UNIVERSIDADE FEDERAL DO CEARÁ
CENTRO DE TECNOLOGIA
DEPARTAMENTO DE TELEINFORMÁTICA
ENGENHARIA DE COMPUTAÇÃO

ANDERSON LEANDRO DE MELO MARQUES ROCHA

UMA ARQUITETURA PARA SISTEMAS DE RASTREABILIDADE EM CADEIAS LOGÍSTICAS USANDO BLOCKCHAIN

FORTALEZA
2023

ANDERSON LEANDRO DE MELO MARQUES ROCHA

UMA ARQUITETURA PARA SISTEMAS DE RASTREABILIDADE EM CADEIAS LOGÍSTICAS USANDO BLOCKCHAIN

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Computação do Centro de Tecnologia da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Engenharia de Computação.

Orientadora: Prof. Dra. Atslands Rego da Rocha

FORTALEZA
2023

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Sistema de Bibliotecas
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

R571a Rocha, Anderson Leandro de Melo Marques.
Uma arquitetura para sistemas de rastreabilidade em cadeias logísticas usando Blockchain / Anderson Leandro de Melo Marques Rocha. – 2023.
61 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Centro de Tecnologia, Curso de Engenharia de Computação, Fortaleza, 2023.
Orientação: Profa. Dra. Atslands Rego da Rocha.

1. Blockchain. 2. Rede. 3. Rastreabilidade. 4. Logística. 5. Hyperledger. I. Título.

CDD 621.39

ANDERSON LEANDRO DE MELO MARQUES ROCHA

UMA ARQUITETURA PARA SISTEMAS DE RASTREABILIDADE EM CADEIAS LOGÍSTICAS USANDO BLOCKCHAIN

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Computação do Centro de Tecnologia da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Engenharia de Computação.

Aprovada em:

BANCA EXAMINADORA

_____

Prof. Dra. Atslands Rego da Rocha (Orientadora)
Universidade Federal do Ceará (UFC)

_____

Prof. Dra. Nídia Glória da Silva Campos
Instituto Federal do Ceará (IFCE)

_____

Prof. Dr. Antonio Rafael Braga
Universidade Federal do Ceará (UFC) – Campus Quixadá

À minha família por todo suporte na minha criação. Obrigado por estarem ao meu lado nas minhas batalhas. Aonde cheguei na vida (e aonde chegarei) devo a eles.

AGRADECIMENTOS

Aos meus pais, Josefa Leandro de Melo e Valdemar Marques Rocha, por todo amor, carinho e apoio dado ao longo da jornada de minha vida, e também por todo o esforço e trabalho dedicados à minha criação e à me darem as melhores oportunidades de educação possíveis para mim.

A meu irmão, Nícolas Leandro de Melo Marques Rocha, pela parceria, amizade e amor que construímos juntos. Agradeço por encorajar o meu crescimento e sempre estar ao meu lado mesmo nos momentos mais difíceis.

Aos professores Prof. Dr. Flávio Rubens de Carvalho Sousa e Prof. Dr. Danielo Gonçalves Gomes por me darem as primeiras ideias e feedbacks para a construção da solução a ser apresentada neste artigo.

À Prof. Dra. Atslands Rocha pelos ensinamentos e aconselhamentos durante a minha graduação e por me orientar neste trabalho, sempre mostrando os melhores caminhos possíveis a percorrer e pelos valiosos feedbacks e valiosas e constantes sugestões de melhoria, sempre visando a excelência.

RESUMO

A rastreabilidade é fundamental para garantir a qualidade e segurança dos produtos, desde a sua fabricação e processamento, até o consumidor final. Esse acompanhamento da jornada do início ao fim constitui-se um grande desafio logístico, pois os agentes envolvidos em cada etapa da cadeia são muitas vezes independentes entre si, tornando o processo de verificação e auditoria entre as partes mais burocrático e complexo. Tendo em vista esse contexto, o objetivo principal deste trabalho é propor uma arquitetura para rastreabilidade de cadeias logísticas utilizando tecnologia *Blockchain*, de forma a mostrar que essa arquitetura consegue resolver muitos dos problemas das cadeias logísticas e também que possa servir como base para o desenvolvimento de outras soluções mais complexas e específicas do mesmo tipo. Selecionou-se a rede *blockchain* Hyperledger Fabric através de comparação entre Ethereum e Hyperledger. Para as simulações, foram usados os conceitos de virtualização por meio de *containers*, utilizando-se o Docker para as simulações dos experimentos. Foram conduzidos testes que buscaram mostrar o funcionamento da arquitetura desenvolvida a partir da criação de um modelo em pequena escala com containers Docker, validando a implementação da rede e a execução do contrato inteligente, com o correto registro das informações na *blockchain*, bem como a sua correta leitura. Os resultados desses testes demonstraram a viabilidade da arquitetura em pequena escala, validando, assim, os princípios básicos que permitem também o seu funcionamento em grande escala. Com base nesses testes realizados, conclui-se que a solução apresentada é viável, cumprindo com os objetivos inicialmente propostos.

Palavras-chave: *Blockchain.* Rede. Rastreabilidade. Logística. *Hyperledger. Docker.*

ABSTRACT

Traceability is essential to ensure the quality and safety of products, from manufacturing and processing to the final consumer. This monitoring of the supply chain from beginning to end constitutes a major logistical challenge, as the agents involved in each stage of the chain are often independent of each other, making the verification and auditing process between the parties more bureaucratic and complex. In this context, the main objective of this work is to propose an architecture for the traceability of logistic chains using Blockchain technology in order to show that this architecture can solve many of the problems of logistic chains and also that it can serve as a basis for the development of other more complex and specific solutions of the same type. We selected the Hyperledger Fabric blockchain network by comparing Ethereum and Hyperledger. We used concepts of virtualization using containers and Docker for the simulations of the experiments. We conducted tests to show the correct functioning of the architecture developed. From creating a small-scale model with Docker containers to validating the implementation of the network and the execution of the smart contract, with the correct registration of information in the blockchain, as well as its correct reading. The results of these tests demonstrated the viability of the architecture on a small scale, thus validating the basic principles that also allow its operation on a large scale. Based on our tests, we concluded that the presented solution is viable, fulfilling the initially proposed objectives.

Keywords: Blockchain. Network. Traceability. Logistics. Hyperledger. Docker.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ESG – *Environment, Social e Governance*

CNPJ – *Comprovante Nacional de Pessoa Jurídica*

B2B – *Business to Business*

B2C – *Business to Consumer*

RAM – *Random Access Memory*

SUMÁRIO

1 INTRODUÇÃO

O conceito de logística está associado à ideia de integração de várias atividades e funções, abrangendo toda a cadeia de suprimentos, desde a matéria-prima até a entrega do produto ao cliente [BALLOU, 2001]. Com relação à definição de logística, especificamente, pode-se afirmar que esta envolve o planejamento e a coordenação do fluxo de materiais, produtos em elaboração e produtos prontos da fonte ao consumidor final, e informações associadas a este fluxo de bens, atendendo às necessidades dos clientes dentro de um sistema integrado em vez de se adotar o gerenciamento isolado de uma série de atividades distintas [BORGES, 2006]. Toda empresa precisa lidar com os processos logísticos de seus produtos e serviços, e a otimização dessa logística garante à empresa vantagens de mercado e maior entrega de valor a seus clientes.

No entanto, existem diversos problemas associados às empresas envolvidas em setores de logística e de distribuição de produtos. Conforme comentado por [MATTOS, 2011], operações globalizadas trouxeram o aumento da distância média de transporte, maior dependência de parceiros na rede de suprimentos, busca pela redução dos níveis de estoque e o conflito de diferenças regionais, fatores que juntos tendem a aumentar a vulnerabilidade da cadeia logística. Ainda em [MATTOS, 2011], o grande desafio atual da gestão logística é estruturar uma cadeia com boa responsividade e flexibilidade, para responder a mudanças nas estratégias do negócio e impactos gerados por eventos externos, ao mesmo tempo em que se obtém ganhos através das cadeias enxutas. Cada vez mais torna-se importante conduzir um processo que garanta segurança para os envolvidos e qualidade para os produtos, buscando promover o equilíbrio entre oportunidades de ganhos e a redução de perdas, e, com isso, agregando mais valor ao produto final.

Existem dois requisitos essenciais para o bom funcionamento da cadeia de suprimentos: a rastreabilidade e a transparência. [ARANDA, 2022]. A rastreabilidade pode ser usada como um sistema de monitoramento e gestão eficaz com potencial para aumentar a transparência da informação [BEULENS *et al*, 2005], diminuindo a incidência de riscos e sinistros, reduzindo a enormidade e o impacto de tais incidentes, facilitando a identificação de produto(s) e/ou lotes afetados, identificar os problemas e os responsáveis [OPARA, 2003][WILSON and CLARKE, 1998]. Informações rastreáveis também podem fornecer aos consumidores informações valiosas sobre a origem dos produtos e podem contribuir para as crenças dos consumidores em relação aos atributos de credibilidade [KHER *et al*, 2010]. Os

sistemas que permitem a rastreabilidade ajudam a minimizar a produção e distribuição de produtos inseguros ou de baixa qualidade, minimizando assim o potencial de má publicidade, responsabilidade e reclamações. Além disso, o aumento da transparência em toda a cadeia de suprimentos pode reduzir a assimetria de informações [AUNG and CHANG, 2014].

Paralelamente a esses desafios, observamos o desenvolvimento de novas tecnologias que podem ser utilizadas dentro desse contexto. Uma das tecnologias que mais se desenvolvem atualmente é a *blockchain*. Surgida inicialmente no mercado financeiro das criptomoedas [NAKAMOTO, 2008], essa tecnologia vem sendo utilizada em outros setores para além deste. Na *blockchain*, os bens são representados de forma digital, em uma base de dados segura e confiável à qual múltiplos participantes podem ter acesso para escrita e/ou leitura de dados [GREVE, 2018]. Outra característica que auxilia o gerenciamento é que seus dados são registrados sequencialmente, permitindo que seja possível criar análises do uso dos recursos em uma série histórica [COSTA *et al*, 2021]. Devido a essas características, uma das possíveis aplicações dessa tecnologia é justamente no contexto de rastreabilidade de cadeias logísticas.

1.1 Motivação e Proposta de Solução

A gestão da logística e de todas as informações produzidas na cadeia, permite a avaliação dos pontos fortes e fracos da cadeia de fornecimento, auxilia a tomada de decisões, diminui os custos para manter um nível de operações próximo do ideal, aumenta da qualidade e a partir disso há aumento da competitividade, agregando valor [LUSTOSA et. al., 2008]. Porém, essa gestão fica prejudicada pelas deficiências e limitações dos métodos tradicionais de registro e compartilhamento de informações, dos quais podemos citar: falta de transparência, dificuldade de confirmação da autenticidade dos dados, lentidão nos processos de verificação e dificuldades na colaboração entre os diferentes participantes da cadeia. Além disso, a crescente demanda dos consumidores por informações detalhadas sobre a procedência e a qualidade dos produtos exige uma abordagem que seja mais visível e confiável de todas as etapas envolvidas.

Com isso, sistemas de rastreabilidade ganham muita importância nesse contexto abordado, para ajudar a melhorar essa gestão das cadeias logísticas. Conforme comentado por [SILVA, 2020], para a rastreabilidade efetiva, com garantia de dados verídicos e transparentes, é necessário haver com clareza a forma de estruturação e armazenamento dos dados. A

tecnologia *blockchain*, com suas características de transparência de informações para os participantes da rede, base de dados segura e imutável, e registro sequencial dos dados, oferece essa clareza do acompanhamento das etapas e segurança das informações registradas, sendo, portanto, uma solução promissora para aprimorar a rastreabilidade na cadeia logística e ajudar a resolver os problemas inerentes a essas cadeias.

1.2 Objetivos

1.2.1 Objetivo Principal

O objetivo principal deste trabalho é apresentar uma arquitetura de um sistema de rastreabilidade, que utiliza tecnologia *blockchain*, aplicada em uma cadeia logística simplificada. Busca-se mostrar de forma prática, com a construção de uma solução em pequena escala, a viabilidade do emprego dessa tecnologia e os benefícios que ela traz, dentre os quais se destacam transparência das informações, segurança dos dados e clareza do acompanhamento do produto ao longo das etapas da cadeia.

1.2.2 Objetivo Específico

Os objetivos específicos deste trabalho são:

i. Demonstrar a viabilidade do uso de *Blockchain* nesse tipo de arquitetura;

ii. Desenvolver uma estrutura pré-definida que possa servir como modelo para a construção de outras soluções mais complexas e específicas do mesmo tipo, economizando tempo e esforço e contribuindo para o desenvolvimento da área;

iii. Desenvolver um contrato inteligente que permita a execução de um fluxo de uma cadeia logística.

1.3 Procedimentos Metodológicos

Na primeira etapa deste trabalho, realizou-se um estudo sobre o uso de *Blockchain* em sistemas de Rastreabilidade de Cadeias Logísticas. Foi feito um levantamento das principais

blockchains usadas nesse tipo de solução e foi definida com qual tipo de rede seria usada neste trabalho.

Na segunda etapa, foi pensado como deveria ser uma cadeia logística simplificada e qual seria a lógica do fluxo de um produto nessa cadeia. Definir essa lógica foi importante para abstrair melhor como seria o fluxo da informação nessa cadeia e como deveriam ser as regras a serem obedecidas para termos essa abstração.

