

USO DE BLOCKCHAIN E CONTRATOS INTELIGENTES NA GESTÃO DO CICLO DE VIDA DE CARROS

Yuri Matheus Hartmann, Prof. Marcel Hugo – Orientador

Curso de Bacharel em Ciência da Computação
Departamento de Sistemas e Computação
Universidade Regional de Blumenau (FURB) – Blumenau, SC – Brasil

yhartmann@furb.br, marcel@furb.br

Resumo: A alta burocracia que se encontra em nossa sociedade, gera situações de desencontro de informações e dados pulverizados entre sistemas sem a confiança nos dados. Isso acontece em várias áreas da sociedade, uma delas é no mundo automotivo, que para se possuir um carro, uma série de eventos é enfrentada, podendo gerar informações divergentes e pulverizadas em diversos sistemas. Com a tecnologia blockchain privada Hyperledger Fabric e juntamente com os contratos inteligentes, o mundo automotivo foi escolhido para testar essas tecnologias, mais especificamente o ciclo de vida de um carro. Utilizando os benefícios da blockchain privada e da modularidade dos contratos inteligentes, se criou um protótipo para avaliar se a solução seria viável e atenderia possíveis situações reais. Com isso se chegou em um protótipo que se conclui ser viável no mundo real, centralizando as informações e com a garantia dos dados sem um único ponto de falha.

Palavras-chave: Blockchain. Contratos inteligentes. Hyperledger Fabric. Ciclo de vida de um carro

1 INTRODUÇÃO

A burocracia pode trazer consequências no mundo, como descreve Motta (2017, p. 4), “burocracia é poder, controle e alienação.”, ou seja, o grande volume de burocracia acaba prejudicando a nossa sociedade, que abre margem para cada vez mais softwares poderem atender as pessoas e simplificar suas vidas. No mundo dos carros não é diferente. Enfrentamos diversas situações ao possuir um carro, como documentação, emplacamento, seguro, financiamento, revisões, manutenções, acidentes eventuais. Todas elas geram muita informação, que é armazenada em sistemas de informação, porém na sua grande maioria que não estão integrados, mantendo informações específicas sobre um veículo de forma pulverizada em várias bases de dados. Caso as montadoras, oficinas, concessionárias, seguradoras e outras empresas envolvidas no ciclo de vida de um carro fossem participantes de algum mecanismo de compartilhamento e troca de informações, essas informações estariam completas, potencializando seu uso e simplificando os processos.

O termo *blockchain* teve sua primeira aparição em 2009 por Satoshi Nakamoto, num artigo denominado Bitcoin: A Peer-to-Peer Electronic Cash System, que introduzia uma moeda virtual descentralizada que se chama Bitcoin. O que mais chamou atenção na criação de Nakamoto, foi a arquitetura *blockchain* (NAKAMOTO, 2009).

A *blockchain* é uma espécie de livro razão, que possui propriedades fortes, sendo elas: a descentralização, consensualidade e a assinatura das transações. Os registros são de confiança e salvos de forma descentralizada, onde as transações ocorrem havendo concordância e segurança entre todos da rede sem a necessidade de um ponto centralizado. Essa peculiaridade é garantida pelo modo de funcionamento ser *peer-to-peer* (ponto a ponto), que certifica que o registro e sua validação na rede aconteçam de forma autônoma. A consensualidade funciona como uma espécie de corrente que une as transações, que por sua vez, essas transações são juntadas formando um bloco. Os blocos são o agrupamento das transações e sua assinatura, sincronizando com os demais participantes da rede. A assinatura dos blocos é gerada formando um *hash*, que é calculado utilizando um método que gera uma identificação única. Caso algum dado seja alterado, o *hash* gerado será diferente. Após a assinatura, os dados são salvos no livro razão (NARAYANAN *et al.*, 2016).

Com as características das *blockchains* pode-se criar diversas aplicações com regras de negócios diferentes e personalizadas para a área de atuação e assim nascem os contratos inteligentes, que foram demarcados pela primeira vez por Szabo (1997, p. 1) como "um conjunto de promessas acertadas em um encontro de mentes". Isso significa que pode ser executado e validado por todos de modo que não precise de intermediários confiáveis já que todos da rede podem validar a transação antes dela ser aceita, evitando fraudes e dados divergentes. O contrato inteligente é basicamente um código, chamado de *chaincode*, com regras que são executadas e armazenadas dentro da *blockchain*, que pode ser rede Ethereum, Hyperledger Fabric, dentre outros.

O *chaincode* é onde está definido a regra que as transações devem seguir. Nele estão os cálculos, validações, sendo não apenas executado por um participante da rede e sim por vários ou até mesmo por todos. Assim garante-se que tudo foi executado e validado de forma igual, sem apresentação de manipulação dos dados ou adulteração no código (WOOD, 2014).

A partir dos contratos inteligentes juntamente com a *blockchain* pode-se criar e automatizar diversos processos dentro da nossa sociedade, desde os mais simples, como a circulação de dinheiro virtual, que hoje se conhece como criptomoedas, até os mais complexos com contratos entre pessoas e empresas. Porém quando se pensa em redes de *blockchain* com contratos inteligentes que cuidariam do ciclo da vida de um carro, desde sua fabricação, até revisões e transferências, é evidente que a *blockchain* não pode ser aberta para qualquer indivíduo ou entidade: apenas montadoras podem adicionar veículos, apenas oficinas podem cadastrar manutenções e assim por diante. Por isso existem as redes *blockchain* privadas, como a Hyperledger Fabric, onde os participantes da rede precisam de uma espécie de ingresso válido para poder participar. Essa rede privada garante que as informações e execuções de contratos apenas sejam disponibilizados para participantes com permissão.

Com a tecnologia *blockchain* juntamente com contratos inteligentes pode-se criar uma rede para gerir diversas áreas da nossa sociedade. A área escolhida neste trabalho para utilizar e se beneficiar dessas tecnologias é o mundo automotivo, mais especificamente o ciclo de vida de um carro. Diante do exposto, neste trabalho foi desenvolvido um protótipo para gerenciar o ciclo de vida de um carro utilizando *blockchain* privada e contratos inteligentes.

2 FUNDAMENTAÇÃO TEÓRICA

Nessa seção serão apresentados os conceitos relacionados à *blockchain* e conceitos técnicos das ferramentas usadas incluindo a plataforma Hyperledger Fabric.

2.1 BLOCKCHAIN

Para Tapscott (2016), *blockchain* é uma tecnologia que tem sido cada vez mais utilizada em diferentes setores, desde o financeiro até o de saúde e logística. A tecnologia ganhou destaque por sua aplicação em criptomoedas, mas suas aplicações vão muito além disso.

De acordo com Narayanan *et al.* (2016), *blockchain* é uma tecnologia que permite o registro de transações de forma segura e descentralizada. Ela funciona como um livro de contabilidade digital que registra as transações em blocos interconectados. Esses blocos são protegidos com criptografia e distribuídos em vários computadores, garantindo a segurança e a integridade das transações. Isso provê características únicas: descentralização, transparência e imutabilidade.

