detém os termos do acordo contratual e implementa este acordo enquanto garante verdade, transparência e entendimento entre partes.

Além disso, um contrato inteligente pode ser considerado apenas um nome para um código de computador que é executado no blockchain e interage com os estados do blockchain [34]. Nesse contexto, contratos inteligentes são programas de computador pré-escritos e auto-executados que são armazenados em uma plataforma compartilhada (blockchain) e executados pela rede de computadores, garantindo a veracidade nas transações blockchain. Eles interpretam e armazenam dados em um banco de dados com contínua atualização nas listas de transações.

Portanto não se faz necessário uma terceira parte para articular, verificar ou garantir o acordo entre partes. Enquanto o blockchain possibilita o armazenamento de dados de maneira confiável e distribuída, os contratos inteligentes nos fornecem o julgamento confiável e distribuído dos acordos.

O fato destes programas serem executados em um blockchain os faz diferentes dos outros tipos de softwares existentes. O programa em si é registrado no blockchain, garantindo então sua permanência única [35], o software pode por si só gerenciar seus ativos, ou seja, no caso do bitcoin pode armazenar e alocar pedaços de criptomoeda. O programa é desencadeado pela plataforma significando que irá funcionar como designado e que ninguém poderá obstruir sua aplicação.

Novas tecnologias têm surgido possibilitando que tarefas ainda mais complexas possam ser realizadas através de linguagens de programação completamente desenvolvidas, em termos de Internet das coisas (IOT) um contrato inteligente pode ser usado por exemplo para destravar a porta de uma residência. Surgem questões relacionadas a como tal tecnologia irá se alinhar ao sistema legal existente, dado que um contrato físico escrito pode ser lido e entendido facilmente, enquanto que um contrato inteligente não.

Enquanto um contrato tradicional define regras em torno de um acordo entre uma ou múltiplas partes, os contratos inteligentes vão além e garantem essas regras controlando a transferência de moeda ou ativos sob condições específicas. Contratos inteligentes e licenças tendem a decentralizar o modelo sob o aspecto de quem precisa ter crédito, e fazendo isso tende a eliminar

taxas intermediando serviços. [35]

Plataformas de blockchain como Ethereum e bitcoin possuem limitações devido a questões de segurança que restringem o acesso a dados externos (exemplo, preço, clima, localização, etc.) que são levados em consideração no desempenho contratual e melhores formas de pagamento para as partes envolvidas. Por esse motivo, links confiáveis para fontes de dados externos são necessários para que alguns contratos inteligentes possam ser executados.

Por conta disso, os contratos inteligentes podem ser divididos em determinísticos e não-determinísticos.

Contratos inteligentes determinísticos são códigos de contratos inteligentes que não dependem de informação externa além dos dados presentes no blockchain onde serão executados e irão funcionar efetivamente. Em outras palavras, a rede blockchain facilita para que o contrato inteligente tenha informação suficiente para tomar decisões. Exemplo: loteria peer-to-peer : os fundos estão na rede blockchain e os números são randomicamente gerados pelo código de contratos inteligentes. Ao fim os fundos são transferidos para a conta do ganhador via endereço na rede blockchain. [36]

A rede não-determinística de contratos inteligentes auxilia códigos de contrato inteligente que não possuem informação suficiente para a tomada de decisão, portanto fonte externa de dados se faz necessária, usualmente conhecida como "Oracle" no domínio da ciência da computação. Exemplo: decisões sobre valores baseados no comportamento humano, eventos (preço que sobe ou desce) ou predições. [36]

No domínio da ciência da computação, Oracles são conhecidos por suas habilidades em fornecer informações externas que o sistema por si só não é capaz de capturar. Quando aplicado a contratos inteligentes, oracles agem como agentes programáveis que fornecem a um contrato inteligente dados que este necessita (inbound oracles) assim como também lança pagamentos ou informa ao sistema interno o que um contrato inteligente conclui (outbound oracle)[37]. Oracles armazenam os dados e apenas repassam o que é importante ou relevante para a execução de um contrato inteligente, garantindo segurança e alta privacidade da rede aumentando a eficiência.

É desejável que uma aplicação de contratos inteligentes apresente algum nível de não-determinismo e mesmo que seja, ainda sim reduz riscos de fraude.

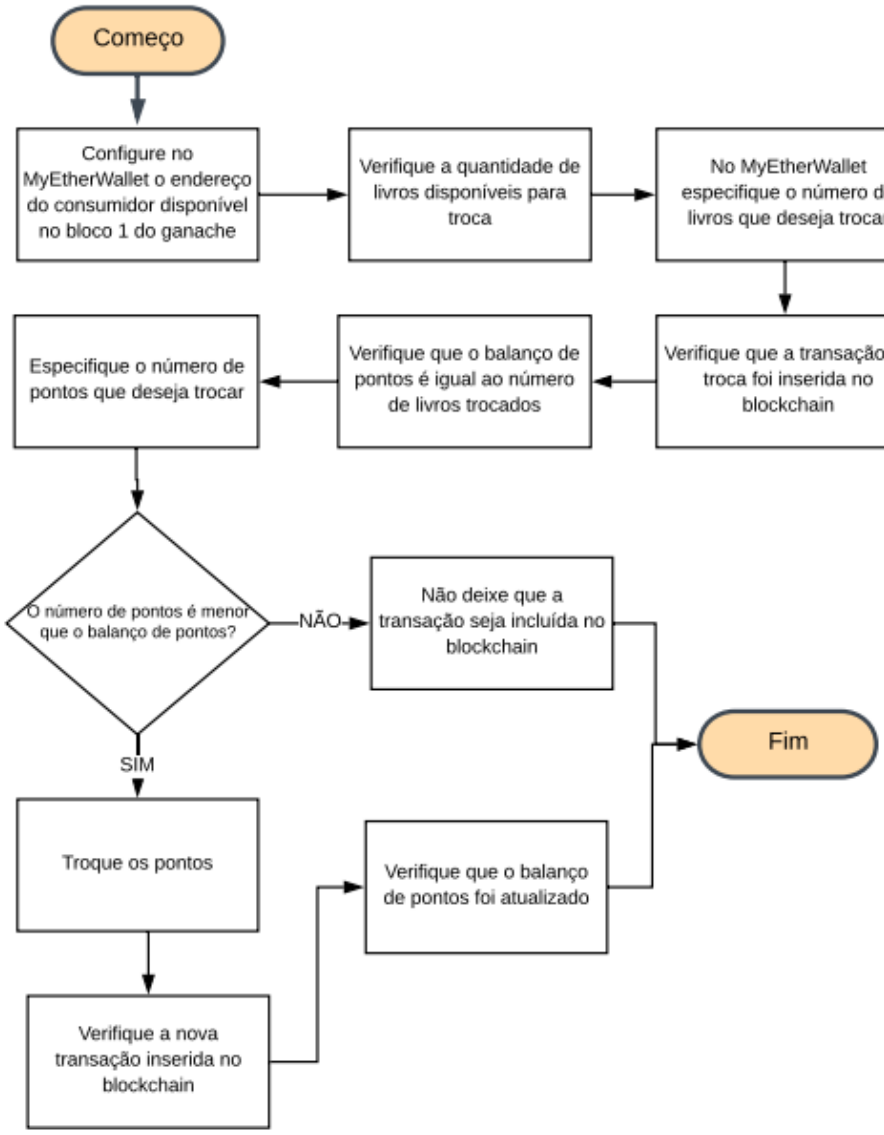
Embora o conceito de contratos inteligentes ainda esteja em sua fase inicial, existe um potencial visível que irá romper com o sistema legal e bancário atual, não se pode afirmar que tal ferramenta irá substituir documentos legais. Embora bancos e advogados realizem tarefas

repetitivas ainda se pagam altas taxas para que estes serviços sejam executados. algumas destas tarefas podem ser automatizadas proporcionando economia de tempo e recursos financeiros para a população.