Na terceira etapa, foi desenvolvido o código do contrato inteligente, e realizou-se os primeiros testes do contrato e uma plataforma específica para isso.

Na quarta etapa, foi desenvolvida a rede com seus respectivos participantes, em uma escala reduzida e com processos de virtualização. O contrato inteligente foi instalado em cada um dos participantes dessa rede, criando a característica de descentralização tão comum desse tipo de tecnologia.

Por último, foi simulado a criação de produtos fictícios, que buscassem simular o fluxo desses produtos na cadeia logística. Foi analisada a execução do contrato inteligente nessa cadeia, bem como os componentes dessa rede interagiam entre si. Posteriormente, foram analisadas determinadas métricas dessa rede criada, para poder verificar se o desenvolvimento da solução conseguiu cumprir com os princípios básicos postulados teoricamente.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 Blockchain: visão geral

A *blockchain* é um livro-razão seguro, compartilhado e distribuído que facilita o processo de gravação e rastreamento de recursos sem a necessidade de confiança em uma autoridade central, permitindo que duas partes comuniquem-se e troquem recursos em uma rede, na qual decisões são tomadas pela maioria, e não por uma única entidade [SALMAN *et al,* 2019].

Embora tenha surgido no universo financeiro, dentro do contexto de descentralização proposto pelo Bitcoin [NAKAMOTO, 2008], as possibilidades de uso do *Blockchain* vão muito além dessa área. Atualmente, a tecnologia *blockchain* pode ser adotada por qualquer segmento que utiliza plataformas digitais para transações de informação, estando presente em

setores como logística, mercado de seguros, streaming, autenticação de documentos, dentre muitos outros.

Para utilizar o *Blockchain*, é preciso entender como é o seu funcionamento. O mecanismo permite a validação de dados com um processamento de rede distribuído em computadores privados e compartilhados através de uma nuvem. Ou seja, qualquer transação de um ativo online é feita apenas entre dois ou mais agentes, sem a intervenção de um servidor central.

Além disso, a informação só é trocada e validada após o consenso entre as duas ou mais partes envolvidas. Para garantir a convergência das visões que cada parte envolvida tem do estado da *blockchain* em um dado instante, a *blockchain* utiliza um mecanismo de consenso que permite que redes distribuídas ou descentralizadas tomem decisões unânimes quando necessário [SANKAR *et al*, 2017].

Como a rede não está centralizada em nenhum lugar e possui diversas camadas de segurança, invadi-la é extremamente difícil. Daí, cada vez mais a *blockchain* vem sendo usada para criar aplicações que exijam grande nível de segurança.

Também devido à natureza de bloco em cadeia, é possível verificar outros blocos anteriormente adicionados, tendo-se assim uma linha cronológica e um sistema de controle de versão.

2.1.1 Exemplos de Uso em Rastreabilidade de Cadeias Logísticas

O *blockchain* promete uma série de avanços na cadeia de distribuição, dando mais eficiência, visibilidade e transparência às operações. Cada vez mais redes de distribuição e logística vem usando esse tipo de tecnologia e garantindo assim mais vantagens competitivas aos seus negócios.

Por exemplo, no Brasil, o Carrefour vem adotando uma estratégia de usar *blockchain* para rastrear carne suína e bovina, laranjas e outros alimentos [CHINAGLIA, 2021]. O rastreamento vai desde a criação do animal em áreas rurais, ou da lavoura, passando pelas etapas de processamento e indo até a venda nas prateleiras dos supermercados. Os produtos ganham uma identificação dentro do *blockchain*, que garante a confidencialidade dos dados. O consumidor tem acesso a essas informações ao escanear um *QR code* presente nas embalagens. Caso alguns dos fornecedores ao longo da cadeia produtiva não cumpram com os contratos que garantem os princípios ESG (*environmental*, *social* and *governance*) na cadeia produtiva, o Carrefour cancela o contrato com os fornecedores de alimentos. Isso é uma forma

de obrigar esses fornecedores a investirem cada vez mais em uma cadeia socioambiental sustentável.

Assim como Carrefour, outras empresas, como Lojas Renner [BRAGADO, 2022], também vem adotando o uso de *Blockchain* em sistemas de rastreabilidade de produtos. Essa vem sendo uma tendência que grandes marcas ao redor do mundo vem adotando como uma forma de gerar transparência nos agentes envolvidos na fabricação e transporte de seus produtos. A partir dessa transparência, a empresa se adequa melhor aos princípios ESG.

2.1.2 Contratos Inteligentes

Um contrato inteligente é uma regra de negócio contendo a lógica a ser executada em uma *blockchain*, podendo ser uma simples atualização de dados ou uma complexa execução de um contrato com condições anexas [MIERS *et al*, 2020]. Essa regra de Negócio é expressa em código executável.

Presente em diversas plataformas de *blockchain*, os contratos inteligentes expandem as possibilidades dessa tecnologia, que além de armazenar estado, passa a armazenar comportamentos. Qualquer regra pode ser implementada, permitindo que a tecnologia blockchain seja utilizada para aplicações muito mais amplas [AVRAMENKO, 2017].

2.1.3 Tipos de Blockchains

As *blockchains* podem receber diversos tipos de classificação de acordo com os critérios a serem analisados. Um desses critérios, e que será mais explorado aqui, é o critério de permissionamento.

Conforme abordado por [MIERS et al, 2020], uma *blockchain* pode ser classificada como permissionada ou não permissionada. No modelo permissionado uma organização ou um consórcio de organizações são responsáveis pela validação e armazenamento das transações realizadas. Neste modelo, os participantes são previamente conhecidos e dependem de autorização para integrar a rede, gerar ou validar transações. No modelo não permissionado, a validação e armazenamento das transações dependem do trabalho de um grupo de agentes anônimos, conhecidos como mineradores. A atuação dos mineradores é incentivada através da distribuição de *tokens*, como recompensa pelo trabalho realizado. Outra

característica deste modelo é a transparência; os participantes devem ter acesso às informações referentes às transações processadas pela rede, incluindo os endereços e a composição dos blocos.

Já segundo [MIERS et al., 2019], ainda com base no modelo de permissionamento adotado, as *blockchains* são categorizadas como públicas, consorciadas ou privadas, como melhor mostrado abaixo:

- Em uma *blockchain* pública, todos podem ler, enviar ou validar transações, além de participar do processo de consenso distribuído, sendo considerado um modelo totalmente descentralizado.
- Uma *blockchain* consorciada é composta por duas ou mais instituições parceiras, que podem alterar as regras conforme suas necessidades. O consenso é obtido através de um processo realizado por um grupo específico de participantes sendo, desta forma, parcialmente descentralizado.
- Já uma *blockchain* privada é utilizada em modelo de organização única, que pode alterar o funcionamento conforme seus interesses e necessidades. É um modelo centralizado com processo de consenso simples de ser obtido.

O processo de consenso define se qualquer entidade pode participar do processo ou apenas entidades pré-selecionadas [Miers et al. 2019].

2.1.3.1 Ethereum

Apesar de *blockchain* ter inicialmente surgido no contexto financeiro, com a criptomoeda Bitcoin, foi a partir da criação da Ethereum que o uso de *Blockchain* foi expandido para outras fronteiras.

Ethereum é uma plataforma descentralizada capaz de executar contratos inteligentes e aplicações descentralizadas usando a tecnologia *blockchain*. Fundada em 2014 por Vitalik Buterin, foi a primeira a introduzir o conceito de Contratos Inteligentes. Hoje em dia, a *Ethereum* é um ecossistema bastante genérico e serve a uma ampla gama de propósitos, permitindo a construção de aplicativos descentralizados (denominados *DApps*) que fazem uso de contratos inteligentes (desenvolvidos em uma linguagem própria chamada *Solidity*), sendo baseado em algoritmos de consenso para validar transações.

A versão pública da *Ethereum* se caracteriza como uma rede do tipo não permissionada, ou seja, qualquer um pode ter acesso às informações transacionadas nessa rede.

Além disso, existe a criptomoeda *Ether*, associada a essa rede. Sempre que algo for feito na rede, deve ser usado Ether.

2.1.3.2 Hyperledger

Entretanto, por ser uma tecnologia incipiente, as implementações de *blockchain* existentes não conseguiram atender à multiplicidade de requisitos inerentes ao complexo mundo das transações comerciais. Os desafios de escalabilidade e a falta de suporte para transações confidenciais e privadas, entre outras limitações, tornam seu uso impraticável para muitos aplicativos críticos para os negócios [HYPERLEDGER WHITEPAPER].

Nesse contexto, surgiu o Hyperledger. O projeto Hyperledger é uma iniciativa da *Linux Foundation*, e é focado no desenvolvimento de um conjunto de estruturas, ferramentas e bibliotecas para implantação de *blockchain* em nível corporativo. O modelo *Hyperledger* é um projeto multiplataforma, modular e de código aberto. Esta abordagem apresenta, como diferenciais, características como extensibilidade, flexibilidade e a capacidade de modificar qualquer componente sem afetar o sistema [HYPERLEDGER ARCHITECTURE, 2017]. Além disso, os contratos inteligentes do Hyperledger recebem o nome de *chaincode.*

Com esse foco em transações de indústrias e grande corporações, busca-se resolver muitos dos problemas presentes atualmente em *blockchains* aplicadas à ambientes corporativos e à indústria.

Também possui como característica ser do tipo permissionada, ou seja, os participantes são previamente conhecidos e dependem de autorização para integrar a rede, gerar ou validar transações [MIERS *et al*, 2020].

A partir do *Hyperledger*, frameworks são desenvolvidos. Um desses frameworks é o Fabric, que tem como objetivo permitir o desenvolvimento de aplicações empresariais com arquitetura modular, possibilitando a adoção de diferentes mecanismos de consenso e serviços únicos de filiação [MIERS *et al*, 2020].

2.1.3.2.1 Hyperledger Fabric

O *Hyperledger Fabric* é uma arquitetura modular de *blockchain* que permite a conexão de componentes como mecanismo de consenso, serviços de filiação e funções de transação [MIERS et al, 2020].

Em [MIERS et al, 2020], é falado que o modelo proposto pelo Hyperledger Fabric permite que as organizações participem de múltiplas redes *blockchain* independentes, através dos canais. Toda transação deve ser executada em um canal, no qual todos os participantes devem ser autenticados e autorizados a realizar transações. O canal oferece o compartilhamento de uma infraestrutura, mantendo a privacidade dos dados e da comunicação, e é composto pelos nós membros (*peer*) pertencentes às organizações participantes, nós âncora (*anchor peer*), nós de ordenamento das transações (*order peer*), ledger e chaincode. Todas as organizações que compõem o canal devem possuir ao menos um *anchor peer*. Os anchor peer são responsáveis por comunicar-se com os *orderer peer*, obter os blocos contendo as transações, e disseminá-los para os demais nós de suas organizações. Apesar de poderem pertencer a múltiplos canais, e possuir diversos ledger, os dados dos canais são privados, e não podem trafegar entre eles. Já os *orderer peer* são responsáveis por ordenar as transações através de um mecanismo de consenso, montar os blocos e entregá-los aos anchor peer.

Além disso, o *chaincode* pode ser desenvolvido em Go, Javascript e, eventualmente, outra linguagem como Java.

2.1.3.3 Comparação entre Ethereum e Hyperledger

Tendo em vista as duas *blockchains*, foi feita uma comparação para avaliar qual das duas seria melhor de ser aplicada na construção do projeto deste trabalho. A Tabela 1 a seguir, busca mostrar uma comparação entre ambas as redes *blockchain*:

Tabela 1 – comparação *Ethereum x Hyperledger*

CRITÉRIO	ETHEREUM	HYPERLEDGER FABRIC
Usabilidade	Ideal para uso B2C (mais para o lado do cliente)	Ideal para uso B2B (mais para o lado da empresa)
Consenso	Usa o Consenso de **Prova de Trabalho** (***Proof of Work - PoW***)	Não tem mecanismo de consenso. Usuários criam seus próprios algoritmos de consenso
Permissão	não permissionada	permissionada
Tokens	possui o Ether (ETH)	não possui criptomoeda/token embutida
Confidencialidade	blockchain pública (todos os participantes podem acessar as transações)	blockchain privada (precisa de permissão para ter acesso)
Visibilidade e Segurança	todas as transações são inteiramente transparentes e qualquer um com acesso à Internet pode visualizar	Isso é altamente seguro e confidencial. As organizações ou indivíduos que possuem o Certificado de Autorização só podem visualizar todas as transações na rede.
Contratos Inteligentes	chamados de *Smart Contracts*	chamados de *chaincode*
Linguagem dos Contratos Inteligentes	escritos em *Solidity*	escritos em Go, Java, Javascript
Velocidade das transações	+/- 20 transações por segundo	+/- 2000 transações por segundo
Governança	desenvolvedores da Ethereum	Linux Foundation

Fonte: o autor

Com base nessa comparação, pode-se observar que para um contexto de rastreabilidade de uma cadeia logística a *blockchain* ideal é a *Hyperledger*. Essa rede (mais precisamente o framework Hyperledger Fabric) foi escolhida para a construção de nossa arquitetura.