Ela é descentralizada, o que significa que não é controlada por uma única entidade ou autoridade. Em vez disso, ela é mantida por uma rede de nós. Um nó é um computador que roda a *blockchain* e executa contratos inteligentes e possui uma cópia completa do livro razão salvo que valida as transações e mantém uma cópia do registro. Tudo isto mantém a rede mais resistente a falhas e ataques maliciosos. Outra característica que destaca Narayanan *et al.* (2016) é que a rede *blockchain* possui transparência. Como todos os nós da rede possuem uma cópia do registro, qualquer pessoa pode verificar a autenticidade das transações. Isso pode ser particularmente útil em setores como o de logística, onde a rastreabilidade é essencial para garantir a integridade da cadeia de suprimentos. Além disso, a tecnologia *blockchain* é imutável, o que significa que as transações registradas não podem ser alteradas ou excluídas garantindo a integridade do registro e a segurança das transações.

Para Tapscott (2016), a tecnologia *blockchain* tem várias aplicações em diferentes setores. Uma das aplicações mais conhecidas é no mercado de criptomoedas, sendo a base do Bitcoin e outras criptomoedas. Porém, suas aplicações vão muito além. Por exemplo no setor financeiro, pode ser usada para reduzir custos, aumentar a eficiência e melhorar a segurança das transações. Empresas como a JP Morgan e a Goldman Sachs já estão utilizando a tecnologia para transações internacionais e liquidação de títulos.

Outro setor que pode se beneficiar da tecnologia *blockchain* é o de saúde, como afirma Tapscott (2016). A tecnologia pode ser usada para criar um registro médico eletrônico seguro e descentralizado, permitindo que os pacientes tenham mais controle sobre seus dados médicos e garantindo que os dados sejam precisos e seguros. A tecnologia também pode ser usada em setores como o de logística e transporte, permitindo a rastreabilidade dos produtos e melhorando a eficiência da cadeia de suprimentos.

2.2 CONTRATOS INTELIGENTES

Os contratos inteligentes, também conhecidos como *smart contracts*, representam uma das aplicações mais relevantes da tecnologia *blockchain* como explica Swan (2015, p. 16). Eles consistem em programas que são executados dentro da *blockchain* e foram concebidos para serem autônomos e confiáveis, o que possibilita a eliminação de intermediários e a redução de custos.

Contratos inteligentes são programas de computador que são armazenados em uma *blockchain* e executados quando forem invocados. Eles são compostos por uma série de regras e condições que são codificadas. Os contratos inteligentes rodam em uma rede *blockchain*, a rede mais famosa é a rede Ethereum, uma plataforma *blockchain* que suporta a criação de aplicativos descentralizados e contratos inteligentes. A plataforma permite que os desenvolvedores criem contratos inteligentes personalizados e os publiquem em um *blockchain* público. (WOOD, 2014, p. 1)

Os contratos inteligentes têm várias aplicações em diferentes setores como explica Laurence (2017, p. 131). Uma das aplicações mais populares é no setor financeiro, onde eles podem ser usados para a criação de instrumentos financeiros descentralizados, como empréstimos, títulos e derivativos.

2.3 HYPERLEDGER FABRIC

De acordo com a Hyperledger (2023), o Hyperledger Fabric é uma plataforma *blockchain* de código aberto desenvolvida pela Linux Foundation para criar soluções de negócios para empresas. É projetado para ser flexível, escalável e seguro, com recursos que permitem o desenvolvimento de aplicativos *blockchain* personalizados para atender às necessidades específicas de cada empresa.

O Hyperledger Fabric tem várias características que o tornam uma opção atraente para empresas que desejam implementar soluções *blockchain* em seus negócios. Alguns desses recursos incluem a modularidade, pois é construído em módulos que permitem que as empresas personalizem sua solução de *blockchain* de acordo com suas necessidades específicas. Também possui a característica de ter consenso de múltiplos canais, o que permite que as empresas criem múltiplos canais de comunicação dentro da rede *blockchain*, ajudando a garantir a privacidade e a segurança dos dados segundo a Hyperledger (2023).

Os contratos inteligentes personalizados são chamados de *chaincode* dentro da Hyperledger Fabric (2023). Eles são outra forte vantagem para que as empresas desenvolvam seus próprios contratos inteligentes, pois atendem às necessidades específicas de seus negócios e ainda aproveitam da escalabilidade do Hyperledger Fabric. Esta rede foi projetada para ser altamente escalável, permitindo que as empresas expandam sua solução de *blockchain* conforme sua base de usuários cresce. Segundo Hyperledger (2023), as vantagens trazidas pela rede são de segurança, que contém recursos que garantem a privacidade e a segurança dos dados, e de customização, que permite que as empresas personalizem sua solução de *blockchain* de acordo com suas necessidades específicas.

O Hyperledger Fabric se diferencia de outras *blockchains* por ser autorizado e privado. *Blockchains* públicas que são abertas e sem permissão, permitem que identidades desconhecidas, ou seja, qualquer pessoa participe da rede e usualmente usam protocolos como "prova de trabalho" para validar transações e proteger a rede. Em contrapartida os membros de uma rede Hyperledger Fabric se inscrevem por meio de um Provedor de Serviços de Associação confiável. Dessa forma, o Hyperledger Fabric permite que apenas entidades autorizadas participem da rede, o que aumenta a segurança e a privacidade dos dados transacionados (HYPERLEDGER, 2023).

2.3.1 COMPONENTES DO HYPERLEDGER FABRIC

O Hyperledger Fabric é construído na linguagem Go usando como protocolo de comunicação o Google Remote Procedure Call (gRPC). Nesta seção serão apresentados os principais componentes de uma rede.

Um importante componente dentro do Hyperledger Fabric é o de organizações, ou seja, cada instituição ganha uma chave de identificação fornecida pelo provedor de serviços de associação (Membership Service Provider - MSP). Essas organizações podem ter vários membros com funções diferentes conectados dentro da *blockchain* – funções serão explicadas na seção 2.3.2. Essa ferramenta tem como objetivo gerenciar as credenciais emitidas por cada nó e fazer a validação das mesmas, como esclarece Androulaki *et al.* (2018, p. 8).

O *chaincode*, nome dado para contratos inteligentes dentro do Hyperledger Fabric, é executado em um ambiente isolado, dentro de um interpretador rodando em Docker, trazendo a vantagem de poder ter várias linguagens suportadas. Atualmente se pode escrever um *chaincode* nas linguagens Golang, Java e Node.

Como cada execução do *chaincode* executa isoladamente, há um melhor controle sobre o seu ciclo de vida, ou seja, pode-se iniciar, parar, abortar e reiniciar de forma mais fácil. A comunicação entre o nó e o *chaincode* é via gRPC, por isto essa ligação é independente da linguagem em que o *chaincode* foi implementado.

Uma política de endosso permite que o *chaincode* determine as organizações necessárias para uma transação ser aceita. Essa política usa uma linguagem lógica simples baseada em conjuntos, como no exemplo de Androulaki *et al.* (2018, p. 5): "A e B ou B e C", sendo A, B e C organizações da rede Hyperledger Fabric.

Outro conceito importante é o canal. Canal é onde acontece a comunicação entre os nós. Segundo Hyperledger (2023) um canal no Hyperledger Fabric é uma rede de comunicação privada que permite a troca de transações de forma confidencial entre dois ou mais membros específicos da rede, funcionando como uma sub-rede isolada. Cada canal mais o seu contrato inteligente formam o seu livro-razão, ou seja, dentro de uma rede Hyperledger Fabric pode-se ter vários livros razões e diferentes canais (HYPERLEDGER, 2023).