4.1 Simulação de um Contrato Inteligente – Sistema de troca de livros

O diagrama da figura 18 apresenta uma representação visual dos passos seguidos durante o processo de simulação e teste do contrato inteligente desenvolvido.

Figura 18-Diagrama de atividades do contrato inteligente desenvolvido

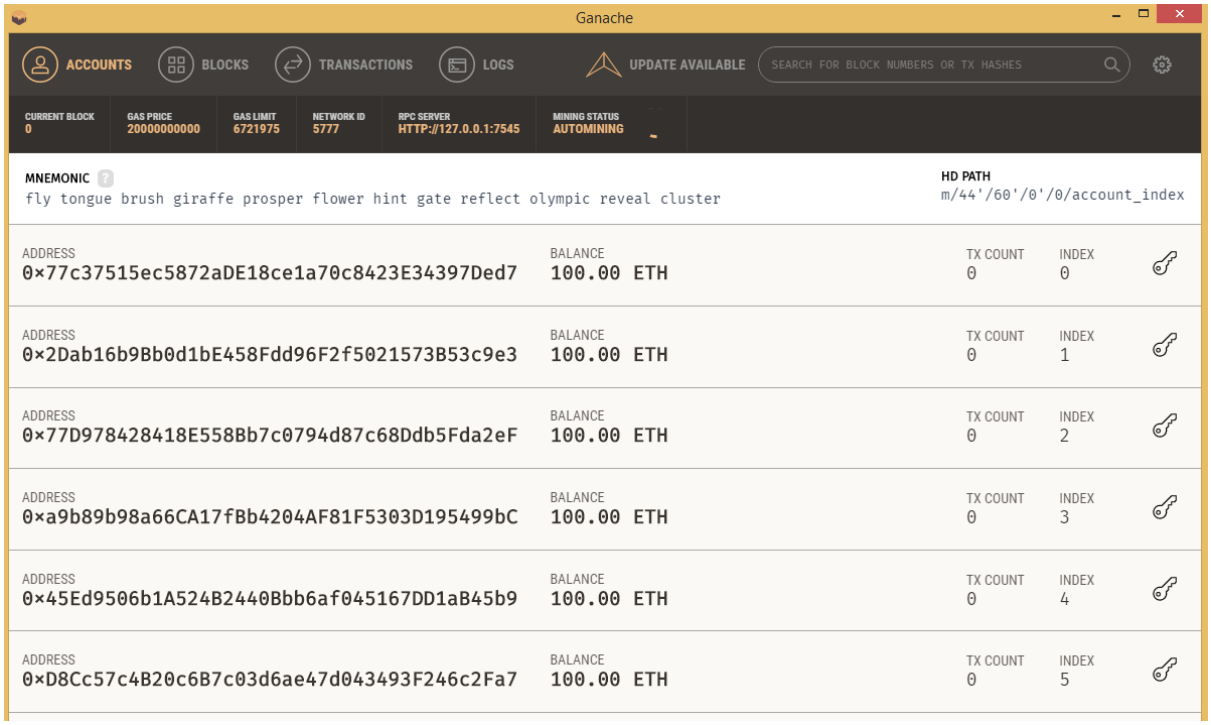


Para a simulação e implantação de contratos, foi escolhido o blockchain pessoal voltado ao desenvolvimento da Etheteum, a escolha ocorreu devido à alta disponibilidade de recursos já existentes para esta aplicação. Foi utilizada a aplicação Ganache para desktop [38]. O contrato foi

construído utilizando o Solidity como linguagem de programação e foi compilado utilizando o compilador solc.

O Blockchain virtual do ganache configura inicialmente dez endereços Etheruem com chaves privadas, conforme mostra figura 19. Não existe o conceito de mineração por conta a cada transação surge um novo bloco como confirmação, possibilitando o desenvolvimento interativo durante a implantação do contrato.

Figura 19-Tela Inicial da aplicação Ganache - Endereços Etheruem disponibilizados



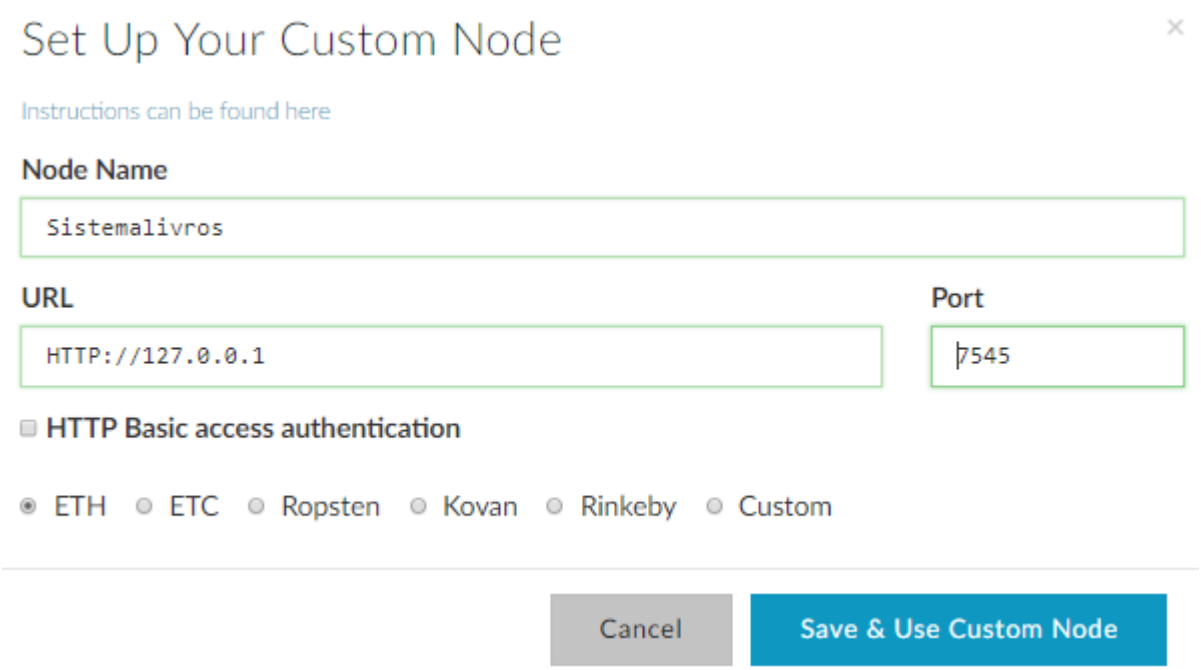
O Solidiy for escolhido por ser uma linguagem orientada ao desenvolvimento de contratos, utiliza linguagem de alto nível para implementação dos mesmos, influenciada em C++, Python e JavaScript. Através do Solidity foi possível simular um sistema de troca de livros, sendo possível desenvolver muitos outros contratos além disso. [39]