2.2 Virtualização

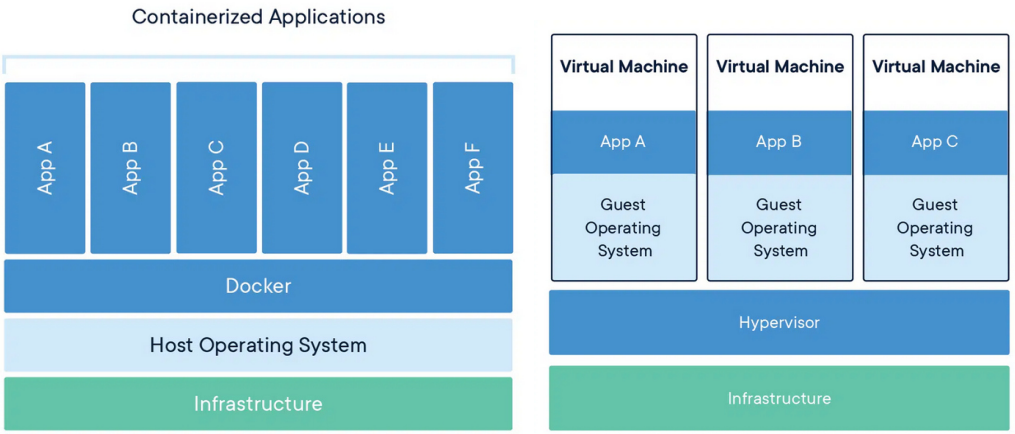
A virtualização é uma tecnologia que cria ambientes virtuais em um único sistema físico, permitindo a execução de sistemas operacionais ou aplicativos independentes, isolados uns dos outros. Isso é possível graças ao uso de um software chamado *hypervisor*, que permite a criação de máquinas virtuais que compartilham os recursos do hardware. As Máquinas Virtuais oferecem isolamento e segurança, funcionando como uma máquina completa com seu próprio sistema operacional e recursos alocados, e, assim, possibilitando o melhor aproveitamento dos recursos do servidor, facilitando o gerenciamento e reduzindo custos. Também é importante destacar que o uso dessas máquinas virtuais facilita a criação de ambientes de teste, desenvolvimento e homologação, oferecendo um ambiente controlado para experimentação.

2.2.1 Containers

Ainda no contexto deste trabalho, também é interessante abordar os conceitos de *containers*. C*ontainer* consiste em uma forma de virtualização que cria um ambiente isolado e utiliza menos recursos de máquina, quando comparado a outros processos de virtualização, como as máquinas virtuais. Um *container* contém um conjunto de processos que são executados a partir de uma imagem, que serve como um modelo para a criação daquela aplicação. É ideal para ser usado para testar um caso específico, dispensando o uso de máquinas virtuais.

Os contêineres são uma abstração na camada do aplicativo que agrupa código e dependências. Vários contêineres podem ser executados na mesma máquina e compartilhar o kernel do sistema operacional com outros contêineres, cada um executando como processos isolados no espaço do usuário. Os contêineres ocupam menos espaço do que as máquinas virtuais (as imagens de *container* geralmente têm dezenas de *megabytes*), podem lidar com mais aplicativos e exigem menos máquinas virtuais e sistemas operacionais.

Figura 1 – Diferença entre *Container* (à esquerda) e Máquina Virtual (à direita)



Fonte: https://www.docker.com/resources/what-container/

2.2.2 Docker

Atrelado ao conceito de *container*, está também o conceito de Docker, uma plataforma open-source que auxilia na construção e implantação de *containers* de forma mais eficiente. O Docker utiliza recursos do kernel do Linux para isolar processos e criar ambientes de execução independentes. Essa abordagem traz inúmeros benefícios, como a capacidade de executar aplicações em diferentes ambientes de forma consistente, aprimorar a escalabilidade e a eficiência dos recursos de hardware, além de facilitar a distribuição e o compartilhamento de software.

A plataforma Docker leva o mesmo nome da empresa que a desenvolveu, Docker Inc®. Ela se tornou amplamente popular na comunidade de desenvolvimento de software devido à sua abordagem inovadora para a virtualização de aplicativos.

Uma das principais vantagens do Docker é sua portabilidade. Os *containers* Docker são criados a partir de *imagens*, que são pacotes autossuficientes contendo todos os componentes necessários para executar um aplicativo, incluindo o código, bibliotecas, dependências e configurações. Essas imagens podem ser facilmente compartilhadas e distribuídas entre diferentes ambientes de desenvolvimento, teste e produção, garantindo que o aplicativo funcione consistentemente em qualquer infraestrutura que possua o Docker instalado.

Para auxiliar no uso e administração dos *containers*, o Docker oferece o Docker Desktop, um software para desktop que fornece uma interface gráfica que facilita o uso e a administração dos *containers* criados em Docker, dispensando em muitos casos o uso de linhas de comando. O uso dessa plataforma representou uma vantagem na hora de colher os resultados do teste e simplificar a análise.

No contexto deste trabalho, a plataforma Docker foi escolhida como recurso de virtualização para realizar os testes do modelo proposto. Ao adotar o Docker, tornou-se possível criar e implantar facilmente um ambiente consistente para executar os experimentos, garantindo a reprodução dos resultados. Além disso, o uso do Docker Desktop permitiu analisar métricas da rede criada e dos *containers* presentes nela de maneira simplificada, oferecendo uma interface gráfica para visualizar e interpretar os dados coletados.

3 TRABALHOS RELACIONADOS

Com o objetivo de desenvolver uma arquitetura modelo, faz-se muito importante ter conhecimento de outras obras que estejam relacionadas com o assunto e que contribuíram para o desenvolvimento do projeto deste artigo. Algumas dessas obras abordam soluções semelhantes para casos específicos de rastreabilidade.

Em [SCHRÖDER, 2019], a autora propôs uma solução para Rastreamento de Medicamentos utilizando etiquetas de identificação por radiofrequência (RFID), cujo número de identificação (ID) pode ser lido por leitores presentes em determinados pontos de verificação ao longo da cadeia de movimentação do medicamento. Ao passar por determinados pontos da cadeia, as informações eram lidas e enviadas para serem armazenadas em uma *blockchain* privada (no trabalho foi usada a rede *Ethereum*). A aplicação desenvolvida para esse trabalho buscou demonstrar a viabilidade dessa solução.

Em [THAKKER *et al*, 2021], os autores abordam a adoção da tecnologia do *blockchain* na indústria diamantífera, com o rastreamento dos lotes de diamantes, e também apresentam prós e contras dessa integração. Além disso, buscam apresentar estudos de caso de empresas que aplicam *blockchain* na rastreabilidade de diamantes, usando-se da rede *Hyperledger*.

Tanto [ARANDA, 2022], quanto [SILVA, 2020] são trabalhos que apresentaram um rico estudo acerca de cadeias de suprimentos, rastreabilidade e seus desafios no contexto

atual. Além disso, discorrem acerca das entidades componentes de uma cadeia desse tipo. Muitos dos conhecimentos apresentados nessas duas obras serviram de inspiração para a construção do modelo de cadeia a ser apresentado neste trabalho. Em [ARANDA, 2022] existe uma apresentação de uma proposta de arquitetura para a rastreabilidade de medicamentos utilizando a tecnologia *Blockchain* Hyperledger, além de apresentar um comparativo das arquiteturas encontradas na revisão da literatura com a arquitetura apresentada pelo autor. Com isso, traz discussões muito interessantes sobre os conceitos de Arquitetura de Solução. Entretanto, o trabalho não chegou a desenvolver a solução proposta. Já em [SILVA et al, 2020] existe o desenvolvimento da solução de arquitetura *blockchain* proposta, inclusive utilizando-se de ambiente *NodeJS* para o desenvolvimento do backend, correspondendo à *blockchain*. Os métodos criados para essa solução foram ricamente documentados neste trabalho.

Em [COSTA, 2021], o autor constrói um mecanismo para gerenciamento de recursos dos nós de computação em névoa e balanceamento de carga de forma descentralizada e autônoma, baseando-se no uso associado de *blockchain Ethereum* para lidar com os desafios do gerenciamento na névoa. A principal contribuição deste trabalho está no uso do recurso da virtualização por meio de *containers* em *Docker* para emular os diferentes nós envolvidos, servindo de inspiração para o desenvolvimento dos testes com a rede criada para este trabalho.

Em [MIERS et al, 2020] e [MIERS et al, 2019], os autores discorrem sobre a arquitetura do Hyperledger, criando um rico material para fonte de consultas acerca dessa *Blockchain*, base para muitos dos conhecimentos aqui mostrados. Também nesse material contém um importante passo a passo que auxiliou na implantação da rede para testes deste artigo, bem como o *deploy* do contrato inteligente nesta mesma rede.

Comparado com os trabalhos anteriormente mencionados, que, no geral, abordam casos específicos de rastreabilidade, o presente trabalho apresenta um modelo geral de aplicação em *blockchain* para o uso em Sistemas de Rastreabilidade. Ao construir uma abordagem mais generalista, esta obra pretende fornecer um modelo que facilite a construção de outras abordagens de rastreabilidade para produtos e situações mais específicas, bem como fornecer uma importante documentação de consulta para os temas relativos a esse assunto. Estes trabalhos relacionados anteriormente também contribuíram para o fornecimento de uma documentação interessante que ajudou no desenvolvimento de testes em pequena escala da solução proposta, e que demonstraram a viabilidade dos princípios básicos do projeto, bem como ajudaram no desenvolvimento da lógica dos códigos e métodos usados no contrato inteligente.

Tabela 2 – Tabela Comparativa dos Trabalhos Relacionados

Referência	Tipo de Blockchain	Linguagem Contrato Inteligente	Ambientes de Execução	Documentação dos Testes e Experimentos	Produto Analisado
[SCHRÖDER, 2019]	Ethereum	Solidity	Ethereum Virtual Machine (EVM)	É documentado	medicamentos
[THAKKER et al, 2021]	Hyperledger	–	–	*Não tem*	diamantes
[ARANDA, 2022]	Hyperledger	–	–	NÃO É documentado	medicamentos
[SILVA, 2020]	blockchain própria	Javascript	NodeJS	É documentado	erva-mate
[COSTA, 2021]	Ethereum	Solidity	Ethereum Virtual Machine (EVM)	É documentado	–
Este Trabalho	Hyperledger	Javascript	Docker	É documentado	Caso geral

Fonte: o autor

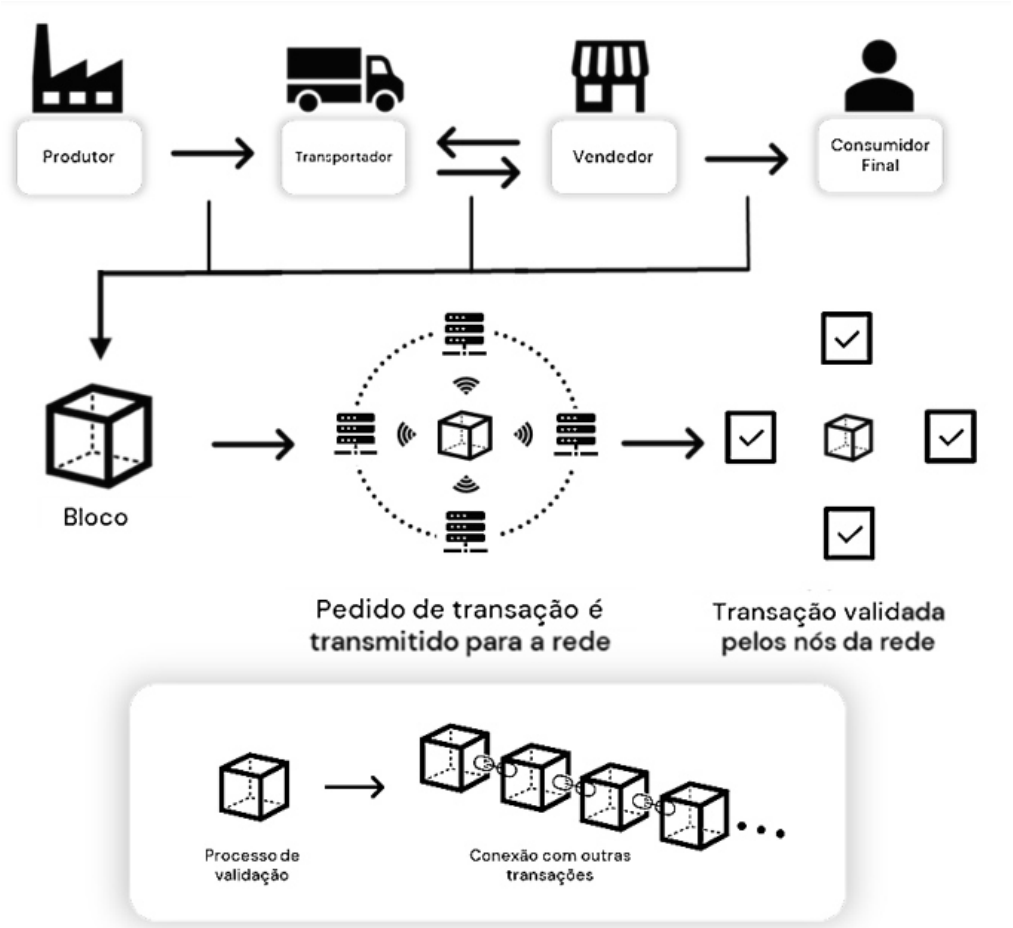
4 METODOLOGIA

4.1 Arquitetura Proposta

O termo *arquitetura* representa a melhor abordagem para visualização de uma ideia, garantindo certo nível de abstração do entendimento de negócios (segmentação de mercado, cadeia de valor, direcionamento chave do cliente, entre outros) e tecnológico (sistemas, interfaces, funcionalidades etc.) [CLOUTIER et al, 2010].