2.3.2 ATORES DA REDE

Dentro da Hyperledger Fabric existem três atores: o cliente, os nós pares e o serviço de ordenação (HYPERLEDGER, 2023). Na seção 2.3.3, será explicado como cada ator interage dentro da rede *blockchain*.

O cliente é o que submete as transações, ajudando na orquestração dos dados e na transmissão das transações.

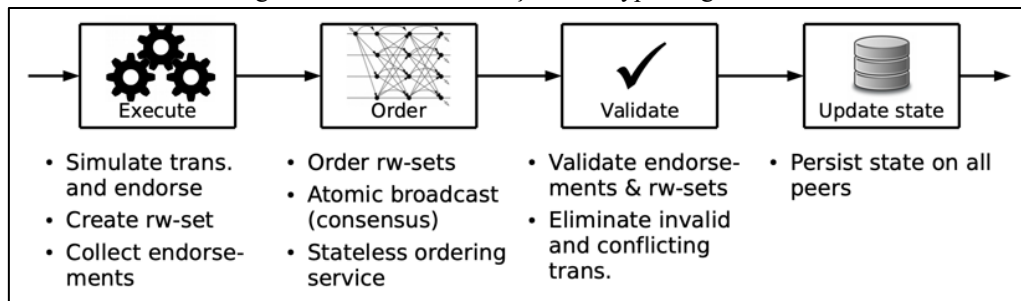
O nó par, ou simplesmente nó, é um componente essencial de uma rede *blockchain* do Hyperledger Fabric. Ele tem a responsabilidade de gerenciar os livros-razão, executar os *chaincodes*, entre outros. Vale ressaltar que um livro-razão registra de forma imutável todas as transações geradas pelos *chaincodes*.

O serviço de ordenação tem como função receber as transações, fazer a ordenação, agrupá-los em bloco e enviar para os nós. Uma peculiaridade desse tipo de nó é que não possuem salvo nenhum estado do livro-razão. Outro ponto é que o serviço de ordenação pode ter mais de uma instância rodando, chamados de ordenadores, e juntos formam o serviço de ordenação.

2.3.3 FLUXO DE TRANSAÇÃO

Nessa seção serão apresentadas as etapas de execução de uma transação dentro da Hyperledger Fabric, que possui uma arquitetura de validação de ordem de execução de *blockchain*. Em resumo, uma transação no Hyperledger Fabric é executada em etapas como demonstrado na Figura 1. Importantes passos são o de execução da transação, a ordenação e empacotamento dentro de um bloco, a validação, encerrando com a transação sendo persistida no livro-razão (ANDROULAKI *et al.*, 2018, p. 5).

Figura 1 – Fluxo de transações no Hyperledger Fabric



Fonte: Androulaki, *et al.* (2018).

2.3.3.1 ETAPA DE EXECUÇÃO

Nessa fase um cliente envia uma solicitação para rede. Lembrando que essa solicitação carrega consigo o nome da organização que está requerendo a transação, os dados da transação, o *chaincode* e método que vão ser invocados na rede. Assim que as informações chegam, cada nó executa o *chaincode* de forma isolada sem qualquer compartilhamento de informações. Nessa execução pode-se invocar outros *chaincodes*, ou seja, uma invocação pode chamar outras, como uma cascata de execuções. Até esse momento nada das informações alteradas é salvo no livro-razão. Apenas os nós criaram uma estrutura de dados contendo o conteúdo a ser alterado que é chamado de *rw-set* (ANDROULAKI *et al.*, 2018, p. 6).

Para finalizar a fase de execução, o cliente espera que todos os nós retornem a estrutura de dados contendo o conteúdo a ser alterado de modo que respeite a política de endosso configurado para o *chaincode* e verifica se todos os nós geraram a mesma estrutura para assim poder enviar a transação para o serviço de ordenação (ANDROULAKI *et al.*, 2018, p. 6).

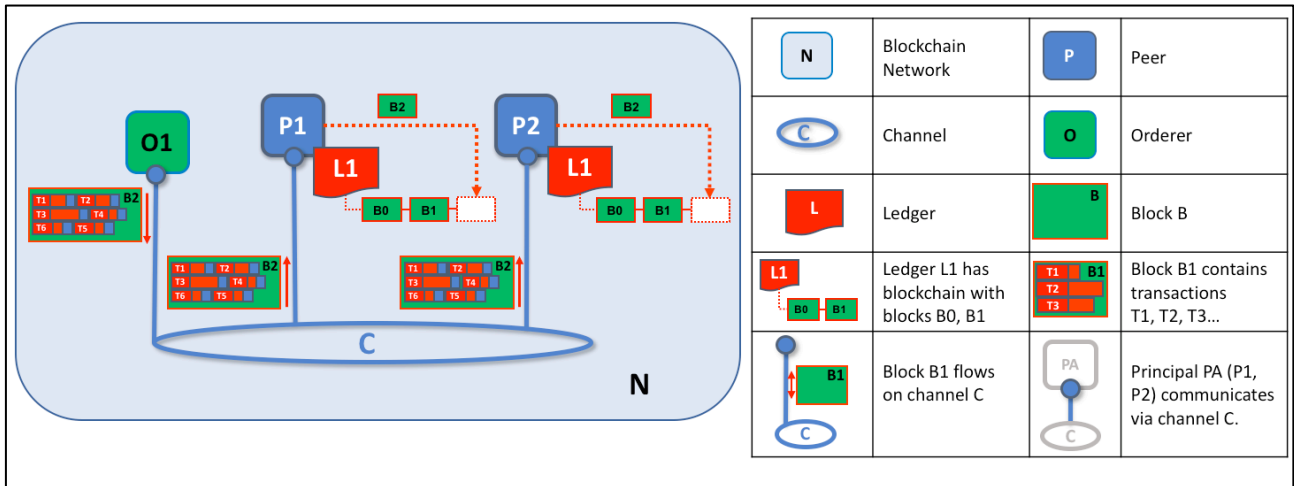
2.3.3.2 ETAPA DE ORDENAÇÃO

De acordo com Hyperledger (2023), assim que a fase de execução termina, o serviço de ordenação recebe a estrutura de dados contendo o conteúdo a ser alterado, o *rw-set*. Esse recebimento acontece de forma concorrente, ou seja, recebe várias transações praticamente ao mesmo tempo. Com isso ele usa algoritmos de ordenação e técnicas de agrupamento para poder ordenar os blocos de forma otimizada e com maior taxa de transferência. Vale destacar que a ordem que as transações são agrupadas num bloco não é necessariamente a mesma que o serviço de ordenação recebeu.

Para finalizar a etapa de ordenação, o serviço empacota as transações com uma ordem definitiva. Este pacote é chamado de bloco, que será enviado a todos os nós da rede. Um ponto importante é que nesta fase a ordem das transações se torna imutável (HYPERLEDGER, 2023).

Como demonstrado na Figura 2, o ordenador recebeu as transações T1, T2, T3 ..., que definiu sua ordem e montou o bloco B2, que por fim enviou para todos os nós da rede para fazer a etapa de validação. Após esta etapa será adicionado no livro-razão.

Figura 2 – Fluxo do serviço de ordenação



Fonte: Hyperledger (2023).