Para a interação com o blockchain foi utilizada a interface aberta e gratuita MyEtherWallet, que permite a interação direta com o blockchain ao mesmo tempo em que dá ao usuário o controle total das chaves e recursos disponíveis para as operações sejam elas de troca ou venda.

Ao criar uma conta no MyEtherWallet é gerado um conjunto criptográfico de números, endereços de chave pública e chave privada, gerenciada pelo usuário em seu browser. O envio da chave privada só pode ser feito pelo usuário que a detém, e caso isto ocorra o usuário que recebe obterá controle total da conta de quem o enviou. [40]

Para configurar o Nó, foi utilizado o mesmo endereço URL do Ganache conforme mostra a figura 20.

Figura 20- Estabelecendo conexão com o Ganache a partir do endereço do bloco.



A imagem 21 apresenta o retorno obtido dada a conexão estabelecida com o blockchain .

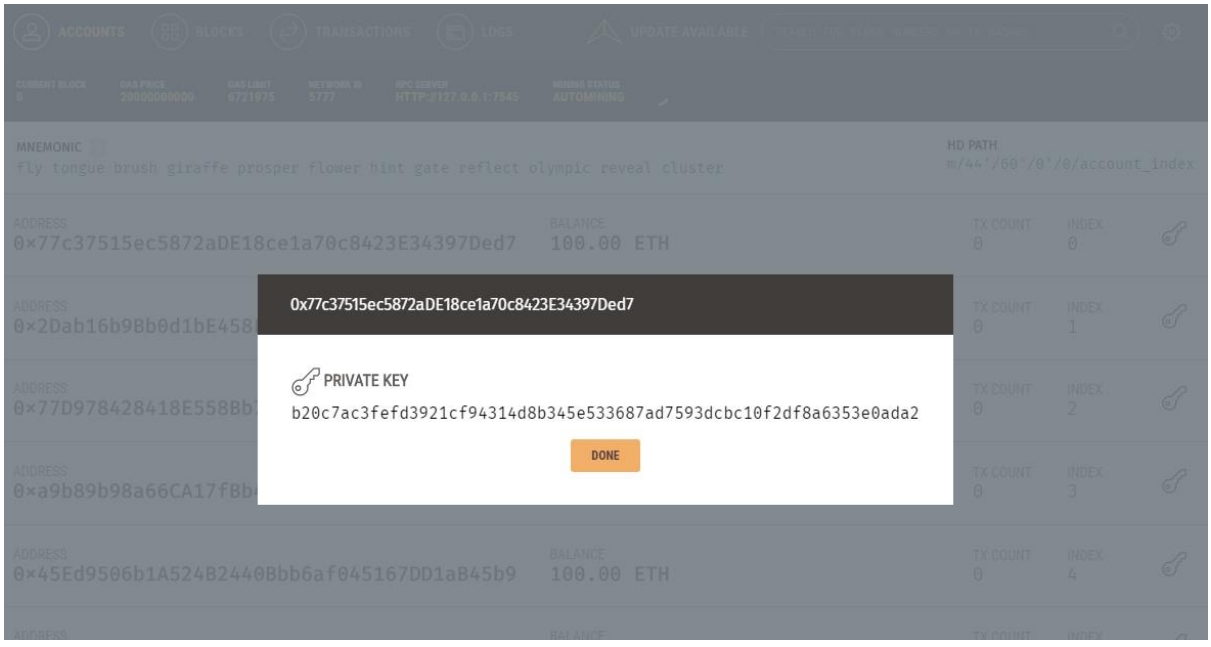
Figura 21- Resultado da conexão com o Blockchain

Após a construção do contrato via Solidity a implementação foi realizada através da plataforma MyEtherwallet. O campo Byte Code preenchido foi extraído do resultado final da compilação no Solidity.

Figura 22- Implementação do Contrato Inteligente desenvolvido e complidado via Solidity.

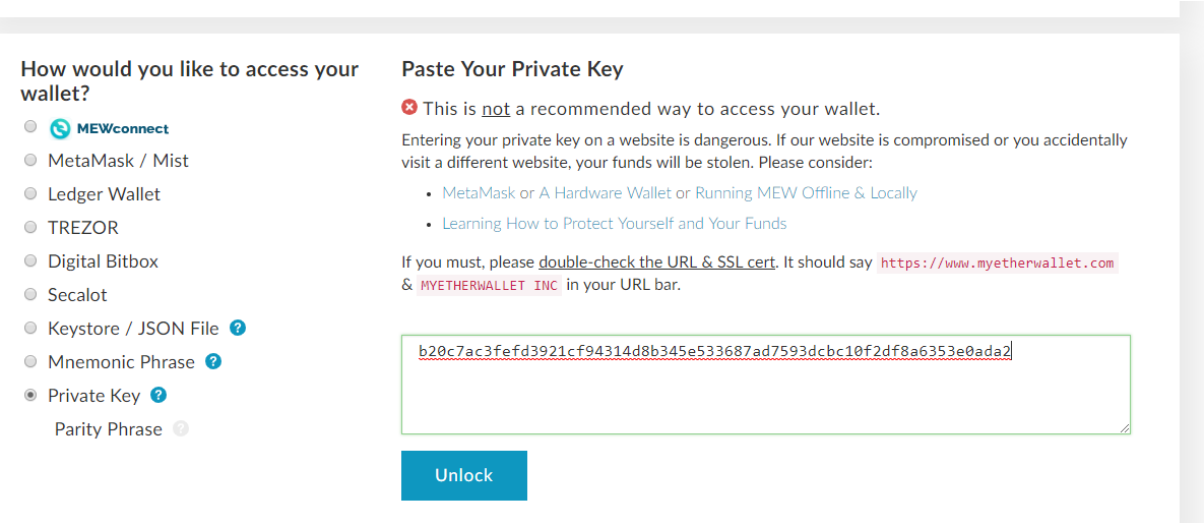
Para acesso à carteira digital onde os recursos foram alocados, foi obtida a chave privada, disponível nos endereços do Ganache conforme ilustra a figura 22.