A Figura 2 apresenta a arquitetura de rastreabilidade de medicamentos utilizando a tecnologia *Blockchain*, incluindo o fluxo que o produto deverá seguir:

Figura 2 – Proposta de arquitetura para a rastreabilidade de produtos utilizando Blockchain



Fonte: o autor

A proposta de arquitetura, conforme mostrada na Figura 2, foi considerada possuindo três entidades (no caso, empresas) para compor essa cadeia: *Produtor*, *Transportador* e *Vendedor*. Cada entidade será responsável por, respectivamente, criar o produto, transportar o produto e vender o produto. Essa venda do produto pode ser para um *Consumidor Final* (caso o *Vendedor* tenha um modelo de negócio do tipo *Business-to-Consumer – B2C),* ou uma outra *Empresa* (caso o *Vendedor* tenha um modelo de negócio do tipo *Business-to-Business – B2B*).

Em uma cadeia logística na vida real, poderão existir vários produtores (no caso de um produto ser feito com peças de diferentes fornecedores); poderão existir vários transportadores (no caso de o produto ser transportado por diversas empresas ao longo do caminho da produção à venda); e, por fim, poderão existir vários vendedores (no caso de ser um modelo de negócio B2B, onde um *vendedor* venderia para outro *vendedor*, antes de chegar no consumidor final; no caso de ocorrer a venda para outra empresa, o produto voltaria

novamente para *Transportador*). Também em uma cadeia logística poderão existir outras etapas além das acima mostradas. Mas, a fim de demonstrar essa simulação, foi considerada que para uma cadeia logística funcionar, as etapas *mínimas* que viabilizam uma análise são as que referenciam a fabricação do produto (com a organização *Produtor*), transporte do produto do local de fabricação até o local de venda (com a organização *Transportador*), e, por fim, a venda do produto propriamente dito (com a organização *Vendedor*). Considera-se que a validação do funcionamento da solução para essa cadeia simplificada valida também o funcionamento do sistema para cadeias mais complexas, conforme abordado anteriormente.

Com base no modelo de permissionamento, a *blockchain* dessa solução deste artigo constitui-se como uma *Blockchain* Consorciada, tendo em vista o consenso ser obtido através de um processo realizado por um grupo específico de participantes sendo, desta forma, parcialmente descentralizado [MIERS et al, 2019].

Para essa simulação, será feita uma abstração do produto. Cada produto conterá informações que serão armazenadas em um "*bloco de informação*". Cada "bloco" será um objeto cujos dados serão armazenados no livro-razão (*ledger*) da Rede criada.

Esses dados conterão as seguintes informações:

- `product_ID`: número de identificação de determinado produto (valor único – *chave primária*)
- `productName`: nome do Produto
- `CNPJ_Produtor`: número único que identifica o agente responsável por *criar o produto*
- `CNPJ_Transportador`: número único que identifica o agente responsável por *transportar o produto*
- `CNPJ_Vendedor`: número único que identifica o agente responsável por *vender o produto*
- `Data_Produtor`: data em que o produto foi construído
- `Data_Transportador`: data em que o Transportador deu início ao transporte do produto, de onde foi produzido (*Produtor*) até onde será vendido (*Vendedor*)
- `Data_Vendedor`: data referente ao momento em que o produto chegou ao *Vendedor*, para posteriormente ser vendido
- `Informacoes_adicionais`: dados que o criador pode estar colocando para agregar mais informações e conhecimentos acerca do Produto.

Para esse trabalho, é necessário utilizarmos uma forma de identificar o agente responsável por cada etapa, e essa identificação deve ser única em toda a rede. Como se tratam de organizações empresariais, a forma de identificação única escolhida foi usar o próprio *CNPJ* (Comprovante Nacional de Pessoa Jurídica) da Empresa responsável por aquela etapa.

A alteração dos dados do tipo *Data* e do tipo *CNPJ* (obedecendo a regras previamente estabelecidas) é que dará a noção de movimentação do produto ao longo da Cadeia. Essa abstração consegue simular de uma forma simples a movimentação de um produto em uma cadeia de distribuição logística, desde a produção até a sua posterior venda. Para que esse fluxo acima seja obedecido, as regras de registro do bloco de informação deverão ser essas:

- "Data_Produtor" deve SEMPRE ser anterior à "Data_Transportador" (se ela existir);
- Quando "Data_Vendedor" for ser inserida pela primeira vez, "Data_Transportador" e "Data_Produtor" não podem ser nulo;
- Da mesma forma, quando "Data_Transportador" for ser inserida pela primeira vez, "Data_Produtor" não pode ser nulo;
- O mesmo raciocínio vale para os CNPJs: quando *CNPJ_Vendedor* for ser inserida pela primeira vez, *CNPJ_Transportador* e *CNPJ_Produtor* não podem ser nulos; da mesma forma, quando *CNPJ_Transportador* for ser inserida pela primeira vez, *CNPJ_Produtor* não pode ser *nulo*

Ao se estabelecer essas regras, o objetivo principal é garantir que o fluxo inicialmente estabelecido (Produtor → Transportador → Vendedor) seja obedecido, bem como ser possível acompanhar o histórico dessa movimentação. Para que uma determinada transação ocorra, deve ser validada pelos agentes através de Mecanismo de Consenso. Cada transação, entre cada um desses agentes, é armazenada num Livro-Razão (é aqui que temos um histórico de transações dessa cadeia logística, que poderá ser acessado posteriormente para consulta e auditoria – o que garantirá um alto grau de segurança e confiabilidade da cadeia).

4.2 Contrato Inteligente

O contrato inteligente vai possuir métodos que permitam essas transações. Cada uma das organizações dessa cadeia irá receber uma cópia desse contrato inteligente. O contrato inteligente, desenvolvido na linguagem *Javascript*, irá conter os seguintes métodos:

- `createProduct`: método para a criação do conjunto de informações que irá representar o produto (só pode ser usado pelo "*Produtor");*
- `readProduct`: método para consultar informações do produto, passando como parâmetro o número de identificação dele (pode ser acessado por qualquer organização);
- `transferProduct`: método para registrar as transferências de produto entre organizações, obedecendo a sequência mostrada na Figura 2. *Transportador* só poderá realizar o registro se produto já tiver sido criado pelo *Produtor*. Por sua vez, o *Vendedor* só poderá realizar o registro se o produto já tiver passado pelo *Transportador*. Também existe uma parte específica desse método para caso de Vendedor que vende o produto para outro *Vendedor*, passando por um *Transportador.* Para esse caso, deverão ser criados novos campos de Data_Transportador e Data_Vendedor, bem como CNPJ_Transportador e CNPJ_Vendedor (sem afetar os que já foram criados anteriormente);
- `queryAll`: método para consultar informações de todos os produtos registrados no Ledger (pode ser acessado por qualquer organização);
- `retrieveHistory`: método para acessar o histórico de Transações de um produto ao longo da cadeia logística, passando como parâmetro o número de identificação dele (pode ser acessado por qualquer organização);
- `deleteProduct`: método para excluir um produto; pode ser executado pelo *Produtor* (desde que produto não tenha chegado nas etapas de *Transportador* e *Vendedor,* demonstrando que o produto foi descartado) e pelo *Vendedor* (na situação que o produto foi vendido – seja consumidor final, ou seja empresa);

4.3 Rede

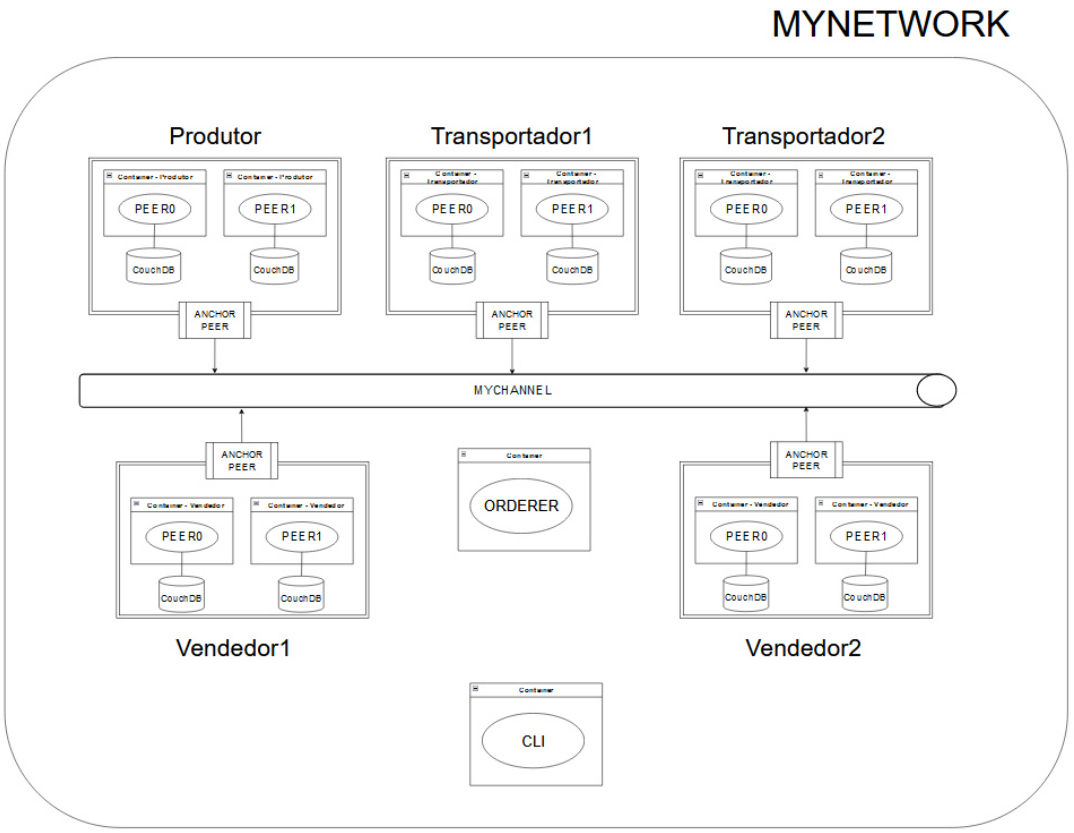
Para este trabalho, o cenário de exemplo é composto por:

- 5 organizações (*Produtor, Transportador1, Transportador2*, *Vendedor1 e Vendedor2*) compostas por 2 peers cada;
- 1 nó ordenador, responsável por executar o "consenso" através do mecanismo SOLO (somente para desenvolvimento);
- 1 canal para comunicação entre todos os participantes da rede;
- 5 certificados de autoridade (um para cada organização).

Ao escolhermos criar dois *peers* por organização buscamos ilustrar um cenário de rede mais realista e distribuído. Ter mais de um *peer* por organização oferece uma maior resiliência à rede, melhorando a tolerância à falhas no caso de um dos *peers* enfrentar algum problema. Também cria uma redundância permitindo que os dados sejam replicados e sincronizados entre os peers, garantindo maior disponibilidade dos dados. Por fim, a rede pode acomodar um maior número de transações e participantes, melhorando a escalabilidade da rede, além de permitir a criação de canais de comunicação privados entre os peers de diferentes organizações.

Vamos denominar as organizações de *Produtor, Transportador1, Transportador2, Vendedor1 e Vendedor2*. *Vendedor1* irá receber o produto de *Transportador1*, e posteriormente vendê-lo para *Vendedor2*, fazendo o produto ser transportado pelo *Transportador2*. No final, Vendedor2 irá vender o produto ao consumidor final, encerrando a cadeia. Essa é a forma mais simples que valida o modelo de arquitetura proposta (embora não seja a cadeia mais simples possível, tendo em vista que poderia existir apenas um *Vendedor* que já vendesse direto para o consumidor final). Essas organizações farão parte da rede, denominado *mynetwork*, e se comunicarão através do mesmo canal, denominado *mychannel*. Os peers das organizações serão denominados peer0 e peer1, por simplicidade. Na Figura 3, vemos uma representação dessa rede:

Figura 3 – Representação da Rede



Fonte: o autor

Por se tratar de uma solução de contexto empresarial e que poderá conter informações que não deverão necessariamente ser acessadas por qualquer um, torna-se mais adequada a escolha por usar uma rede do tipo permissionada (como é o caso da *Hyperledger)*. Com a utilização desse tipo de rede, somente os agentes participantes dela, pertencentes a cada uma das organizações previamente estabelecidas e que receberam a autorização de acesso, é que poderão acessar os dados e fazer alterações dos mesmos, restringindo, assim, o acesso dessas informações por parte de organizações que não tem essa permissão e protegendo a confidencialidade desses dados.

Para a implementação da rede, foi utilizado o recurso de virtualização por meio de *containers*, a partir do *Docker*. Cada *peer* representado na Figura 3 irá ser criado dentro de um *container* específico. O uso de *containers* é indicado para esse tipo de caso, pois consegue criar um ambiente isolado para cada *peer*, economizando mais recursos do que se fossem usadas máquinas virtuais, o que favorece o cenário de simulação.

O *Chaincode* desenvolvido foi instalado em cada *peer*. Um dos peers foi usado para instanciar o *chaincode* no canal (*channel*) – no caso, foi o *peer0* de Produtor. Ao realizar a criação de um produto, automaticamente a rede criou *containers* específicos para cada *chaincode* instalado em cada Organização. A partir desses *containers* novos, é possível efetuar as transações dessa rede.