2.3.3.3 ETAPA DE VALIDAÇÃO

Segundo Androulaki *et al.* (2018, p. 8) na fase de validação, o bloco com as transações ordenadas pelo serviço de ordenação é recebido e passa pela subetapas de verificação da política de endosso e de verificação da estrutura de dados. A primeira subetapa é a de verificação da política de endosso, que já ocorreu na etapa de execução, porém ocorre novamente para certificar-se que não houve alteração no meio do fluxo.

A próxima subetapa é a verificação da estrutura de dados contendo o conteúdo a ser alterado. Nesta subetapa é verificado se todas as alterações ainda fazem sentido. Se tudo está correto a transação é marcada como válida. Caso haja alguma inconsistência, a transação é marcada como inválida. E por fim, os dados são efetivamente inseridos no livro-razão do nó que está executando, o qual futuramente irá se sincronizar com os demais nós da rede (ANDROULAKI *et al.*, 2018, p. 8).

Uma peculiaridade que aponta Androulaki *et al.* (2018, p. 8) é que mesmo as transações consideradas inválidas ficam armazenadas na *blockchain*, mas não fazem parte do estado de um objeto. Isso ocorre porque o pedido para fazer a inserção no livro-razão é independente do estado do *chaincode*.

2.4 TRABALHOS CORRELATOS

Nessa seção são apresentados trabalhos com características semelhantes aos principais objetivos do estudo proposto. O primeiro é um estudo da viabilidade da utilização de *blockchain* em cartórios (MENEZES, 2020) (Quadro 1). O segundo propõe um sistema baseado em contratos inteligentes em plataforma *blockchain* para a concessão de permissão a dados de saúde (JUNQUEIRA, 2020) (Quadro 2). O terceiro apresenta uma abordagem baseada em *blockchain* para armazenar e controlar o acesso aos certificados de alunos do ensino superior (ABREU, 2020) (Quadro 3).

Quadro 1 – Trabalho Correlato 1

Referência	MENEZES, Leonardo Dias. Blockchain e cartórios: uma solução viável?. 2020. 61 folhas. Dissertação (Mestrado em Ciências) - Universidade de São Paulo, São Paulo.
Objetivos	Verificar a viabilidade da aplicação de uma solução de <i>blockchain</i> dentro dos serviços de Cartórios de Notas Brasileiros sem a necessidade de alteração da legislação atual.
Principais funcionalidades	O usuário poderá realizar o upload de documento na plataforma, fazendo sua assinatura com seu certificado digital emitido e reconhecido pela Autoridade Certificadora Central e por fim o usuário pode incluir novas pessoas para fazer o processo de assinatura.
Ferramentas de desenvolvimento	Utilizou a rede de <i>blockchain</i> da <i>Ethereum</i> , escrevendo os contratos inteligentes em Solidity Foi utilizado para desenvolvimento o <i>Truffle</i> e o <i>Ganache</i> que são ferramentas para rodar a rede localmente e fazer testes. Para a interface Web foi utilizado o <i>JavaScript</i> e <i>DApp</i> .
Resultados e conclusões	O estudo realizado aprovou a viabilidade da aplicação respeitando os princípios Histórico, Disciplinares e Legais. A solução se mostra capaz, porém deve haver uma demanda da população e o reconhecimento dos órgãos públicos assim como levar em conta os custos envolvidos para incluir as transações dentro da rede <i>Ethereum</i> que possui uma taxa de transação e execução.

Fonte: elaborado pelo autor.

Quadro 2 – Trabalho Correlato 2

Referência	JUNQUEIRA, Natália Rodrigues. Concessão de permissão a dados de saúde baseada em contratos inteligentes em plataforma de blockchain. 2020. 90 folhas. Dissertação (Mestrado em Ciência da Computação) - Universidade Federal de Goiás, Goiânia.
Objetivos	Explorar o estado da arte na tecnologia <i>blockchain</i> para os dados de saúde usando contratos inteligentes. Desenvolver uma aplicação onde o paciente possa controlar a permissão dos seus dados pessoais de saúde. Analisar as plataformas Ethereum e Hyperledger Fabric como possíveis ferramentas para se desenvolver.
Principais funcionalidades	O cadastramento do paciente dentro da rede <i>blockchain</i> . O contrato inteligente para a concessão de permissão aos dados pessoais de saúde. O histórico de compartilhamento dos dados.
Ferramentas de desenvolvimento	A ferramenta escolhida foi o Hyperledger Fabric para a implementação dos contratos inteligentes.
Resultados e conclusões	A ferramenta escolhida foi o Hyperledger Fabric pois ele permite a configuração de permissão de cada organização, sendo assim não sendo uma <i>blockchain</i> pública, com dados expostos para todos. O projeto conseguiu alcançar os objetivos colocando como ressalva a criptografia dos dados e o baixo desempenho da rede por estar sendo executada em uma máquina virtual.

Fonte: elaborado pelo autor.

Quadro 3 – Trabalho Correlato 2

Referência	ABREU, Antônio Wellington dos Santos. Uma abordagem baseada em blockchain para armazenamento e controle de acesso aos dados de certificados de alunos do ensino superior. 2020. 146 folhas. Dissertação (mestrado) - Universidade Federal do Ceará, Quixadá.
Objetivos	Desenvolver uma aplicação onde seja possível publicar e validar diplomas. Aplicar um questionário para profissionais da área. Analisar o desempenho da aplicação construída.
Principais funcionalidades	Consultar a veracidade dos diplomas. Poder cadastrar novos diplomas e revogá-los. Poder fazer o cadastro no IES manualmente e automaticamente.
Ferramentas de desenvolvimento	Interface <i>frontend</i> chamado Educ-Dapp para a interação dos usuários, construído com JavaScript, HTML e CSS. A camada de <i>blockchain</i> , responsável por gravar os dados.
Resultados e conclusões	O sistema pode fornecer uma alternativa viável para armazenar e controlar os diplomas, pois os dados ficam armazenados de forma segura, aumentando a confiança e transparência na veracidade das informações.

Fonte: elaborado pelo autor.

Como apresentado nos quadros anteriores, os trabalhos correlatos resolvem diferentes problemas. Menezes (2020) propõe o gerenciamento de autenticação, escritura pública e certidões que vem para auxiliar o sistema burocrático envolvendo os cartórios. Junqueira (2020) traz a proposta de deixar os dados pessoais de saúde mais seguros e no controle do paciente para que ele possa decidir com quem compartilhar suas informações sobre saúde. Por fim, Abreu (2020) sugere um sistema para o controle de diplomas nas instituições de ensino superior já que o controle é feito por cada instituição à parte e o fluxo de emissão e validação desses diplomas é um processo manual e que leva tempo para ser efetuado sem contar as possíveis falsificações no processo.

Na construção da arquitetura Menezes (2020) e Abreu (2020) usaram a rede Ethereum como *blockchain*, que tem como característica ser pública e ter custo para inserir uma transação. O fato de ser pública significa que qualquer pessoa pode se conectar na rede e virar um nó, assim ajudando a verificar as transações, porém para as transações serem processadas demandam um custo envolvido que quem estiver adicionando terá que pagar, já que os participantes da rede cada vez que processam um bloco, ficam com uma recompensa por ajudar a rede. E a linguagem utilizada para a construção dos contratos inteligentes foi a Solidity, que é a linguagem que a rede Ethereum suporta. Todavia na arquitetura de Junqueira (2020) não há custo de transação envolvido, pois ela utiliza a rede Hyperledger Fabric que possui a característica de ser privada, ou seja, apenas entidades com a devida autenticação válida podem ser um nó da rede, que por sua vez nessa rede não há custos monetários para fazer transações, apenas custos computacionais. E a linguagem utilizada para a construção dos contratos inteligentes foi Golang, já que a rede Hyperledger Fabric suporta Golang, Javascript e Java.