Figura 23- Chave privada do endereço 1 do Ganache



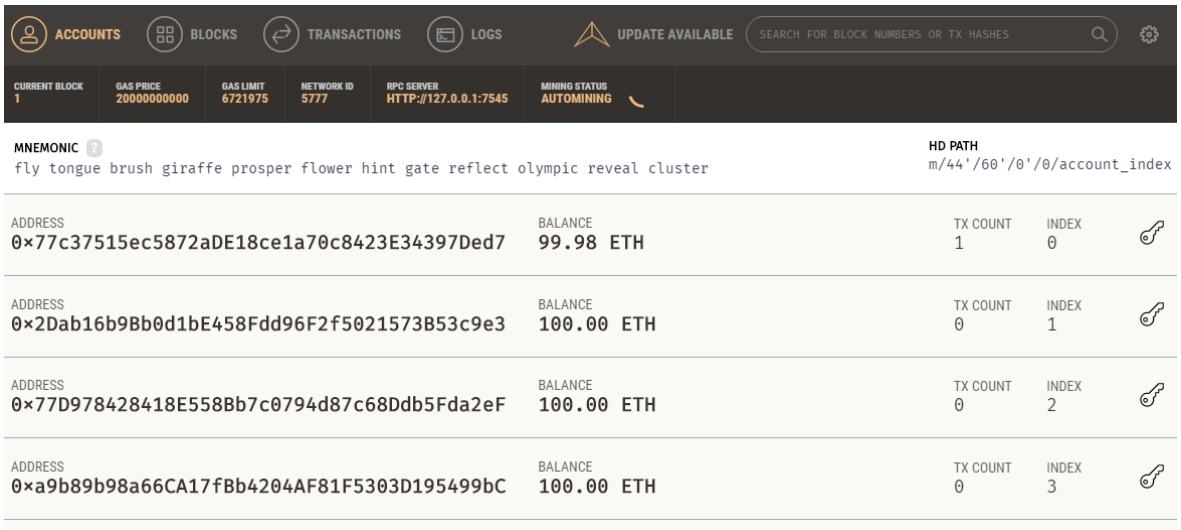
A chave privada obtida foi utilizada para configurar o acesso à carteira conforme a figura 23.

Figura 24- Configurando acesso à carteira a partir da chave provada obtida via Ganache.



Após assinatura da transação um novo bloco é minerado e a opção para implantação do contrato é disponibilizada conforme ilustra a figura 25

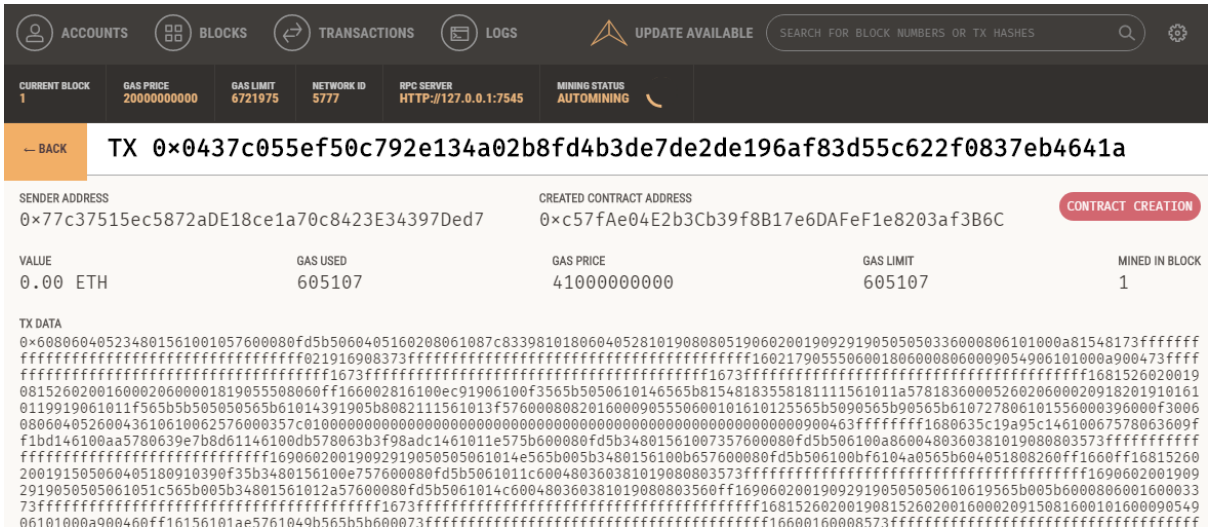
Figura 25- Inserção da transação no Blockchain



É possível notar o incremento na variável TX COUNT e o decremento no valor do campo balance, para que a transação se concretizasse e o contrato fosse implementado foram gastos 0,02 ETH, o que corresponde à R$ 14,78 reais, muito mais vantajoso em relação ao método clássico.

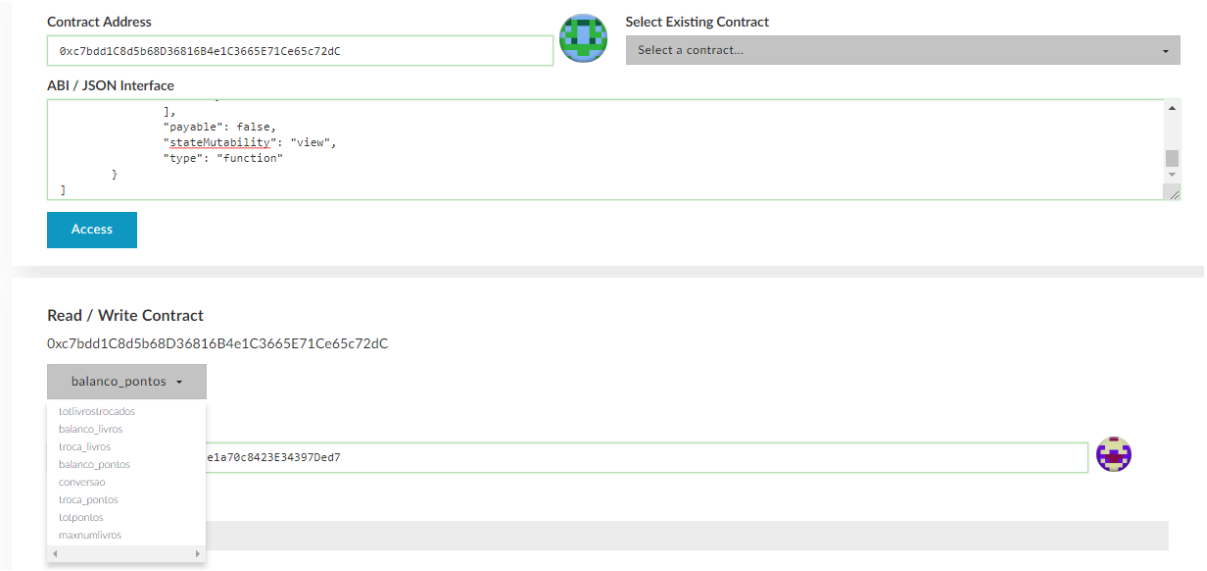
Além disso é possível verificar a adição do novo bloco, detalhes da transação e da criação de um novo contrato.