5 IMPLEMENTAÇÃO E RESULTADOS

Este capítulo visa mostrar o funcionamento da arquitetura desenvolvida a partir da criação de um modelo em pequena escala. A ideia principal por trás disso é validar o funcionamento dos princípios básicos da solução proposta em um modelo de pequena escala e abrir discussões acerca da viabilidade da solução para modelos em grande escala.

A solução para esse projeto consiste em duas partes: o contrato inteligente e a rede implementada. Os testes conduzidos foram divididos em dois tipos: o primeiro tipo de teste buscou validar a correta execução do contrato inteligente; já o segundo tipo de teste buscou verificar a implementação da rede. A validação dos dois tipos de testes nos permite concluir que a solução proposta foi validada como uma solução viável.

Todos os testes foram executados com sucesso em um computador com processador Intel(R) Core(TM) i5-1135G7 @ 2.40GHz 2.42 GHz, Memória RAM de 16GB e sistema operacional Ubuntu 20.04, acessado via WSL instalado em Windows 11. A realização desses testes foi feita a partir da interação via linha de comando pelo terminal.

Os testes para o contrato inteligente foram escritos em *Javascript* utilizando-se o editor de texto *Visual Studio Code*. Inicialmente, para verificar a execução dos métodos antes do *deploy*, foi usado o *IBM Blockchain Platform*, ferramenta mantida pela Linux Foundation, que auxilia no desenvolvimento de contratos inteligentes, ajudando o desenvolvedor a testar e refatorar o código quantas vezes for necessário, antes de realizar o *deploy* do mesmo.

Os testes para a rede foram feitos utilizando a criação de *containers*, por meio da utilização do Docker. Com a criação destes *containers*, buscou-se simular a criação das entidades envolvidas na rede, bem como a interação entre elas. Para verificar melhor os *containers* criados e, assim, poder verificar melhor a rede criada, foi utilizado o software *Docker Desktop,* aplicação que fornece uma interface gráfica que facilita a visualização e gerenciamento dos *containers* criados.

Para a verificação da execução dos métodos do Contrato Inteligente, foram criados cinco lotes de informação que simulavam o funcionamento dessa cadeia logística e o seu rastreamento. Os valores eram fictícios, a fim de demonstração. Eles foram documentados na tabela 3:

Tabela 3 – Lotes de Produtos criados para Simulação

Lote do Produto	Nome do Produto	ID do Produto	CNPJ do Produtor	Informações Adicionais
Lote 1	celular	001	000.001	modelo da cor preta
Lote 2	computador	A100	0/001	processador i5
Lote 3	caneta	1001	111/001	tinta cor azul
Lote 4	soja	ABC-01	100.001	produção colhida no Paraná
Lote 5	vacina COVID	2.001	222/0001	produzida pela Pfizer

Fonte: o autor

5.1 Testes da Rede implementada

5.1.1 Analisar o deploy do Contrato Inteligente na blockchain

O *deploy* do *chaincode* no Hyperledger Fabric abrange tanto a sua instalação quanto a sua instanciação. A instalação do *chaincode* é o processo de disponibilizá-lo em cada peer da rede, ou seja, enviar o código do *chaincode* para cada *peer* e instalá-lo localmente em sua respectiva organização. Após a instalação, a instanciação ocorre, na qual uma instância específica do chaincode é criada em um canal específico da rede. Para que ocorra a instanciação, é necessário informar o nó ordenador, o canal onde está sendo feita a instanciação, o nome do *chaincode* e a linguagem usada, junto de sua versão. Além disso, é necessário informar o comando de início da instanciação do chaincode, acompanhado de uma

payload. Por fim, é importante passar as políticas de consenso adotadas, o que irá validar as transações. No caso desta simulação, foi determinado que as transações só irão ocorrer se autorizadas em conjunto pelas cinco organizações envolvidas (*Produtor, Transportador1, Transportador2, Vendedor1 e Vendedor2*). Combinando essas duas etapas, o *chaincode* é implantado e está pronto para ser invocado e executado pelos participantes da rede no canal especificado.

Para realizar isso, existem duas opções: a primeira forma é executar esses comandos em cada *container*; já a segunda forma é criar um *container* chamado CLI, para realizarmos essas atividades. O *container* CLI (*Command Line Interface* – Interface de Linha de Comando) é uma instância em execução do Hyperledger Fabric que fornece uma interface de linha de comando para interagir com a rede *blockchain*, permitindo que os usuários emitam comandos e interajam com os componentes da rede, como criar e instanciar contratos inteligentes, gerenciar identidades, realizar consultas de transações, entre outras tarefas. Com isso, ele se torna uma interface de fácil utilização para gerenciar e interagir com a rede, permitindo a execução de comandos para realizar diversas operações. Para mudar o *container* onde é executado cada comando, basta mudar as variáveis de ambiente dentro do CLI. Por essa facilidades, a segunda forma foi adotada.

Como resultado, foram criados novos *containers* do tipo “dev-peer”, que são *containers* geralmente usados para desenvolvimento e teste de *chaincodes*. A partir deles, podemos observar a execução dos métodos do contrato inteligente, a partir dos *Logs*, registros que contém informações de atividades e eventos que ocorrem dentro dos *containers*.

5.1.2 Teste de segurança

Foi conduzido o seguinte teste para validar a segurança da rede, descritos na tabela a seguir:

Tabela 4 – Teste de Segurança

Resultado Esperado	Descrição do Teste	Resultado Alcançado
Não deve ser possível organizações que NÃO POSSUEM permissão de acesso à rede pesquisarem informações do produto	Invoca-se o método "readProduct" do *chaincode* através de uma organização com endereço e ID não configurado na rede.	Ocorreu um erro e informa que a conexão não foi possível.
Não deve ser possível organizações que NÃO POSSUEM permissão de acesso à rede pesquisarem informações dos produtos registrados no ledger	Invoca-se o método "queryAll" do *chaincode* através de uma organização com endereço e ID não configurado na rede.	Ocorreu um erro e informa que a conexão não foi possível.
Não deve ser possível organizações que NÃO POSSUEM permissão de acesso à rede pesquisarem histórico do produto	Invoca-se o método "retrieveHistory" do *chaincode* através de uma organização com endereço e ID não configurado na rede.	Ocorreu um erro e informa que a conexão não foi possível.

Fonte: o autor

5.2 Testes do Contrato Inteligente

Foram desenvolvidos testes para testar individualmente os métodos do contrato inteligente.

5.2.1 Testes de Criação de Produto

Foram conduzidos testes para avaliar o processo de criação de um produto. Cada uma das organizações tentou criar um produto. Porém, apenas o Produtor é quem obteve sucesso com essa operação.

Tabela 5 – Teste de Criação de Produto

Resultado Esperado	Descrição do Teste	Resultado Alcançado
"*Produtor*" deve conseguir criar Produto	"*Produtor*" invoca o método "createProduct" e passa as informações necessárias para a criação de um lote de informação (que é a representação do Produto na nossa blockchain)	Produto é registrado com sucesso
"*Transportador1*" e "*Transportador2*" **NÃO** devem conseguir criar Produto	"*Transportador1*" e "*Transportador2*" invoca o método "createProduct" e passa as informações necessárias para a criação de um lote de informação (que é a representação do Produto na nossa *blockchain*)	Ocorreu um erro e Produto NÃO é registrado com sucesso
"*Vendedor1*" e "*Vendedor2*" **NÃO** devem conseguir criar Produto	"*Vendedor1*" e "*Vendedor2*" invocam o método "createProduct" e passa as informações necessárias para a criação de um lote de informação (que é a representação do Produto na nossa *blockchain*)	Ocorreu um erro e Produto NÃO é registrado com sucesso
Produto já criado **NÃO** pode ser criado novamente	Invoca-se o método "createProduct" e passa as informações do ID de um lote já anteriormente criado (qualquer organização pode fazer isso)	Ocorreu um erro e mostra que produto já foi anteriormente criado

Fonte: o autor

5.2.2 Testes de Leitura do Produto

Foram conduzidos testes para avaliar a leitura de informações de um produto, podendo ser feita por qualquer organização.

Tabela 6 – Teste de Leitura do Produto

Resultado Esperado	**Descrição do Teste**	**Resultado Alcançado**
Deve-se consultar informações do produto pelo ID dele (representado por "product_ID")	Invoca-se o método "readProduct" passando como parâmetro ("product_ID"). O método pode ser invocado por qualquer organização (*"Produtor", "Transportador1", "Transportador2", "Vendedor1" e "Vendedor2"*)	Retornou a informação do produto cadastrado com aquele ID, consistente com o que foi feito na criação dele.
É realizada a consulta de um produto que NÃO existe	Invoca-se o método "readProduct" passando como parâmetro um "product_ID" que não foi criado. O método pode ser invocado por qualquer organização (*"Produtor", "Transportador1", "Transportador2", "Vendedor1" e "Vendedor2"*)	Retornou a informação que o produto NÃO EXISTE.

Fonte: o autor

5.2.3 Testes Transferência do Produto

Foram conduzidos testes para avaliar o processo de transferência de um produto. *Transportador1* e *Vendedor1* tentaram fazer essa atualização um produto. *Transportador1* só podia realizar isso se produto tivesse sido criado por Produtor. Já *Vendedor1* só poderia realizar a atualização se *Transportador1* já tivesse atualizado. De forma análoga, *Transportador2* só podia realizar transferência se produtor tivesse em *Vendedor1*. Já *Vendedor1* só poderia realizar a atualização se *Transportador2* já tivesse atualizado. Essas atualizações demonstram em qual etapa da cadeia logística o produto está.

Tabela 7 – Teste de Transferência do Produto

Resultado Esperado	Descrição do Teste	Resultado Alcançado
Vendedor1 e *Vendedor2* **NÃO** devem conseguir efetuar o registro da transferência do produto a partir de *Produtor*	*Vendedor1* e *Vendedor2* invoca o método "transferProduct", passando como parâmetros Id do Produto, nome do Produto, CNPJ do Transportador, além do nome da própria organização.	Ocorreu um erro e retorna como resultado que aquela operação NÃO É POSSÍVEL
Transportador1 deve conseguir efetuar o registro da transferência do produto a partir de *Produtor*	*Transportador1* invoca o método "transferProduct", passando como parâmetros Id do Produto, nome do Produto, CNPJ do Transportador, além do nome da própria organização.	Retornou que a operação foi validada. Além disso, também é registrado a data em que foi efetuada aquela operação.
Vendedor1 deve conseguir efetuar o registro da transferência do produto a partir de *Transportador1*	*Vendedor1* invoca o método "transferProduct", passando como parâmetros Id do Produto, nome do Produto, CNPJ do Transportador, além do nome da própria	Retornou que a operação foi validada. Além disso, também é registrado a data em que foi efetuada aquela operação.

	organização.	
Transportador2 deve conseguir efetuar o registro da transferência do produto a partir de *Vendedor1*	*Transportador1* invoca o método “transferProduct”, passando como parâmetros Id do Produto, nome do Produto, CNPJ do Transportador, além do nome da própria organização.	Retornou que a operação foi validada. Além disso, também é registrado a data em que foi efetuada aquela operação.
Vendedor2 deve conseguir efetuar o registro da transferência do produto a partir de *Transportador2*	*Vendedor2* invoca o método “transferProduct”, passando como parâmetros Id do Produto, nome do Produto, CNPJ do Transportador, além do nome da própria organização.	Retornou que a operação foi validada. Além disso, também é registrado a data em que foi efetuada aquela operação.

Fonte: o autor

5.2.4 Testes Exclusão do Produto

Nestes testes, foram simulados o processo de exclusão do lote de informação do produto da rede. Apenas o *Produtor* e os *Vendedores* poderiam realizar a exclusão. O *Produtor* só poderia realizar a exclusão se o produto ainda não tivesse passado por "*Transportadores*" nem "*Vendedores*". Se o *Produtor* excluísse o produto, significaria que o produto saiu da cadeia de distribuição (por motivos variados), sendo descartado. Por sua vez, os *Vendedores* também conseguiriam excluir produto, desde que já tivesse passado pelas etapas de *Produtor*, *Transportador1* (para *Vendedor1*) e além das etapas de *Vendedor1 e Transportador2* (para *Vendedor2*). Se os *Vendedores* excluíssem o produto significa que o produto foi vendido para o consumidor final ou foi descartado para a venda (motivos variados).

Entretanto, os *Transportadores* não podem excluir o produto. Os *Transportadores* não podem ser capazes de excluir o produto porque essa é uma garantia de segurança para a

cadeia logística, para evitar que o produto tenha seus dados perdidos ao longo da cadeia de distribuição.