Para a interação do usuário com os sistemas, Menezes (2020) e Abreu (2020) utilizaram de uma aplicação web para fazer as interações e Junqueira (2020) utilizou de uma aplicação móvel. Nesse quesito indiferente qual solução escolhida o importante é o usuário conseguir interagir com o sistema de forma fácil.

A partir da comparação das características, é evidente que cada rede de *blockchain* possui suas vantagens e que todas apresentam características marcantes como a descentralização, confiança e verificação dos dados sem a necessidade de um agente terceiro. Diante desse contexto, o protótipo deste trabalho foi implementado usando a rede Hyperledger Fabric, uma rede permissionada, para o controle do ciclo de vida de carros, de tal modo que apenas nós que são permitidos participam da rede, como montadoras, concessionárias, mecânicas e as outras entidades envolvidas.

3 DESCRIÇÃO DO PROTÓTIPO

Nesta seção é apresentada a arquitetura proposta que faz uso de tecnologias como *blockchain* e contratos inteligentes. O objetivo é desenvolver um protótipo de rede *blockchain* privada utilizando Hyperledger Fabric juntamente com os contratos inteligentes para gerenciar o ciclo de vida de carros, desde sua fabricação, licenciamento, transferências, manutenções e financiamentos para assim avaliar a viabilidade do protótipo para funcionamento no mundo real.

3.1 ESPECIFICAÇÃO

O ciclo de vida de automóveis contém múltipla etapas, diversas variações e especificidades de cada tipo de automóvel (carro, moto, caminhão etc.), cada um possuindo diferentes legislações. Para o protótipo deste trabalho focou-se apenas na entidade carro a fim de explorar este domínio de aplicação e não aumentar o escopo para todas as especificidades.

Para conseguir validar o protótipo também foi fechado o escopo quanto a todos os tipos de situações e variações dentro do ciclo de um carro. Uma destas limitações foi excluir a propriedade por uma empresa, ou seja, pessoa jurídica adquirir um carro para fazer parte de sua frota, apenas deixando entidade pessoa física, pois existem variações burocráticas que não trariam vantagens para validar o protótipo. Outra consideração foi a quantidade de atividades que podem ser inseridas na vida de um carro, desde seguro, infrações de trânsito e tudo mais que possa envolver este ciclo. As atividades que foram implementadas estão descritas na forma de requisitos na seção 3.1.2.

3.1.1 PROBLEMA

Um carro quando é fabricado passa por uma série de burocracias em sua vida, desde sua transação comercial, licenciamento, transferências para outras pessoas, financiamentos atrelados e outras situações. Essas situações exigem tempo, esforço e custos para as pessoas e empresas, além da frustração dos usuários que ganhariam muito com uma forma simples e descomplicada para resolver essas burocracias.

Outro problema percebido é a falta de comunicação entre os sistemas, o que pode gerar dados inconclusivos e divergentes em que cada organização poderá ter um dado diferente para o mesmo carro. Tal situação pode causar financiamentos incorretos, licenciamentos indevidos e transferências não permitidas. Para isso um sistema onde tenha todas as informações disponíveis e de confiança é essencial.

3.1.2 REQUISITOS PRINCIPAIS

O protótipo deverá:

- a) armazenar o cadastro de um carro (Requisito Funcional - RF);
- b) permitir montadoras inserir novos carros (RF);
- c) permitir concessionárias vender um carro para uma pessoa física (RF);
- d) permitir mecânicas adicionarem manutenções efetuadas em um carro (RF);
- e) permitir transferências de carros entre pessoas físicas (RF);
- f) ser capaz de identificar transferências inválidas de carros entre pessoas (RF);
- g) disponibilizar uma API que se comunique com a *blockchain* (Requisito Não Funcional - RNF);
- h) disponibilizar uma interface web para interação (RNF);
- i) utilizar rede Hyperledger Fabric (RNF);
- j) utilizar contratos inteligentes para definir regras das transações (RNF).

3.1.3 ORGANIZAÇÕES E CASOS DE USO

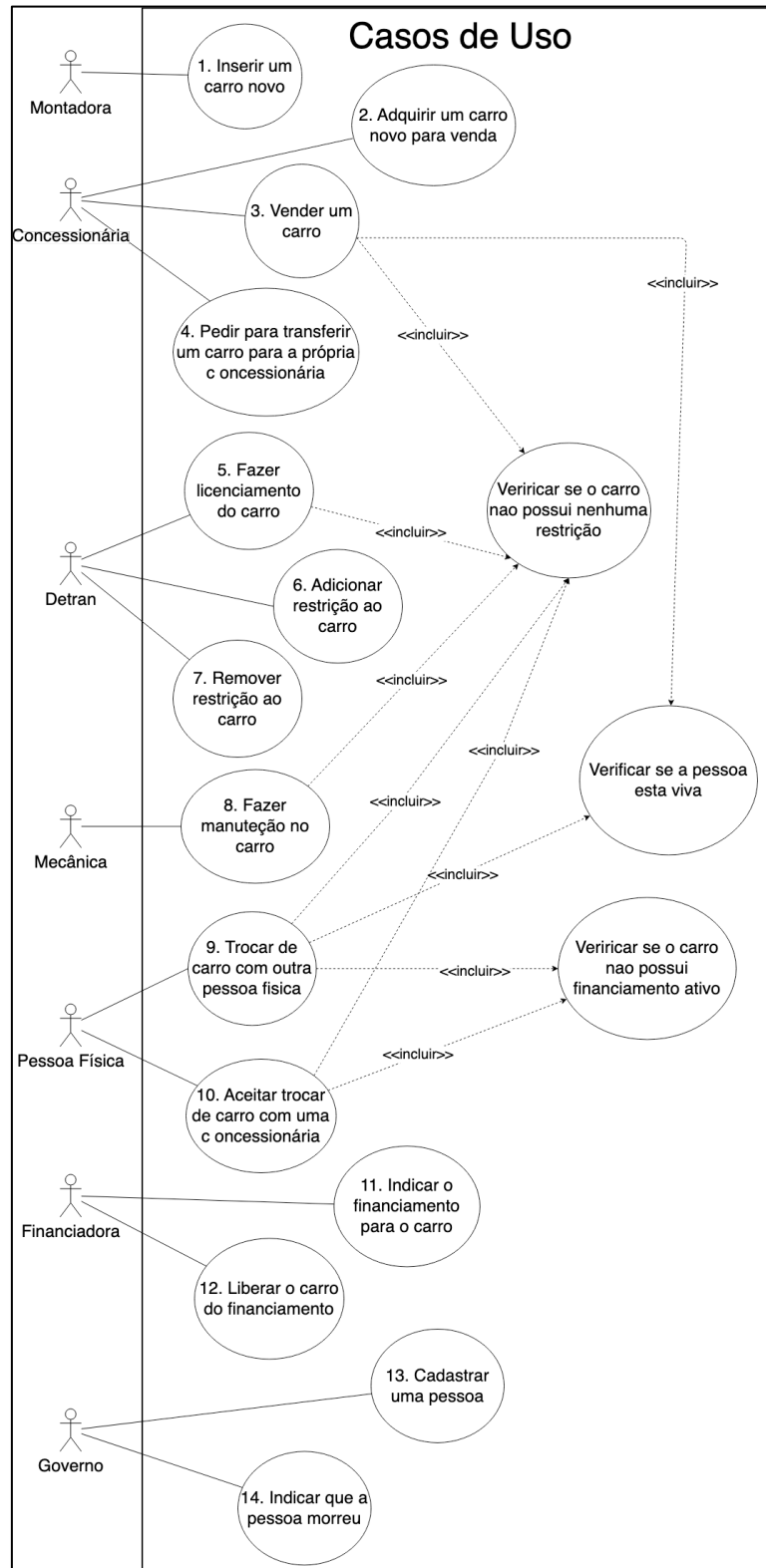
Para a construção do protótipo limitou-se às seguintes organizações, que são atores no modelo de casos de uso da Figura 3:

- a) O governo, responsável por ter o cadastro e atualização dos cidadãos (pessoa física);
- b) O Detran, responsável por fazer o licenciamento dos carros, adição e remoção de restrições do carro,

- indicar a troca de um carro de uma pessoa física para outra pessoa ou para uma concessionária;
- c) As montadoras, que podem inserir um novo carro fabricado;
- d) As concessionárias, que podem adquirir um carro de uma montadora e fazer a venda de carro;
- e) As mecânicas, que podem adicionar manutenções feitas no carro;
- f) As financiadoras, que podem indicar se o carro está alienado em razão de financiamento.

Mais detalhes dos casos de uso podem ser conhecidos no apêndice A.

Figura 3 – Casos de Uso



Fonte: elaborado pelo autor.

3.1.4 CICLO DE VIDA DO CARRO

O protótipo tem como foco construir um ciclo de vida do carro para entender se é possível através do sistema resolver tanto a burocracia envolvida na vida de um carro, quanto ter um sistema que forneça para as organizações envolvidas no processo uma fonte de dados confiável.

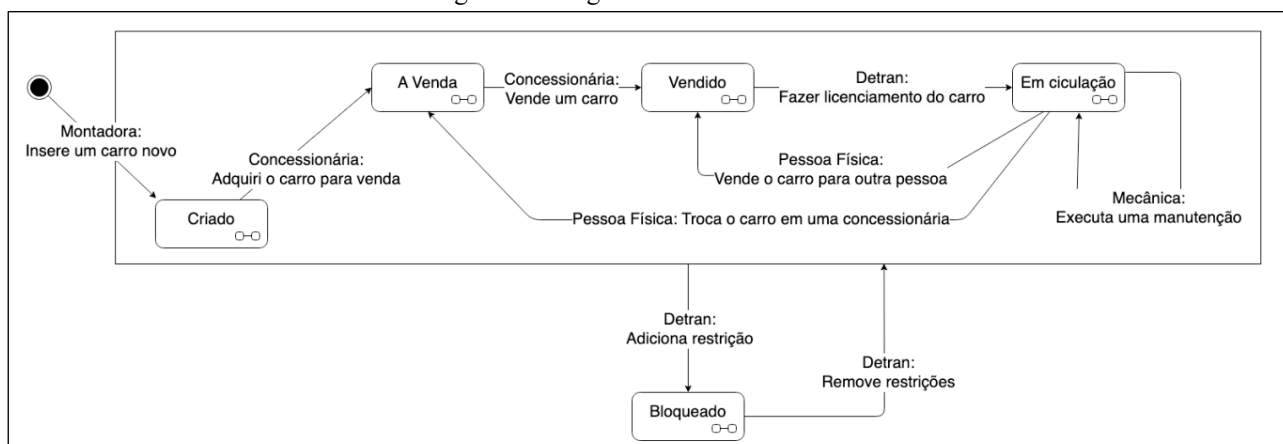
Quando se trata de ciclo de vida de um carro pode-se definir um estado em que o carro se encontra. Como mostrado na Figura 4, os estados seguem uma sequência:

- Criado: considerado quando a montadora fabrica um carro e insere os dados desse novo carro na *blockchain*;
- À Venda: considerado quando um carro é adquirido por uma montadora e fica à venda para as pessoas físicas;
- Vendido: considerado quando o carro é adquirido por uma pessoa física;
- Em circulação: considerado quando o carro efetuou o licenciamento;
- Bloqueado: quando há uma restrição sobre o carro.

Quando o carro se encontra em circulação a qualquer momento pode sofrer uma manutenção e essa ser registrada nos dados do carro e pode ser vendido para outra pessoa, fazendo voltar para o estado de vendido e seguindo o fluxo novamente.

Uma consideração importante é que a qualquer momento pode-se adicionar uma restrição, o que implica em o carro passar ao estado de bloqueado, ou seja, fica impossibilitado de fazer operações até que se remova a restrição, voltando ao seu estado anterior.

Figura 4 – Diagrama de estados do carro



Fonte: elaborado pelo autor.

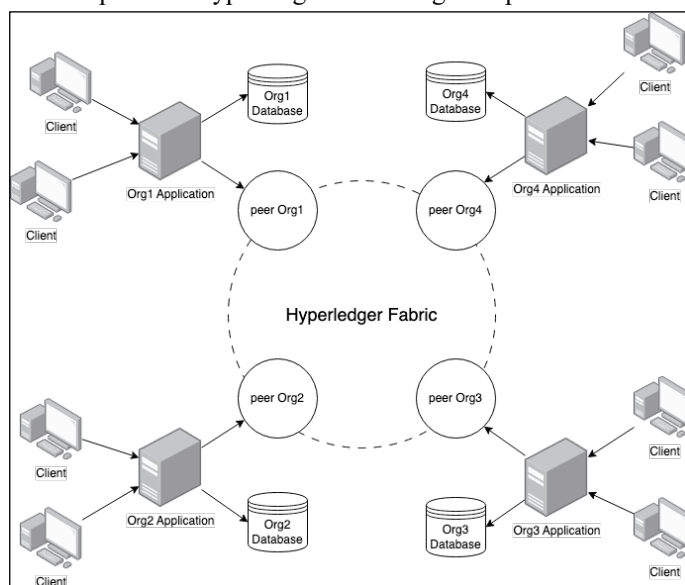
3.2 IMPLEMENTAÇÃO

Nessa seção será explicado a implementação do protótipo, descrevendo como está prevista a arquitetura em um sistema em produção. Também será apresentado a arquitetura construída, as configurações da rede Hyperledger Fabric, a implementação da API de comunicação com a *blockchain* e por fim detalhes da interface.

3.2.1 ARQUITETURA EM UM SISTEMA REAL

Uma das principais intenções de se usar a rede *blockchain* é a centralização dos dados, porém não há uma aplicação centralizada, ou seja, um ponto de falha que se ficar indisponível todos os sistemas também ficarão. Por isso cada organização tem seus nós na rede e cada nó uma cópia das informações. Como é observado na Figura 5, cada organização tem sua própria aplicação rodando na linguagem de programação que estiver, e cada organização tem suas credenciais para poder fazer parte da rede e se conectar nos seus nós. Assim cada organização mantém seu sistema rodando e se uma outra organização ficar indisponível não afetará em nada.