Figura 26- Detalhes do bloco inserido após o processamento da transação



Dado que o contrato foi integrado ao blockchain conforme a imagem 26 a interação se deu a partir aba de interação de contratos disponível no MyEtherWallet, foi obtido o endereço do contrato criado via Ganache e o ABI/ Json interface via Solidity apresentado na figura 27.

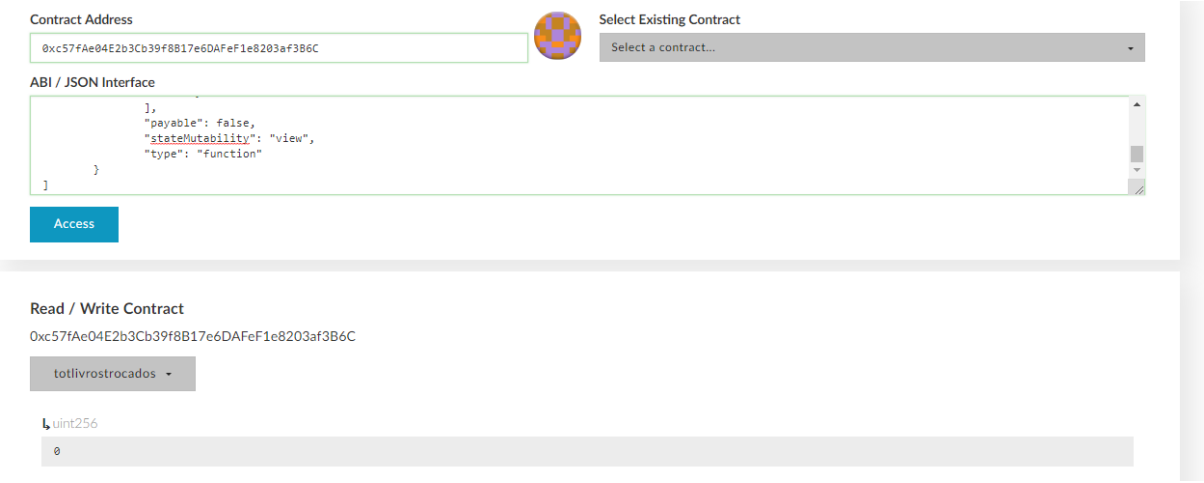
Figura 27-Interação com o contrato via MyEtherWallet.



Foi simulado um sistema de troca de livros, embora já existem alguns sites atuais com sistema parecido, como a container cultura, mas a simulação ocorreu com o intuito de verificar o uso da tecnologia por meio do blockchain. O contrato permite que o usuário troque seus livros e a cada livro trocado o usuário ganha um ponto, que pode ser trocado por qualquer outra recompensa. [41]

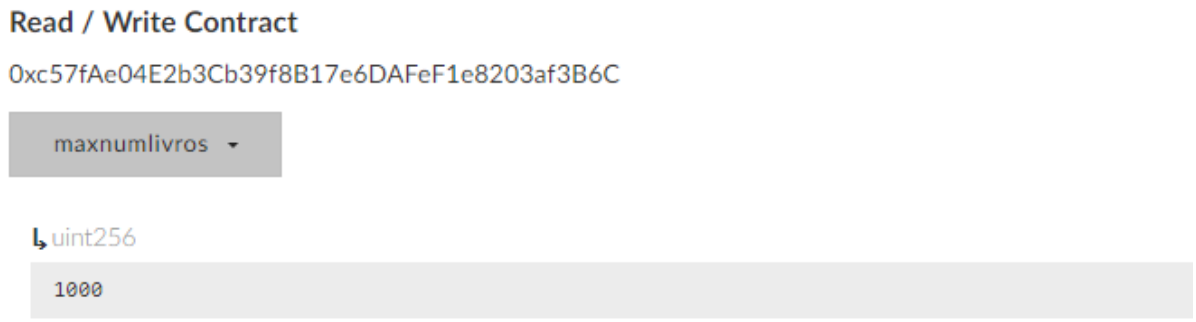
A figura 26 mostra o início da execução, com nenhum livro trocado.

Figura 28- Início da execução com variáveis inicializadas



A quantidade de livros disponíveis da rede foi configurada via Solidity com valor de 1000, conforme ilustra a figura 28.

Figura 29- Quantidade de recurso disponível na rede

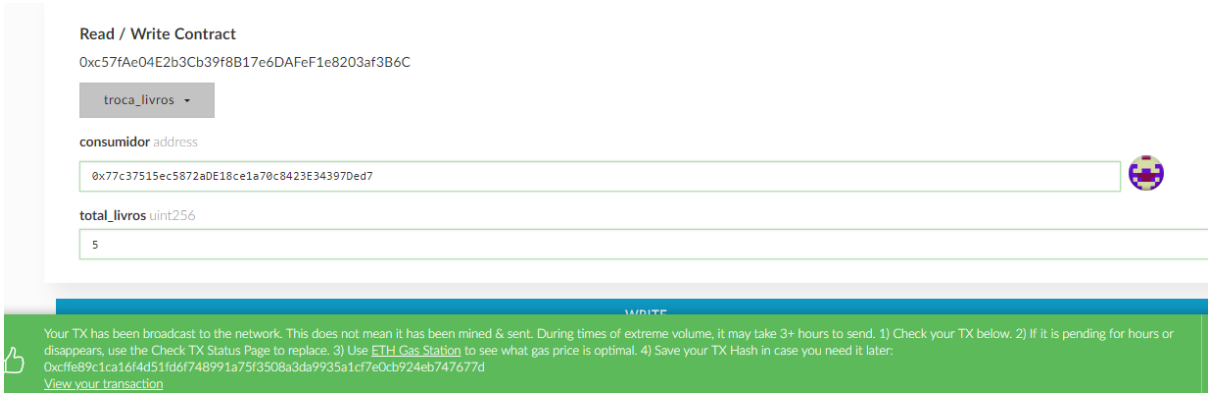


Se houver uma requisição para trocar mais livros que o total disponível a troca não ocorre devido ao tratamento realizado via código através da função *modifier***,** função executada para controlar o comportamento de um contrato inteligente, restringindo uma ação dada condição. Neste caso, verificando se a quantidade especificada para troca é menor que a quantidade de livros totais existentes na rede.

A figura 30 apresenta o resultado de uma solicitação que não pode ser efetuada e a figura 31 mostra uma solicitação de troca com a quantidade abaixo do total disponível.