Tabela 8 – Teste de Exclusão do Produto

Resultado Esperado	**Descrição do Teste**	**Resultado Alcançado**
Produtor consegue excluir produto	*Produtor* invoca o método "deleteProduct" passando como parâmetro o ID do produto e o nome da própria organização.	Retornou que a operação foi realizada com sucesso. Posterior leitura mostra que ID daquele produto já não existe mais.
Transportador1 e Transportador2 **NÃO** podem excluir produto	*Transportador1 e Transportador2* invocam o método "deleteProduct" passando como parâmetro o ID do produto e o nome da própria organização.	Ocorreu um erro e retorna como resultado que aquela operação não é possível
Vendedor1 consegue excluir produto, desde que já tenha passado pelas etapas de *Produtor* e *Transportador1*	*Vendedor1* invoca o método "deleteProduct" passando como parâmetro o ID do produto e o nome da própria organização.	Retornou que a operação foi realizada com sucesso. Posterior leitura mostra que ID daquele produto já não existe mais.
Vendedor2 consegue excluir produto, desde que já tenha passado pelas etapas de *Produtor* , *Transportador1, Vendodor1 e Transportador2*	*Vendedor2* invoca o método "deleteProduct" passando como parâmetro o ID do produto e o nome da própria organização.	Retornou que a operação foi realizada com sucesso. Posterior leitura mostra que ID daquele produto já não existe mais.

Se produto já tiver sido excluído por *Vendedor1*, *Vendedor2* NÃO DEVE conseguir excluí-lo	*Vendedor2* invoca o método “deleteProduct” passando como parâmetro o ID do produto já excluido e o nome da própria organização.	Ocorreu um erro e retorna como resultado que aquela operação NÃO É POSSÍVEL

Fonte: o autor

5.2.5 Teste de Leitura de todos os produtos registrados

Nesses testes foram simulados a leitura de todos os produtos registrados no Ledger, podendo ser feita por qualquer organização. É uma forma de ter uma leitura mais global dos produtos até então registrados.

Tabela 9 – Teste de Leitura de todos os produtos registrados

Resultado Esperado	**Descrição do Teste**	**Resultado Alcançado**
Deve-se consultar informações de todos os produtos registrados no Ledger	Invoca-se o método "queryAll". Não se necessita passar o nome da organização	Retorna a informação de todos os produtos cadastrados naquele momento no *Ledger*.

Fonte: o autor

5.2.6 Teste Verificar Histórico daquele Produto

Nesses testes foram simulados a leitura de todo o histórico de transações de um produto ao longo da cadeia logística. Com esse histórico, podemos ter ideia do “caminho” percorrido por determinado produto ao longo da cadeia logística, facilitando o rastreamento do mesmo, funcionando até mesmo para produtos já excluídos. Qualquer organização pode utilizar esse método.

Tabela 10 – Teste Verificar Histórico daquele Produto

Resultado Esperado	Descrição do Teste	Resultado Alcançado
Deve-se consultar histórico de movimentação de um determinado produto registrado no Ledger	Invoca-se o método "retrieveHistory", passando como parâmetro o “product_ID”. Pode ser acessado por qualquer organização.	Retorna histórico de atualizações de um determinado produto registrado no Ledger.
Deve-se consultar histórico de movimentação de um determinado produto anteriormente excluído do Ledger pelo método “deleteProduct”	Invoca-se o método "retrieveHistory", passando como parâmetro um “product_ID” anteriormente excluído. Pode ser acessado por qualquer organização.	Retorna histórico de atualizações do referido produto.

Fonte: o autor

5.3 Análise dos *containers* criados ao final do processo

Após a criação dos lotes e a simulação do fluxo deles ao longo da cadeia, foi verificado por meio do *Docker Desktop* como os recursos dos *containers* se comportaram. Foram analisados os seguintes parâmetros:

- Uso de CPU: indica a carga de trabalho do processador (CPU)
- Uso de Memória: indica a quantidade de memória RAM sendo ocupada pelos processos em execução, aplicativos e dados armazenados na memória temporária.
- Operações de leitura e escrita no disco rígido: mostra a quantidade de dados lidos e gravados no disco pelo *container*.
- Rede Input/Output: referente à transferência de dados (entrada e/ou saída) entre uma aplicação e uma rede.

Os resultados para as categorias de *containers* criados encontram-se abaixo:

5.3.1 Containers "dev-peer"

Containers do tipo *dev-peer* são criados pela rede durante o processo de execução do *chaincode*. Esses *containers* geralmente são usados para desenvolvimento e teste de *chaincodes* e geralmente não são executados em ambientes de produção, tendo com isso requisitos de recursos menores do que os contêineres usados para tal fim. O objetivo principal aqui é testar o código, processando as transações e atualizações na cadeia de blocos. Portanto, a utilização de CPU e leitura/escrita de disco geralmente é baixa. A maior parte do tráfego de rede desses *containers* é usada para propagar atualizações de transações na rede de pares (peers) da cadeia de blocos. O uso de memória depende do tamanho da cadeia de blocos e do número de transações que estão sendo processadas em um determinado momento.

Tabela 11 – Análise *containers* "dev-peer"

Nome Container	**Uso de CPU (em %)**	**Uso de Memória (em MB)**	**Leitura/Escrita no Disco**	**Rede I/O**
dev-peer0.produtor.mynetwork.com-supplychain-1.0.0	entre 0.00% e 0.01%	57.2 MB	0 Bytes / 2 MB	4.3 MB / 142.6 kB
dev-peer0.transportador1.mynetwork.com-supplychain-1.0.0	entre 0.00% e 0.03%	57.8 MB	0 Bytes / 2.1 MB	4.3 MB / 111.2 kB
dev-peer0.vendedor1.sampledomain.com-supplych	entre 0.00% e 0.03%	56.1 MB	0 Bytes / 2 MB	4.3 MB / 112.9 kB

ain-1.0.0				
dev-peer0.transportador2.mynetwork.com-supplychain-1.0.0	entre 0.00% e 0.02%	50.1 MB	0 Bytes / 2 MB	3.9 MB / 100.9 kB
dev-peer0.vendedor2.sampledomain.com-supplychain-1.0.0	entre 0.00% e 0.03%	51.5 MB	0 Bytes / 2 MB	4.1 MB / 115.6 kB

Fonte: o autor

Com base nos dados coletados, observamos que a utilização da CPU é muito baixa, variando entre 0% e 0,03%, indicando eficiência na execução do *chaincode* e baixo consumo de recursos do processador. O uso de memória está dentro dos limites esperados, variando entre 56,1 MB e 57,8 MB. Não há leitura significativa em disco, apenas escrita mínima, o que é típico para um *container* de desenvolvimento sem armazenamento permanente. O uso de rede é moderado, com envio e recebimento de alguns MB, principalmente para a propagação de atualizações de transações na rede de pares da *blockchain*. Com a análise, percebe-se que os dados coletados são condizentes com o proposto.

5.3.2 Containers “peers”

Os contêineres do tipo "peer" são responsáveis por manter o registro dos livros contábeis (ledgers) distribuídos e compartilhados entre os nós da rede e validação de transações e a execução de contratos inteligentes. Isso pode envolver processamento intensivo e uso significativo de recursos de hardware. Como resultado, eles normalmente têm uma atividade de CPU e rede mais alta do que outros tipos de contêineres, como "ca" ou "orderer". No entanto, o uso específico de recursos pode depender da carga de trabalho e da

configuração do ambiente. Por exemplo, se o *blockchain* estiver processando um grande número de transações, o uso da CPU e rede pode aumentar.

Tabela 12 – Análise *containers* “peers”

Nome Container	Uso de CPU (em %)	Uso de Memória (em MB)	Leitura/Escrita no Disco	Rede I/O
peer0.produtor.mynetwork.com	entre 1.13% e 3.44%	58.1 MB	2.7 MB / 248 kB	3.6 MB / 4 MB
peer1.produtor.mynetwork.com	entre 0.80% e 2.88%	63.5 MB	576 kB / 248 kB	3.2 MB / 3.2 MB
peer0.transportador1.mynetwork.com	entre 1.15% e 2.82%	65.9 MB	1.1 MB / 248 kB	3.5 MB / 3.4 MB
peer1.transportador1.mynetwork.com	entre 1.24% e 2.41%	61.7 MB	324 kB / 236 kB	3.3 MB / 3.3 MB
peer0.vendedor1.mynetwork.com	entre 0.94% e 2.89%	68 MB	1.8 MB / 248 kB	3.5 MB / 3.4 MB
peer1.vendedor1.mynetwork.com	entre 0.89% e 2.81%	59.4 MB	444 kB / 248 kB	3.4 MB / 3.3 MB
peer0.transportador2.mynetwork.com	entre 0.99% e 3.61%	59.1 MB	1.1MB / 230 kB	3.5 MB / 4 MB
peer1.transportador2.mynet	entre 1.05% e 2.99%	60.2 MB	599 kB / 240 kB	3.5 MB / 4.2 MB

work.com

peer0.vendedor2.mynetwork.com	entre 1.99% e 3.71%	59.9 MB	609 kB / 301 kB	3.7 MB / 3.5 MB
peer1.vendedor2.mynetwork.com	entre 0.90% e 3.71%	51.3 MB	454 kB / 258 kB	3.5 MB / 3.4 MB

Fonte: o autor

Com base nos dados coletados, observamos que o uso de CPU nos *containers* é relativamente baixo, variando entre 0,80% e 3,71%. O uso de memória também é adequado, variando entre 51,3 MB e 68 MB. As operações de leitura e escrita em disco são baixas, indicando um uso moderado. O volume de dados de rede está na faixa de 3,2 MB a 4,2 MB, considerado razoável para a comunicação entre os *containers*. Em geral, os *containers* apresentam um desempenho equilibrado e dentro dos parâmetros esperados.

5.3.3 Container "orderer"

Uma vez que o Orderer precisa se comunicar com outros nós na rede para processar transações, os valores abaixo podem variar dependendo da carga de trabalho e das especificações do sistema.

Tabela 13 – Análise *container* "orderer"

Nome Container	**Uso de CPU (em %)**	**Uso de Memória (em MB)**	**Leitura/Escrita no Disco**	**Rede I/O**
orderer.mynetwork.com	entre 0.00% e 0.04%	28.8 MB	192 kB / 40 kB	355.5 kB / 730.8 kB

Fonte: o autor

Com base nos dados coletados, observamos que o uso de CPU pelo *container* é muito baixo, o que é esperado quando não há transações ou processamento intenso ocorrendo. O uso

de memória também é relativamente baixo, adequado para o Orderer que tem uma carga menor. A leitura e gravação em disco são baixas, uma vez que o estado da *blockchain* é mantido em memória. O tráfego de rede é igualmente baixo, indicando que o *container* Orderer não está sobrecarregado com o tráfego de rede.

5.3.4 Containers "ca" (Certificado de Autoridade)

Os contêineres de Certificado de Autoridade (CA) geralmente não exigem muitos recursos de computação e, portanto, tendem a ter baixas taxas de uso de CPU e memória. No entanto, eles precisam ler/gravar em disco os certificados digitais e as chaves privadas que gerenciam, e a largura de banda da rede pode ser necessária para que os usuários solicitem e recebam certificados.

Tabela 14 – Análise containers "ca"

Nome Container	**Uso de CPU (em %)**	**Uso de Memória (em MB)**	**Leitura/Escrita no Disco**	**Rede I/O**
ca.produtor.mynetwork.com	entre 0.00% e 0.07%	18.9 MB	15.5 MB / 0 Bytes	3.4 kB / 0 Bytes
ca.transportador1.mynetwork.com	entre 0.00% e 0.01%	24.5 MB	812 kB / 0 Bytes	3.7 kB / 0 Bytes
ca.vendedor1.mynetwork.com	entre 0.0% e 0.05%	22.5 MB	9.8 MB / 0 Bytes	3.4 kB / 0 Bytes
ca.transportador2.mynetwork.com	entre 0.00% e 0.01%	19.5 MB	8.1 MB/ 0 Bytes	3.9 kB / 0 Bytes
ca.vendedor2.mynetwork.com	entre 0.0% e 0.03%	21.5 MB	9.5 MB / 0 Bytes	3.5 kB / 0 Bytes

Fonte: o autor

Com base nos dados coletados, observamos que o uso da CPU nos *containers* "CA" é extremamente baixo, variando entre 0,00% e 0,07%. O uso de memória também é baixo, variando entre 18,9 MB e 24,5 MB, o que é esperado para essa categoria. As operações de leitura e gravação em disco são relativamente baixas, indicando que os *containers* "CA" não realizam muitas operações intensivas de certificados no disco. O volume de dados de entrada e saída de rede é muito baixo, o que é normal para um *container* "CA" que lida com autenticação e emissão de certificados digitais.

5.3.5 Containers "couchdb"

O CouchDB é um banco de dados NoSQL orientado a documentos que armazena dados em formato JSON, e o tamanho do banco de dados e a quantidade de documentos armazenados afetam a quantidade de memória necessária. O CouchDB será o banco de dados usado nessa rede para armazenar os estados globais do ledger. O tamanho do banco de dados depende do número de transações e do número de peers que estão armazenando o estado global. Portanto, é normal que a quantidade de memória usada varie de acordo com o tamanho do banco de dados.