Figura 5 – Arquitetura Hyperledger Fabric sugerida para um sistema em produção



Fonte: elaborado pelo autor.

Com isso se tem que cada organização é totalmente independente uma da outra. Cada uma com sua aplicação e clientes conectados, sem sofrer em nada com a indisponibilidade de outra organização e até mesmo a possibilidade de adicionar e remover organizações a qualquer momento sem dificuldades.

Também se identifica que cada uma tem seu banco de dados com as informações salvas que fazem sentido para seu negócio, e caso não possuam alguma informação podem acionar a rede para buscá-la. Porém o ideal para informações mais acessadas é mantê-las dentro do seu próprio banco de dados.

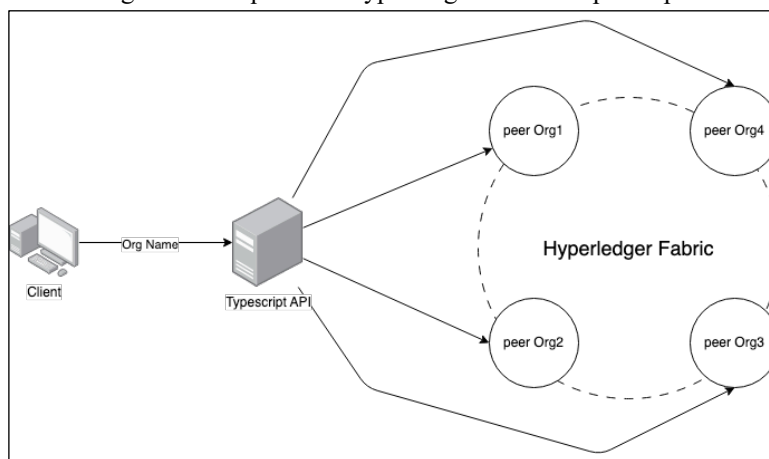
3.2.2 ARQUITETURA NO PROTÓTIPO

Como a intenção do protótipo é interagir o máximo com a rede *blockchain* e estressar o seu uso e habilidades, se deixou de lado a construção de sistemas separados para cada organização. Deste modo não se tem vários sistemas clientes diferentes para cada organização, mas um único protótipo que simula esta multiplicidade de organizações e tarefas.

A arquitetura construída se preocupou em construir no protótipo apenas um sistema de conexão, que será uma API em Typescript, com rede *blockchain* e apenas um cliente, implementado em React com Typescript para interagir e visualizar os dados. Isto trouxe uma economia de tempo de desenvolvimento, mas ainda permitindo estressar mais pontos da rede Hyperledger Fabric.

Como se observa na Figura 6, há um cliente que pode se conectar e selecionar qual organização quer simular, ou seja, o login para entrar é apenas a opção de organizações que há dentro da rede. Assim cada chamada para a API levará consigo o nome da organização que está simulando. Dentro da API será feita a autenticação com a rede, passando os certificados e será executada a transação, que será explicado como funciona na seção 3.2.6.

Figura 6 – Arquitetura Hyperledger Fabric no protótipo



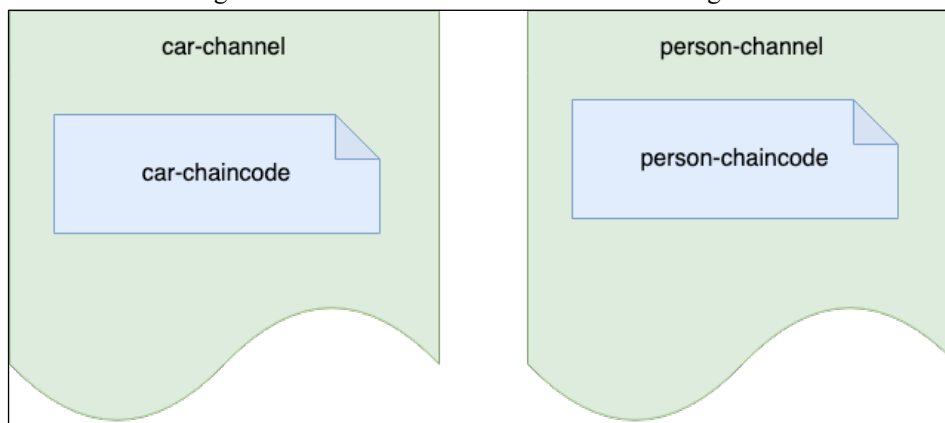
Fonte: elaborado pelo autor.

3.2.3 CONFIGURAÇÕES DO HYPERLEDGER FABRIC

A configuração da rede é parte importante para executar as transações e para isso se definiu as organizações que fazem parte da rede. As organizações estão listadas na seção 3.1.3 com seus respectivos casos de uso.

Na configuração de canais, explicado na seção 2.3.1, foram criados dois canais, como se observa na Figura 7. Um canal voltado para informações das pessoas (*person-channel*), onde dentro dele instalou-se o contrato de pessoas, responsável por cuidar das informações das pessoas. Foi criado como um canal separado pois pensou-se em uma rede que pode expandir para outras áreas da sociedade, por exemplo, área da saúde, a qual poderia receber acesso ao canal da pessoa e criar seu próprio canal de saúde. Assim dados do cadastro de pessoa estariam centralizados em um único canal.

Figura 7 – Estrutura dos canais e contratos inteligentes



Fonte: elaborado pelo autor.

O outro canal é o de carro (*car-channel*), onde se instalou o contrato de carros. É nele que toda a burocracia envolvendo o ciclo de vida ocorre. Importante detalhar que o contrato inteligente do carro acessa informações do canal de pessoa, por meio do contrato inteligente lá instalado. Isso garante, por exemplo, que quando um carro for vendido possa ser verificado se a pessoa que comprou realmente existe.

Outra configuração importante é as organizações e seus nós pares e os ordenadores. Como o ambiente é executado na máquina local usando Docker, foi preciso deixar um número não muito grande pois não seria possível rodar as instâncias em um computador pessoal. Para isso foi criado apenas um nó par para cada organização e apenas um nó que compõe o serviço de ordenação. Quanto às organizações, foram criadas duas organizações de montadoras, duas concessionárias, duas mecânicas e uma financiadora.

3.2.4 CONTRATOS INTELIGENTES CRIADOS

Como mencionado na seção 3.2.3, foram criados dois contratos inteligentes, um sendo o de pessoa e o outro de carro. Os contratos inteligentes foram implementados em Typescript, pois o Hyperledger possui bibliotecas oficiais para Typescript, Java e Go.

Para facilitar o desenvolvimento, criou-se um contrato base com métodos comuns, assim como decoradores para fazer a validação de organizações permitidas para cada método e outro para a decompor a resposta.

No Quadro 4, há alguns exemplos de métodos criados para ajudar na implementação dos contratos inteligentes. O método `GetState` (linha 79) recupera informações do livro-razão. O método `PutState` (linha 66) salva informações no livro-razão e o método `HasState` (linha 70) verifica se já há um registro para a chave no livro-razão. Como se observa, as informações do livro-razão são acessadas sempre pelo `ctx: Context`, que é onde se interage. Esse contexto é injetado automaticamente pela biblioteca oficial. Um detalhe importante é que para buscar e salvar informações no livro-razão usa-se sempre a chave, que será o identificador único daquele registro.