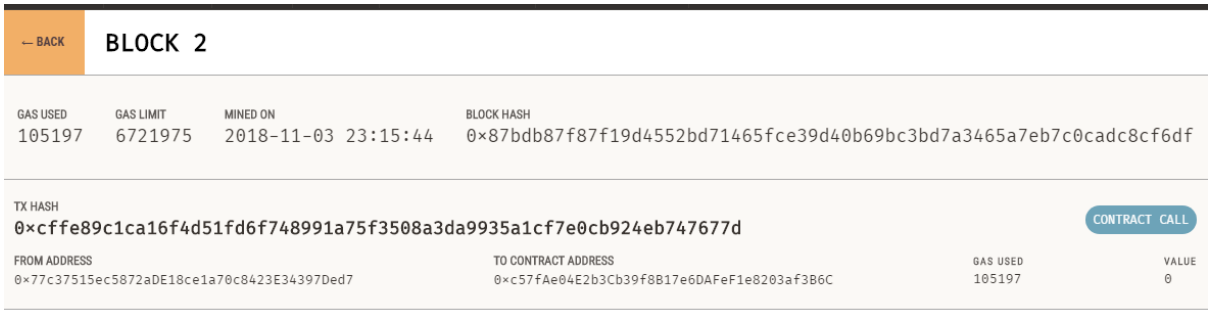
Figura 30- Aplicação do modifier restringindo uma solicitação feita pelo usuário

Figura 31- Inclusão de uma transação para atender a quantidade de recurso solicitada



O campo consumidor recebe o endereço do remetente (sender addres) disponível no Ganache, o envio da transação gera a criação de um novo bloco no blockchain, conforme a figura 32.

Figura 32- Bloco criado a partir da inserção da solicitação processada.



A figura 33 mostra o balanço de livros trocados por um determinado usuário enquanto a figura 34 apresenta a quantidade de pontos resultante da troca dos livros.

Figura 33- Saldo de recursos trocados após a solicitação ser processada

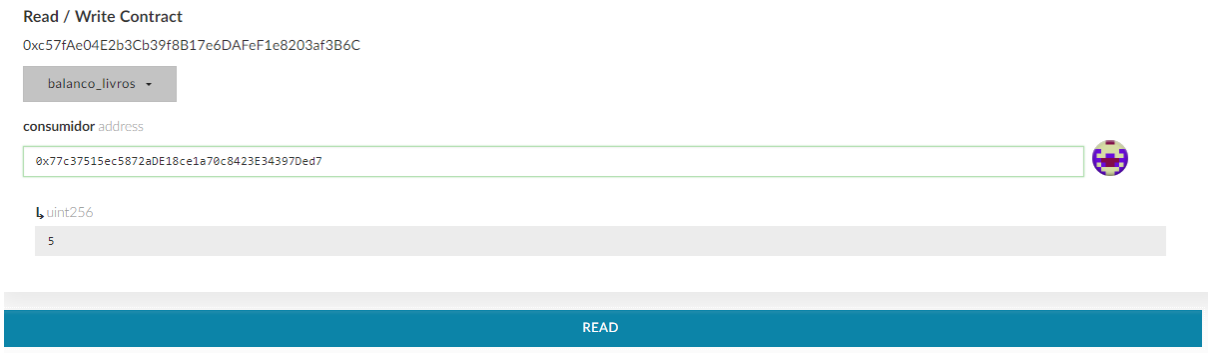
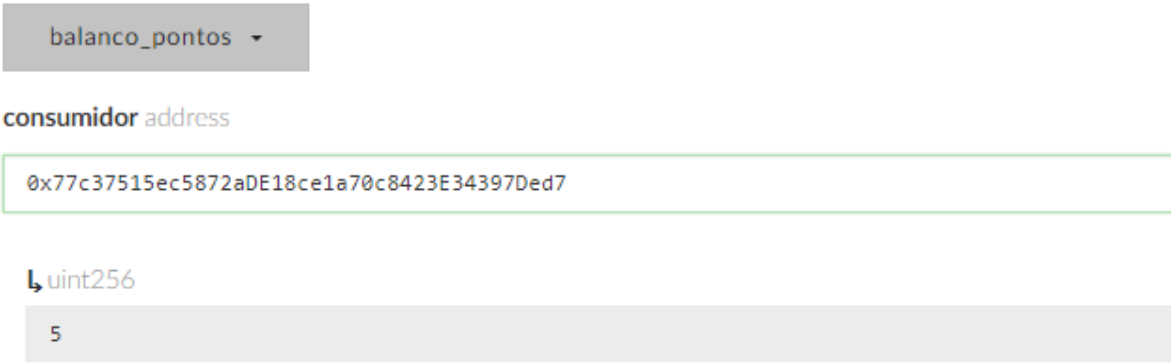
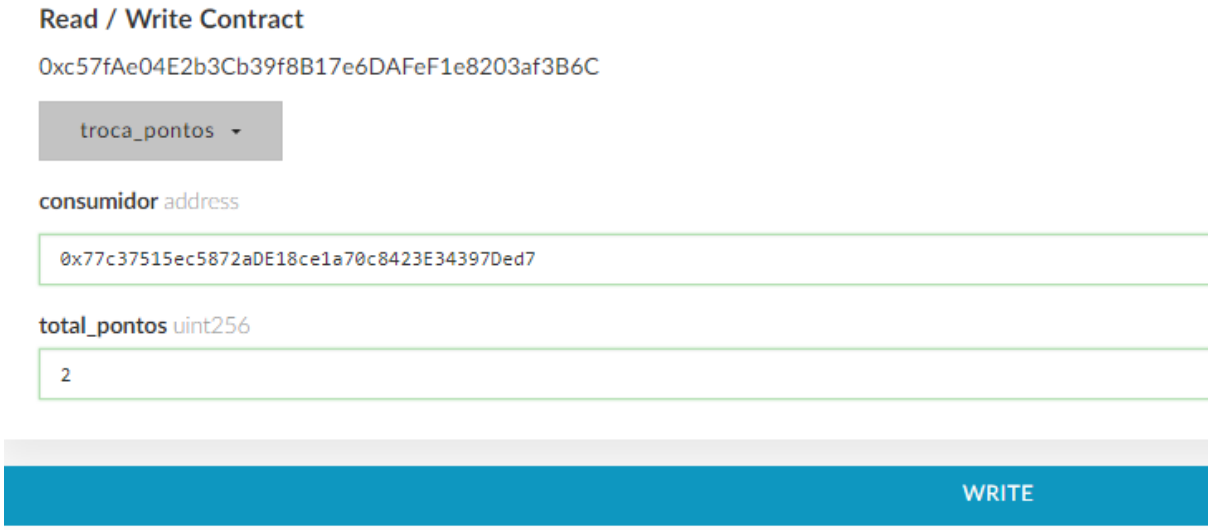


Figura 34- Quantidade de pontos recebidos dada uma solicitação.



Caso o usuário queira trocar seus pontos por alguma outra recompensa e tenha quantidade de pontos necessária para troca, uma nova transação é adicionada ao bloco, como por exemplo a transação da figura 35.