Tabela 15 – Análise containers "couchdb"

Nome Container	**Uso de CPU (em %)**	**Uso de Memória (em MB)**	**Leitura/Escrita no Disco**	**Rede I/O**
couchdb.peer0.produtor.mynetwork.com	entre 0.69% e 1.89%	87.9 MB	18.5 MB / 700 kB	51 kB / 60.4 kB
couchdb.peer1.produtor.mynetwork.com	entre 1.09% e 3.67%	92.5 MB	2.6 MB / 700 kB	46.1 kB / 55.5 kB
couchdb.peer0.transportador1.my	entre 0.41% e 3.15%	89.9 MB	2.2 MB / 700 kB	47.1 kB / 52.9 kB

network.com				
couchdb.peer1.transportador1.mynetwork.com	entre 0.28% e 2.24%	107.2 MB	764 kB / 700 kB	47.5 kB / 55.4 kB
couchdb.peer0.vendedor1.mynetwork.com	entre 0.78% e 2.02%	89.8 MB	912 kB / 700 kB	46.1 kB / 52.9 kB
couchdb.peer1.vendedor1.mynetwork.com	entre 1.16% e 2.90%	93.5 MB	4.2 MB / 700 kB	46.3 kB / 55.2 kB
couchdb.peer0.transportador2.mynetwork.com	entre 0.95% e 2.94%	101.1 MB	882 kB / 700 kB	50.1 kB / 55.2 kB
couchdb.peer1.transportador2.mynetwork.com	entre 1.15% e 3.59%	95.3 MB	1.1 MB / 700 kB	49.3 kB / 55.3 kB
couchdb.peer0.vendedor2.mynetwork.com	entre 1.26% e 3.34%	91.7 MB	2.5 MB / 700 kB	46.5 kB / 55.1 kB
couchdb.peer1.vendedor2.mynetwork.com	entre 0.98% e 3.67%	102.3 MB	2.1 MB / 700 kB	46.5 kB / 55.1 kB

Fonte: o autor

Com base nos dados coletados, observamos que o uso de CPU nos *containers* "CouchDB" varia entre 0,28% e 3,67%, o que está dentro dos limites esperados. O uso de memória varia entre 87,9 MB e 107,2 MB, sendo o tipo de container com maior uso de memória devido à natureza do banco de dados orientado a documentos. As operações de leitura e gravação em disco estão dentro do normal, variando entre 764 kB e 18,5 MB,

dependendo das transações e solicitações do CouchDB. O volume de dados de rede é moderado, com comunicação necessária para sincronização e replicação de dados entre os peers.

5.3.6 Container "CLI"

Tabela 16 – Análise *container* "CLI"

Nome Container	Uso de CPU (em %)	Uso de Memória (em MB)	Leitura/Escrita no Disco	Rede I/O
cli	entre 0.00% e 0.02%	87.4 MB	31.8 MB / 0 Bytes	1.3 MB / 931.8 kB

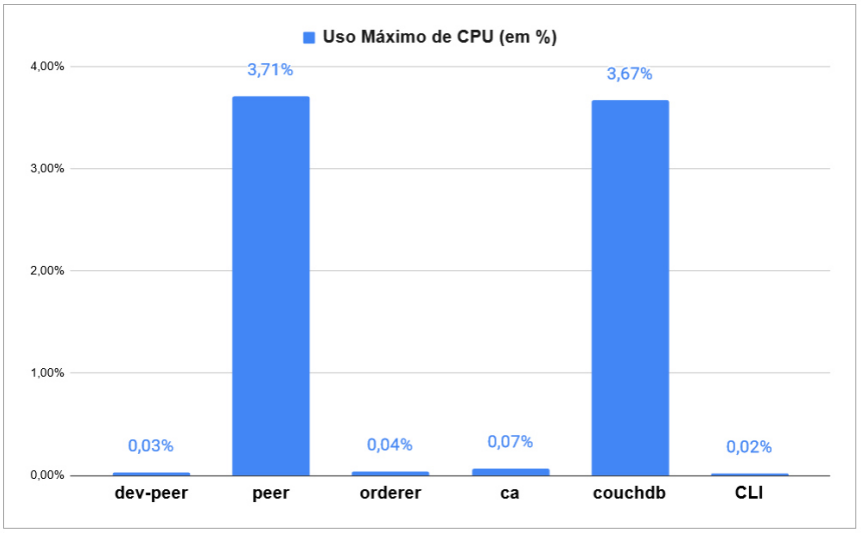
Fonte: o autor

Com base nos dados coletados, observamos que o uso de CPU no *container* "CLI" é muito baixo, indicando poucos processos intensivos em recursos. O uso de memória é baixo, com 87,4 MB, adequado para as tarefas de gerenciamento e execução de comandos. O uso do disco é principalmente para leitura, consultando informações da blockchain e escrevendo logs. Os valores de entrada e saída de rede indicam atividade moderada, com troca de dados entre o container "CLI" e outros componentes da rede, em quantidades pequenas, como esperado em uma interação via linha de comando.

5.3.7 Comparativo de métricas

Nesta seção, foram feitas comparações dos parâmetros analisados, sendo avaliados por cada container.

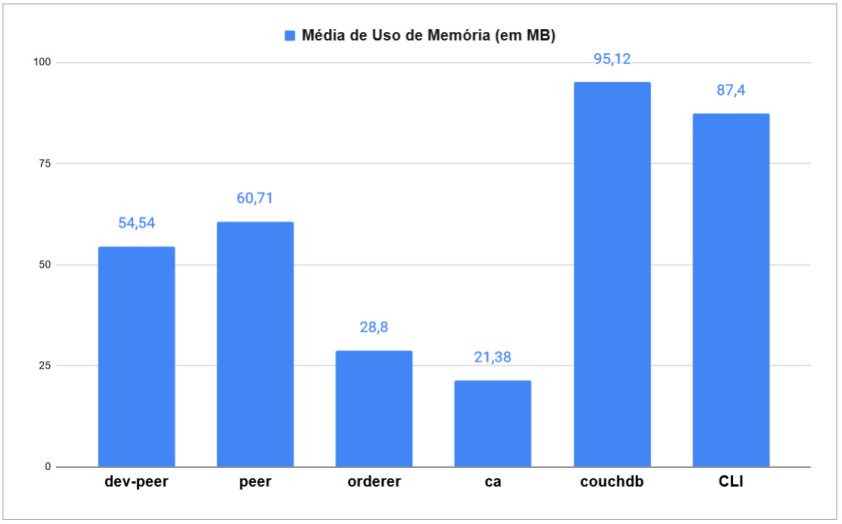
Figura 4 – Uso máximo de CPU (em %)



Fonte: o autor

No gráfico da Figura 4, percebemos que os dois tipos de container que mais usam de Recursos de CPU são os containers do tipo *peer* e *couchdb*. O uso específico de recursos e registro de dados pode depender da carga de trabalho e da configuração do ambiente, podendo aumentar com o aumento do número de transações. A análise desse gráfico é condizente com o esperado, tendo em vista que esses dois containers são os que possuem os maiores processamentos, logo utilizam mais recursos da CPU.

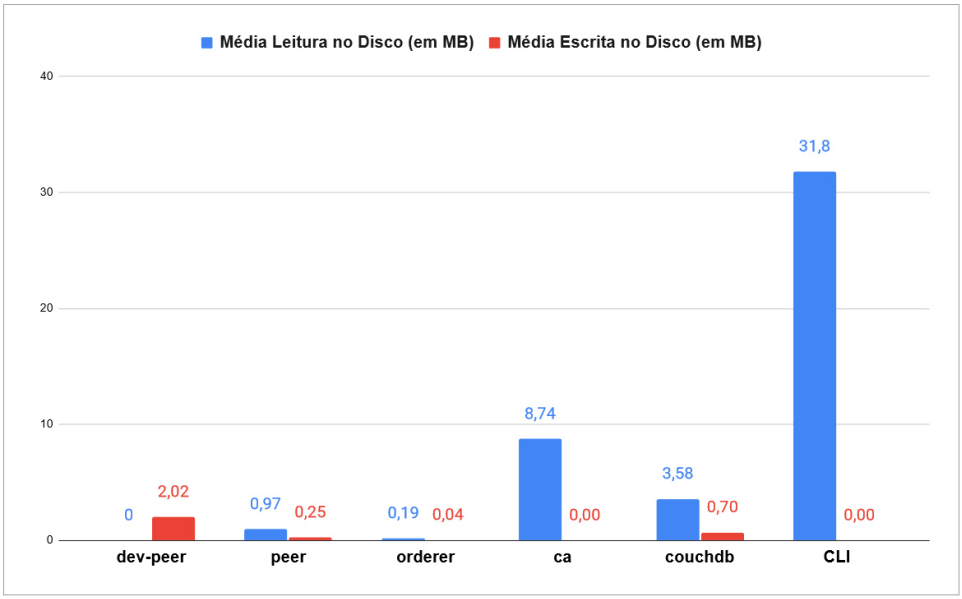
Figura 5 – Média de Uso de Memória (em MB)



Fonte: o autor

Analisando o gráfico da Figura 5, percebemos que o tipo de container com maior média do Uso de Memória são os containers do tipo *CouchDB*. Por se tratarem de banco de dados orientados a documentos, era esperado que esse tipo de *container* possuísse a maior média de uso de memória, e os resultados dos testes realizados comprovaram isso. Em seguida, observamos o *container* do tipo *CLI* com o uso de memória também mais elevado em relação ao demais, mas dentro do normal para um *container* que deve ser adequado para as tarefas de gerenciamento e execução de comandos. Os tipos *dev-peer* e *peer* possuem uma média de uso dentro do esperado, com o consumo de memória dependendo da complexidade e do tamanho do *chaincode* em execução e da quantidade de tarefas necessárias em execução.

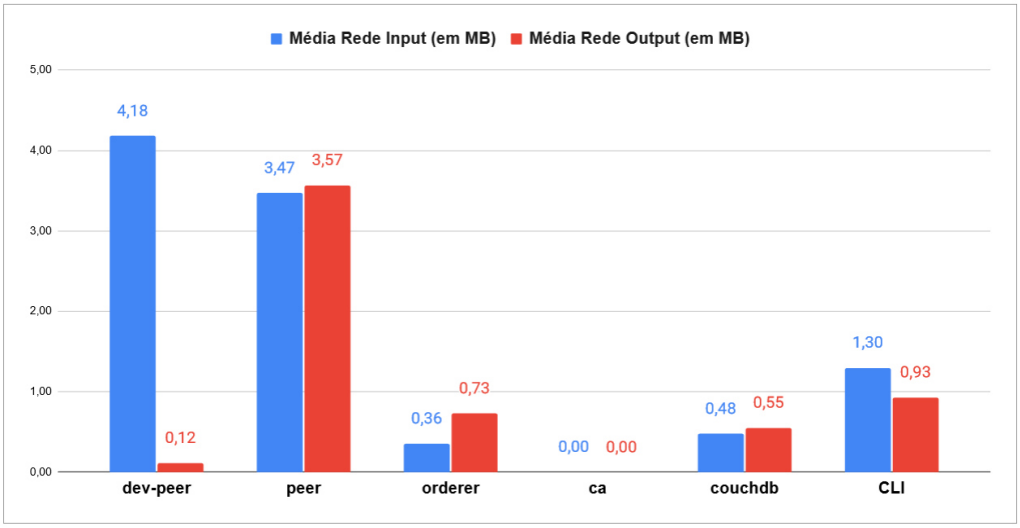
Figura 6 – Volume Médio de Leitura e Escrita de dados no Disco (em MB)



Fonte: o autor

Analisando o gráfico da Figura 6, observamos que os maiores volumes de leitura no disco são registrados nos containers do tipo *CLI* e *CA*. No caso do *CLI*, deve-se ao fato de tratar-se de um container que precisa receber as operações passadas via linha de comando, portanto o volume de leitura acaba se destacando. Já o container *CA* apresenta grande também um considerável volume de leitura pois, devido à emissão dos certificados de autoridade, esse informação precisa ser lida principalmente pelos containers do tipo peer, para eles conseguirem ter a permissão para as operações. Já o volume de Escrita no disco é mais destacado no dev-peer, devido à sua natureza de executar os métodos do contrato inteligente. No geral, observamos que o volume de leitura foi maior que o de escrita. Isso se deve porque o número de transações (e por consequência, de registros) não foi tão alta. Mas quanto maior o número de transações, maior seria o volume de dados de escrita.

Figura 7 – Volume Médio de Input e Output de dados na Rede (em MB)



Fonte: o autor

Paralelo ao volume médio de dados de leitura e escrita no disco, podemos analisar o volume médio de dados de rede, tanto de *input* quanto de *output* de dados, no gráfico da Figura 7. Foi observado que a média do volume de dados de *input* na rede do container tipo *dev-peer* foi o mais destacado, o que é condizente com o observado no gráfico da figura 6, onde esse mesmo tipo de container apresentava mais destacado o volume de dados de escrita no disco. Continuando as comparações dos gráficos da Figura 6 e da Figura 7, observamos que, no geral, os dados de Escrita no Disco e Input na Rede são correlacionados entre si.

A análise comparativa dessas métricas validam o funcionamento dos princípios básicos de funcionamento da *blockchain* nessa rede de pequena escala. Logo, os princípios básicos também são válidos em uma rede de grande escala, nos demonstrando a viabilidade do uso desse tipo de tecnologia.

5.4 Exemplo de Aplicabilidade

No trabalho [SILVA, 2020] *Método de Rastreabilidade de Produtos agrícolas com a utilização de Blockchain*, foi realizado um estudo do estado da arte da gestão da cadeia de suprimentos, além do desenvolvimento de um método baseado em *blockchain*, e aplicação do método na rastreabilidade de produtos agrícolas. Para validação do método foi definida a erva-mate na cadeia. Analisaremos esse trabalho de [SILVA, 2020] sob a ótica do modelo de arquitetura proposta nesta presente obra.