Figura 35- Troca de pontos por uma recompensa



E o saldo de pontos é atualizado conforme figura 36.

Figura 36- Atualização do saldo de pontos após a troca

Este trabalho não se deteve a nenhuma ideia empreendedora específica, mas sim no funcionamento e uso da tecnologia, os pontos por exemplo podem ser revertidos em descontos em estabelecimentos parceiros, diminuindo o saldo de pontos de quem decidiu trocar. Importante notar aqui que as aplicações dos contratos inteligentes vão muito além de um sistema de troca de livros, podendo se estender a serviços financeiros, compra e venda de casas etc.

O código é capaz de formalizar negociações definindo regras restritas e consequências da mesma forma de um documento tradicional, de uma maneira mais rápida, barata e menos burocrática.

5. CONCLUSÃO

Através deste trabalho foi possível obter um entendimento maior em relação a tecnologia do blockchain, foi realizada uma contextualização que abrange desde o seu surgimento até conceitos por trás de seu funcionamento. O crescente uso de dispositivos móveis e o aumento significativo no número de diferentes tipos de transações reforçam a importância em encontrar tecnologias alternativas para garantir transparência, segurança, integridade e eficiência nas transações.

Todo o conhecimento teórico levantado foi visto em prática através da construção do blockchain desenvolvido utilizando a linguagem de programação Python. A construção do blockchain partiu de princípios básicos e pode ser aprimorada conforme crescem as atualizações e aplicações da tecnologia. Muito embora o futuro deste sistema computacional descentralizado ainda esteja incerto e a abordagem presente neste trabalho tenha sido simples, é possível esperar um futuro mais transparente, integro, autêntico, confiável e menos burocrático no que depender do uso da tecnologia.

Por meio deste trabalho foi possível concluir que o blockchain deve levar à transformação digital por meio da automação de processos e codificação de contratos. Dentre o potencial da tecnologia do blockchain, foi possível entender o funcionamento e codificar um contrato inteligente simples verificando que o protocolo de computador auto executável foi capaz de obtendo as informações, processá-las e tomar as devidas ações de acordo com as regras estabelecidas.

Atualmente os contratos inteligentes podem ser codificados em qualquer blockchain. Neste trabalho foi utilizado o Ethereum, que fornece uma capacidade ilimitada de processamento e criação de aplicações. Os contratos inteligentes não requerem um intermediário centralizador e podem reduzir custos de transações e preços para o consumidor final, aumentando a liberdade de negócios e garantindo o direito à livre troca e ao livre comércio em diversas áreas como: Logística e suprimentos, conteúdo protegido por direitos autorais, eleições, Internet das coisas, leis de propriedade, setores imobiliários, serviços financeiros entre outros. [40]

Por fim, é possível constatar que contratos inteligentes se tornarão uma ferramenta útil para impulsionar negócios, diminuir os conflitos e a necessidade de procedimentos legais tradicionais, conferindo rapidez e segurança para empresas e clientes.

Como ponto de melhoria, as requisições HTTP construídas neste trabalho podem passar a considerar apenas a requisição POST e não GET, pois via programação, os dados postados pelo método GET podem ser manipulados, o que compromete a segurança da rede.

6. BIBLIOGRAFIA

[1]S. WILSON, "Rising Tide: Exploring pathways to growth in the mobile semiconductor insdustry," Deloitte University Press, 2013.

[2]J. MACAULAY, L. BUCKALEW, and G. CHUNG, "Internet of Things in Logistics," Troisdorf, 2015.

[3]GUDKOVA,D. Spam: quarterly highlights .2016. Disponível em <https://securelist.com/spam-and-phishing-in-q3-2016/76570/> Acessado em: 2 de Setembro de 2017.

[4] BARABBA, V.P. Revisiting Plato's cave: business design in an age of uncertainty. In TAP SCOTT, D. (ed.). Blueprint to Digital Economy. New York: McGraw-Hill, 1998, 410p.

[5]BITCOIN SERIES 24: The Mega-Master Blockchain List .Ledra Capital .Disponível em: <http://ledracapital.com/blog/2014/3/11/bitcoin-series-24-the-mega-master-blockchain-list> Acessado em: 5 de Julho de 2017.

[6] HARBER S., STORNETTA W. S, How to Time-Stamp a Digital Document, 1990.

[7] NAKAMOTO S. Bitcoin: a peer-to-peer electronic cash system. 9. 2018 disponível em www.Bitcoin.Org Acessado em 15 de Fevereiro de 2018.

[8] NARAYANAN, A.; BONNEAU, J., FELTEN. E.; MILLEE, A.; GOLDFEDER, S. (2016). Bitcoin and cryptocurrency technologies: a comprehensive introduction. Princeton: Princeton University Press.

[9] TEL, G. Criptography in Context, 2008. Acessado em 10 de Agosto de 2018. Disponível em: <https://www.staff.science.uu.nl/~tel00101/liter/Books/CrypCont.pdf>

[10] Disponível em: https://en.wikipedia.org/wiki/Cryptographic_hash_function Acessado em 15 de julho de 2018.

[11] Distributed Ledger. Disponível em: <https://www.investopedia.com/terms/d/distributed-ledgers.asp> Acessado em 18 de julho de 2018.

[12]VITALIK, B. The meaning of Decentralization, 2017.

Disponível em: < https://medium.com/@VitalikButerin/the-meaning-of-decentralization-a0c92b76a274>. Acessado em: 15 de Agosto de 2018

[13]VAIDYA, K. Bitcoin's implementation of Blockchain, 2016 Disponível em: <https://medium.com/all-things-ledger/bitcoins-implementation-of-blockchain-2be713f662c2> Acessado em: 25 de Agosto de 2018.

[14] EREMENKO, K. How does Bitcoin / Blockchain Mining work? . 2018 Disponível em: <https://medium.com/swlh/how-does-bitcoin-blockchain-mining-work-36db1c5cb55d>. Acessado em 26 de Agosto de 2018.

[15]ASOLO, B. Full Node and Lightweight Node, 2018. Disponível em: <https://www.mycryptopedia.com/full-node-lightweight-node/>Acessado em 26 de Agosto de 2018.

[16]WEISZFLOG, W. Michaelis Dicionário Brasileiro da Língua Portuguesa . [S.l.]: Melhoramentos, 2018.

[17] KONSTANTOPOULOS, G. Understanding Blockchain Fundamentals, Part 1: Byzantine Fault Tolerance. 1. 2017.

Disponível em < https://medium.com/loom-network/understanding-blockchain-fundamentals-part-1-byzantine-fault-tolerance-245f46fe8419>. Acessado em: 26 de Agosto de 2018.

[18] STEVENS, A. Understanding the Byzantine Generals' Problem (and how it affects you). 1. 2018. Disponível em: < https://medium.com/coinmonks/a-note-from-anthony-if-you-havent-already-please-read-the-article-gaining-clarity-on-key-787989107969 >. Acessado em 28 de Agosto de 2018.

[19]BLOCKGEECKSs. Basic Primer: Blockchain Consensus Protocol. 2018. Disponível em: <https://blockgeeks.com/guides/blockchain-consensus/>. Acessado em: 10 Setembro de 2018.

[20] Blockgeecks. Proof of Work vs Proof of Stake: Basic Mining Guide. 2018. Disponível em: <https://blockgeeks.com/guides/proof-of-work-vs-proof-of-stake/>. Acessado em: 05 de Setembro de 2018.

[21] JIMI, S. Blockchain: how mining works and transactions are processed in seven steps. 2018. Disponível em: <https://medium.com/coinmonks/how-a-miner-adds-transactions-to-the-blockchain-in-seven-steps-856053271476>. Acessado em: 06 de setembro de 2018.

[22] TANYA. Explaining bitcoin transaction fees. Disponível em: <https://support.blockchain.com/hc/en-us/articles/360000939883-Explaining-bitcoin-transaction-fees>. Acessado em: 10 de Setembro de 2018.

[23] TANYA. Transaction fees. Disponível em: <https://support.blockchain.com/hc/en-us/articles/360000939903-Transaction-fees>. Acessado em: 12 de Setembro de 2018.

[24] SUN, F. UTXO vs Account/Balance Model. 2018. Disponível em: <https://medium.com/@sunflora98/utxo-vs-account-balance-model-5e6470f4e0cf>. Acessado em: 05 de Outubro de 2018.

[25] SHARMA, T. K.. How Does Blockchain use public key Cryptography?.2018. Disponível em: <https://www.blockchain-council.org/blockchain/how-does-blockchain-use-public-key-cryptography/>. Acessado em: 28 de Outubro de 2018.

[26] ANACONDA. 2018. Disponível em: <https://docs.anaconda.com/anaconda/>. Acessado em: 20 de Setembro de 2018.

[27]RONACHER, A. Flask: User's Guide. Disponível em: <http://flask.pocoo.org/>. Acessado em: 26 de Setembro de 2018.

[28] POSTMAN. 2018. Disponível em: <https://www.getpostman.com/>. Acessado em: 20 de Setembro de 2018.

[29] HART O, MOORE J (2002) Contracts as reference points. Q J Econ CXVII:1–48

[30] The Law Handbook: What is a contract? Available online at: http://www.lawhandbook.org.au/07_01_01_what_is_a_contract/. Acessado em: 20 de Novembro 2017

[31] SZABO N. (1997) Formalizing and securing relationships on public networks.

[32] CASSANO J, What are smart contracts? Cryptocurrency's Killer App. Available online at: http://www.fastcompany.com/3035723/app-economy/smart-contracts-could-be-cryptocurrencys-killer-app. Acessado em: 15 Jul 2017.

[33] IDELBERGER F, GOVERNATORI G, RIVERET R, SARTOR G (2015) Evaluation of logic-based smart contracts for blockchain systems. In: Alferes JJ, Bertossi L, Governatori G, Fodor P,Dumitru R (eds) Rule technologies. research, tools, and applications 10th international symposium, RuleML 2016, Stony Brook, NY, USA, July 6–9, 2016. Volume 9718 of the series lecture notes in computer science, pp 167–183, Springer, Switzerland

[34] GREENSPAN G, Why many smart contract use cases are simply impossible. Available online at:http://www.coindesk.com/three-smart-contract-misconceptions/. Acessado em: 8 de Agosto de 2017.

[35] TROY S. What is a smart contract and what's it good for? Available online at: http://searchcio.techtarget.com/feature/What-is-a-smart-contract-and-whats-it-good-for. Acessado em: 8 de Agosto de 2017.

[36] LEWIS A, A gentle introduction to smart contracts. Available online at: https://bitsonblocks.net/2016/02/01/a-gentle-introduction-to-smart-contracts/.

[37] SmartContract: Smart contract oracles. Disponivel em: htttp://about.smartcontract.com Acessado em: 20 de Setembro de 2018.

[38] GANACHE. 2018. Disponível em: <https://truffleframework.com/ganache>. Acessado em: 22 de Setembro de 2018.

[39] SOLIDITY. 2018. Disponível em: <https://solidity.readthedocs.io/en/v0.4.25/>. Acessado em: 26 de Setembro de 2018.

[40] Disponível em: <https://coincenter.org/entry/what-are-smart-contracts-and-what-can-we-do-with-them>. Acessado em: 10 de Novembro de 2018.

[41]CONTAINER CULTURA. 2018 Disponível em: https://www.containercultura.com.br Acessado em: 03 de Dezembro de 2018

APENDICE I

```
**Código para criação do sistema de troca de livros

//v.0.4.25

//contratos
pragma solidity ^0.5.0;

contract testelivros {

//definindo variáveis públicas acessadas por pessoas que fazem ou não parte do contrato

//número total de livros disponíveis para troca
   uint public maxnumlivros = 1000;

//conversão de fichas para reais
   uint public conversao = 1;

//Variavel para mudar o valor, livros comprados pelos consumidores até que não hajam mais
livros disponíveis
   uint public totlivrostrocados = 0;
    uint public totpontos = 0;


//variável privada é acessada somente por quem faz parte do contrato
//mapping array com variável de entrada que retornará o endereço dos consumidores

   mapping(address => uint) balancolivros;
   mapping(address => uint) balancopontos;

//modifier vai checar se a troca poderá ser realizada
   modifier pode_trocar_livros (uint total_livros){
      require(total_livros*conversao + totlivrostrocados <= maxnumlivros);
      _;
   }


      modifier pode_trocar_pontos (uint total_pontos){
      require( total_pontos>0 && total_pontos< totpontos);
      _;
   }

   //obtendo o total de pontos de um consumidor
   function balanco_pontos (address consumidor) external view returns (uint){
   // usa o mapping para pegar o endereco como argumento e retornar o balanco
      return balancopontos[consumidor];
   }

   //obtendo o total de um consumidor
   function balanco_livros (address consumidor) external view returns (uint){
   // usa o mapping para pegar o endereco como argumento e retornar o balanco
```

```
        return balancolivros[consumidor];
    }

    //trocando livros
    //primeiro aplica o modifier para ver se a troca pode ser realizada
    function troca_livros(address consumidor, uint total_livros) external
    pode_trocar_livros(total_livros){
        uint livros_trocados=total_livros;
        balancolivros[consumidor] += livros_trocados;
        balancopontos[consumidor]=balancolivros[consumidor]*conversao;
        totlivrostrocados += livros_trocados;
        totpontos += livros_trocados;

    }

    function troca_pontos(address consumidor, uint total_pontos) external
    pode_trocar_pontos(total_pontos){
        uint pontos_trocados=total_pontos;
        balancopontos[consumidor] -= total_pontos;
        totpontos -= pontos_trocados;


    }









}
```