Em [SILVA,2020], foi discorrido sobre os pontos críticos da cadeia produtiva da erva-mate, possuindo cinco principais entidades:

(i) o Produtor, responsável por gerir os dados de quais insumos foram inseridos, quais as mudas utilizadas, a localização da plantação e se essa área é uma área com

sombreamento ou não, fator importante pois afeta a qualidade e também define qual será o tipo da erva, e também como está sendo feito o armazenamento, atentando sempre para duas variáveis de ambiente durante esse processo que são a temperatura e luminosidade, pois afetam diretamente a qualidade da erva-mate;

(ii) a Transportadora, responsável por fornecer a rota, condições de armazenamento, tempo de transporte e datas de início e fim de percurso realizado em dois momentos, no envio da erva para a Indústria e também no envio do produto final para a Comercialização;

(iii) a Indústria, responsável por fornecer dados de beneficiamento da erva, como por exemplo a secagem e fragmentação;

(iv) a Comercialização, fornecedora do tempo de prateleira, lote e prazo de validade da erva-mate industrializada;

(v) o Consumidor Final do produto, permite a rastreabilidade dos demais processos realizados pelas entidades que estão dispostos anteriormente a eles.

A partir do exposto e tendo por base a Figura 2, podemos considerar a entidade **"(i) Produtor"** como sendo análoga à entidade *Produtor* de nosso trabalho, que estaria invocando a função createProduct para o cadastro das informações de produto erva-mate. A entidade **"(ii) Transportadora"** seria análoga à entidade *Transportador*, que estaria invocando a função transferProduct para atualizar os dados do Produto. A entidade **"(iii) Indústria"** seria uma empresa classificada como *Vendedor* e que possuiria um modelo de negócio B2B, ou seja, de venda direta para outra empresa (no exemplo de [SILVA, 2020] seria uma venda para **"(iv) Comercialização"**). Para isso, o produto sairia de "*Vendedor*" e voltaria para "*Transportador*". Já a entidade **"(iv) Comercialização"** também seria uma empresa classificada como *Vendedor* e que possuiria um modelo de negócio B2C, ou seja, de venda direta para **"(v) Consumidor Final"**. Ao fazer isso, o *Vendedor* **"(iv) Comercialização"** estaria invocando a função deleteProduct indicando que o produto chegou ao final de sua cadeia. A alteração dos dados do tipo *Data* e do tipo *CNPJ* dará a noção de movimentação do produto ao longo da Cadeia, bem como informações acerca do histórico.

6 CONCLUSÃO

Neste trabalho, apresentamos a proposta de uma arquitetura para um sistema de rastreabilidade de produtos em uma cadeia de distribuição logística, onde os dados de registro desses produtos são armazenados em uma *blockchain* privada, do tipo Hyperledger Fabric. A solução foi desenvolvida considerando-se uma cadeia de logística simplificada, constituída por apenas cinco organizações, chamadas de forma genérica de *"Produtor", "Transportador1", "Vendedor1", "Transportador2" e "Vendedor2"*. Optou-se por construir essa rede de forma genérica, como uma forma de criar um modelo simplificado que poderá ser expandido para cadeias mais complexas e/ou específicas, com diversos participantes envolvidos na fabricação, distribuição e venda dos produtos. Todos seriam registrados na *blockchain*. Cada produto é abstraído nessa rede como um "lote de informação", ou seja, um conjunto de informações que caracterizam aquele produto. No contrato inteligente, foram escritas regras de negócio que estabelecem o funcionamento do fluxo da cadeia . O objetivo disso era simular o fluxo real da cadeia de distribuição logística, desde a fabricação do produto até a posterior comercialização do mesmo a um consumidor final, passando antes pela etapa de revenda do produto.

Os testes realizados na rede dos *containers* validaram os princípios básicos discutidos para a construção do modelo de arquitetura proposto. Também foi validado o uso do Hyperledger Fabric como uma solução viável para se aplicar nesse tipo de situação, bem como o funcionamento dos métodos do chaincode. Os testes realizados na rede também mostraram que ela era uma rede segura. Com isso, podemos concluir que o desenvolvimento da arquitetura valida o modelo proposto e cumpre com um dos objetivos deste trabalho que é servir de modelo para outros trabalhos do gênero. As simulações feitas com os produtos genéricos e o posterior estudo de caso da erva-mate, apresentado em [SILVA, 2020], também demonstraram a utilização do nosso modelo de arquitetura em casos mais específicos, demonstrando a viabilidade e flexibilidade do uso deste trabalho como modelo para a construção de aplicações mais específicas. Com isso, os objetivos inicialmente postulados para esse trabalho são atingidos.

Entre as dificuldades encontradas durante a realização deste trabalho, destacam-se a maior parte da documentação e trabalhos encontrados sobre uso de *blockchain* em sistemas de rastreabilidade serem desenvolvidos a partir do Ethereum, o que exigiu uma curva de aprendizado maior para desenvolver o projeto com Hyperledger Fabric. Também pode-se citar

como dificuldade encontrar referências de trabalhos com o desenvolvimento de códigos de contrato inteligente para poder servir como parâmetro. Por fim, pode ser destacada como dificuldade a realização dos testes via linha de comando, pois, embora o Docker Desktop tenha sido uma aplicação que tenha ajudado na avaliação das métricas, a execução dos comandos neste trabalho precisou ser realizada a partir das linhas de comando, deixando a execução pouco prática.

6.1 Contribuições deste trabalho

A principal contribuição deste trabalho para a comunidade acadêmica é a construção do modelo de um sistema de rastreabilidade usando o Hyperledger Fabric. Também ao longo desta obra, é feito um estudo detalhado sobre o Hyperledger que pode servir como base para estudos futuros. A maior parte dos trabalhos pesquisados continham um desenvolvimento na rede Ethereum, e o desenvolvimento prático realizado aqui na rede Hyperledger representa um importante diferencial nesta obra.

A construção da arquitetura foi desenvolvida de tal forma que possa servir como base para que aplicações mais complexas (com a participação de mais organizações e agentes). A ideia de construir um modelo simplificado como o mostrado aqui serve ao propósito de criar esse template para ser explorado de forma mais específicas em outras aplicações.

Também pode ser destacado como contribuição uma análise sobre a forma como os modelos de negócios das empresas influencia na cadeia logística (algo não muito explorado nos Trabalhos Relacionados para esta obra), bem como uma discussão sobre como este projeto impacta no desenvolvimento dos negócios do tipo ESG (*environmental*, *social* and *governance*).

6.2 Trabalhos Futuros

A criação de um modelo de arquitetura foi desenvolvida com sucesso nesta obra. A partir dos resultados observados, podemos em trabalhos futuros explorar novas possibilidades a partir do que foi desenvolvido neste trabalho.

Podemos destacar como possível trabalho futuro o desenvolvimento da proposta de um modelo de arquitetura que agregue sensores que utilizam tecnologia da Internet das Coisas (IoT – *Internet of Things*). Como colocado em [SILVA, 2020], a rastreabilidade pode ser considerada com base na integração entre a tecnologia *blockchain* e a Internet das coisas, a fim de melhorar a segurança e a qualidade dos produtos alimentícios e, ao mesmo tempo, diminuir a possibilidade de más condutas, como fraude, corrupção, adulteração e informações distorcidas.

A coleta de dados realizada ao longo desta cadeia e armazenada na *Blockchain* também pode ser algo de bastante interesse para as empresas envolvidas. A exploração desses dados para o desenvolvimento de estratégias na área de Inteligência de Mercado, pode representar uma fonte de melhorias aos modelos de negócios dessas empresas, bem como a oportunidade da exploração de novos mercados e geração de novas receitas. Além disso, essa coleta de dados pode servir como análise para a cadeia logística, podendo ser aproveitada para a melhoria da mesma.

Por fim, podemos destacar como possibilidade futura a melhoria da forma de leitura dos dados dos produtos dessa cadeia. Para isso, o desenvolvimento de aplicações que possam realizar a leitura de *QR Code* (que conteria o ID do Produto) representaria uma simplificação na forma de acesso de dados desses produtos.

REFERÊNCIAS

BALLOU, Ronald H. Gerenciamento da cadeia de suprimentos: planejamento, organização e logística empresarial. Porto Alegre: Bookman, 2001.

BORGES, Marcelo Alexandre. Gerenciamento da Cadeia Logística: oportunidades de criação de valor através da logística de distribuição. Gestão e Desenvolvimento, v. 3, n. 1, p. 49-56, 2006.

MATTOS, Marina Guimaraes. Gestão de Riscos em Cadeias de Suprimentos: Estudo exploratório sobre a experiência Brasileira. 2011.

BEULENS, A., Broens, D., Folstar, P. and Hofstede, G. Food safety and transparency in food chains and networks Relationships and challenges. Food Control 16(6), 481-486 (2005).

OPARA, L. U. Traceability in agriculture and food supply chain : a review of basic concepts, technological implications, and future prospects. Food, Agriculture & Environment 1(1), 101-106 (2003).

WILSON, T. P., & CLARKE, W. R. Insights from industry food safety and traceability in the agricultural supply chain : using the Internet to deliver traceability. Supply Chain Management 3(3), 127-133 (1998).

KHER, S., Frewer, L., Jonge, J., Wentholt, M., Davies, O., Luijckx, N. and Cnossen, H. Experts ' perspectives on the implementation of traceability in Europe. British Food Journal 112(3), 261-274 (2010).

AUNG, M. and CHANG, Y. Traceability in a food supply chain : Safety and quality perspectives. Food Control 39, 172-184 (2014).

NAKAMOTO, S. Bitcoin: A Peer-to-Peer Electronic Cash System (2008).

GREVE, Fabíola et al. Blockchain e a Revolução do Consenso sob Demanda. In: Minicursos do Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos. Ed. 36. São Paulo: Sociedade Brasileira de Computação, 2018.

COSTA, A. B., Rocha, A. R. D., Delicato, F. C., & Souza, J. N. D. (2020). Balanceamento de carga na borda da rede usando blockchain das coisas.

LUSTOSA, Leonardo Junqueira; DE MESQUITA, Marco Aurélio; OLIVEIRA, RODRIGO J. Planejamento e controle da produção. Elsevier Brasil (2008).

SILVA, Hugo Leonardo Petla. Método de rastreabilidade de produtos agrícolas com a utilização de blockchain. Dissertação (Mestrado em Computação Aplicada) - Universidade Estadual de Ponta Grossa, Ponta Grossa, 2020.

SALMAN, T., Zolanvari, M., Erbad, A., Jain, R., and Samaka, M. Security services using blockchains: A state of the art survey. IEEE Communications Surveys Tutorials, vol. 21, no. 1, pp. 858–880, Firstquarter 2019.

SANKAR, L. S.; SINDHU, M.; SETHUMADHAVAN, M. Survey of consensus protocols on blockchain applications. In 2017 4th International Conference on Advanced Computing and Communication Systems (ICACCS). IEEE, 2017, pp. 1–5.

AVRAMENKO, Alexey. Ethereum and smart contracts basics. 2017. Disponível em: https://medium.com/@olxc/ethereum-and-smart-contracts-basics-e5c84838b19. Acesso em: 10 de maio de 2023.

MIERS, Charles Christian et al. Blockchains com Hyperledger: conceitos, instalação, configuração e uso. Sociedade Brasileira de Computação, 2020.

MIERS, C., Koslovski, G., Pillon, M., Simplicio, M., Carvalho, T., Rodrigues, B., and Battisti, J. Análise de Mecanismos para Consenso Distribuído.

SCHRÖDER, Bruna. "Rastreamento de medicamentos com identificação por radiofrequência e armazenamento em Blockchain." (2019).

ARANDA, Rodrigo Spessoto. Proposta de arquitetura para a rastreabilidade de medicamentos nas cadeias de suprimentos hospitalares utilizando Blockchain. 2022. Tese de Doutorado. Universidade de São Paulo.

THAKKER, U., Patel, R., Tanwar, S., Kumar, N., and Song, H. Blockchain for Diamond Industry: Opportunities and Challenges. IEEE Internet of Things Journal, vol. 8, no. 11, pp. 8747-8773, 1 June1, 2021, doi: 10.1109/JIOT.2020.3047550.

COSTA, Alexandre Barroso. MECAGERAN - Mecanismo de Gerenciamento de Recursos em Ambiente de Névoa; 2021; Dissertação (Mestrado em Engenharia de Teleinformática) Universidade Federal do Ceará.

CLOUTIER, R., Muller, G., Verma, D., Nilchiani, R., Hole, E., & Bone, M. The concept of reference architectures. Systems Engineering, 13(1), 14-27.

CHINAGLIA, R. Cointelegraph Brasil. 2021. Disponível em: https://cointelegraph.com.br/news/carrefour-expands-blockchain-use-for-fresh-food-tracking. Acesso em: 30 de junho de 2023.

BRAGADO, L. Época Negócios. 2022. Disponível em: https://epocanegocios.globo.com/Tecnologia/noticia/2022/05/renner-lanca-calcas-jeans-rastreadas-por-blockchain.html. Acesso em: 2 de julho de 2023.

Hyperledger Whitepaper. Disponível em: https://docs.google.com/document/d/1Z4M_qwILLRehPbVRUsJ3OF8Iir-gqS-ZYe7W-LE9gnE/pub. Acesso em: 2 de julho de 2023.

Hyperledger Architecture. Introduction to Hyperledger Business Blockchain Design Philosophy and Consensus. 2017. Disponível em: https://www.hyperledger.org/wp-content/uploads/2017/08/Hyperledger_Arch_WG_Paper_1_Consensus.pdf. Acesso em: 2 de julho de 2023.