Quadro 4 – Métodos do contrato base

```

64 ✓ export class BaseContract extends Contract {
65
66     protected async PutState(ctx: Context, key: string, value: object): Promise<void> {
67         await ctx.stub.putState(key, Buffer.from(stringify(sortKeysRecursive(value))));
68     }
69
70 ✓     protected async HasState(ctx: Context, key: string): Promise<boolean> {
71         const stateJSON = await ctx.stub.getState(key)
72         if (!stateJSON || stateJSON.length === 0) {
73             return false;
74         }
75
76         return true;
77     }
78
79 ✓     protected async GetState(ctx: Context, key: string): Promise<any> {
80         const stateJSON = await ctx.stub.getState(key)
81         if (!stateJSON || stateJSON.length === 0) {
82             throw new Error(`The object with key=${key} does not exist`);
83         }
84
85         return JSON.parse(stateJSON.toString());
86     }
87

```

Fonte: elaborado pelo autor.

Como se observa no Quadro 5, uma das características citadas é que de dentro de um contrato inteligente pode-se executar métodos de outros canais e contratos inteligentes. Esse método `getPerson` (linha 9) está no contrato inteligente de carros e é usado para recuperar as informações de uma pessoa. Para isso usou-se o `invokeChaincode` (linha 10) passando como parâmetros o nome do *chaincode*, o método a ser executado com os parâmetros necessários e por fim o canal em que está. Uma observação importante é que a organização que estiver executando precisa ter acesso também ao canal que esse outro contrato inteligente se encontra.

Quadro 5 – Exemplo de buscar informações de outros contratos inteligentes

```

9 ✓     private async getPerson(ctx: Context, cpf: string): Promise<any> {
10         const result = await ctx.stub.invokeChaincode(
11             'person',
12             [
13                 "PersonContract:ReadPersonAlive",
14                 cpf
15             ],
16             "person-channel"
17         )
18
19         if (result.status !== 200) {
20             throw new Error(result.message);
21         }
22
23         return JSON.parse(result.payload.toString());
24     }
25

```

Fonte: elaborado pelo autor.

No Quadro 6, primeiro pode-se notar, nas linhas 56 e 78, o uso dos métodos bases criados no contrato base para facilitar o desenvolvimento da lógica e ver um exemplo de como os decoradores foram empregados. O primeiro `@Transaction()` (linha 46), vem da biblioteca oficial do Hyperledger, que serve para indicar que se trata de uma transação. O segundo, `@BuildReturn()` (linha 47), foi um dos decoradores implementados para receber o retorno da função e transformar a resposta de JavaScript Object Notation (JSON) para *string*. O terceiro, `@AllowedOrgs(["govMSP"])` (linha 48), serve para ver se a organização que está executando a transação está na lista de autorizados.

Outro detalhe é que ao usar o `this.PutState(ctx, person.cpf, person)` (linha 78), foi utilizado o CPF como chave por se tratar de um identificador único por pessoa.

Quadro 6 – Método de criar pessoa no contrato inteligente

```

46     @Transaction()
47     @BuildReturn()
48     @AllowedOrgs(["gov"])
49     public async CreatePerson(
50         ctx: Context,
51         cpf: string,
52         name: string,
53         birthday: string,
54         motherName: string
55     ): Promise<object> {
56         if (await this.HasState(ctx, cpf)) {
57             throw new Error(`The person ${cpf} already exists`);
58         }
59
60         const birthday_date = new Date(birthday)
61
62         if (!isCorrectDate(birthday_date)) {
63             throw new Error(`The birthday is a invalid date`);
64         }
65
66         if (birthday_date > new Date()) {
67             throw new Error(`The birthday is bigger than today`);
68         }
69
70         const person: Person = {
71             cpf: cpf,
72             name: name,
73             birthday: new Date(birthday),
74             motherName: motherName,
75             alive: true
76         };
77
78         await this.PutState(ctx, person.cpf, person)
79         return person;
80     }

```

Fonte: elaborado pelo autor.

3.2.5 COMO SÃO ARMAZENADOS OS DADOS NA BLOCKCHAIN

Para armazenar os dados na *blockchain* como mencionado na seção 3.2.4, usa-se um atributo identificador para salvar os dados, sendo que o conteúdo pode ser salvo como o desenvolvedor desejar. No caso do protótipo foi utilizado o padrão JSON. Assim para salvar ou resgatar as informações do carro foi criado um objeto, como o exemplo da Figura 8, em que há os campos com seus respectivos valores. Como foi apenas utilizado um contrato inteligente para gerenciar todos os atributos do carro, então todas as informações ficam salvas integralmente no objeto do carro.

Figura 8 – Estrutura do objeto de carro na *blockchain*

```

1     {
2         "chassisId": "774d7ed1-2a45-4eb1-bc04-d955fc67d651",
3         "brand": "montadoraC",
4         "model": "Model 1",
5         "color": "Blue",
6         "year": 2020,
7         "ownerCpf": "111.111.111",
8         "ownerDealershipName": null,
9         "maintenances": [],
10        "restrictions": [],
11        "financingBy": "financiadoraR"
12    }

```

Fonte: elaborado pelo autor.

3.2.6 API DE COMUNICAÇÃO COM A REDE BLOCKCHAIN

Para a comunicação do *frontend* ou qualquer outro cliente com a *blockchain* é preciso um serviço intermediário. Com isso se construiu uma API (Application Programming Interface) via HTTP onde são recebidas as informações e executada na *blockchain*.

Para fazer a execução na *blockchain* a partir da API, usou-se da própria biblioteca oficial do Hyperledger Fabric que traz facilidades para fazer a conexão. A conexão é feita a partir de 3 informações. A primeira é um *connectionProfile*, que tem todas as informações de uma organização, como o link dos nós, seu identificador, e dados de outros nós para se comunicarem como observado no Quadro 7. A segunda informação é um certificado para se autenticar na rede. Por fim a terceira informação é a chave privada do certificado. Um detalhe importante é que para cada organização são necessárias estas três informações.

Para ser o mais simples e direto possível apenas uma rota foi construída na API, a rota `/:typeTransaction/:channelName/:chaincodeName/:transactionName`, que recebe as informações no caminho da rota. A primeira informação diz respeito ao tipo de transação, que pode ser *submit* ou *evaluate*, isso é o tipo de transação que irá ser executado na *blockchain*. A de *submit* é quando irá mudar o estado da *blockchain*, ou seja, inserir algo novo, esse tipo de transação é o mais lento pois passa por todas as etapas. O segundo tipo, *evaluate*, é apenas para ler informações, serve para justamente ser mais rápido.

Outra informação passada pelo caminho da rota é o nome do canal que se quer executar, seguido do nome do contrato inteligente e do nome da transação, ou nome do método, que será executado. Por fim, no corpo da requisição deve-se enviar os dados que espera receber no método que será executado.

Quadro 7 – Exemplo de *connectionProfile* do Detran

```
name: fablo-test-network-detrans
description: Connection profile for detrans in Fablo network
version: 1.0.0
client:
  organization: detrans
organizations:
  detrans:
    mspid: detransMSP
    peers:
      - peer0.detrans.car-lifes-cycle.com
      - peer0.gov.car-lifes-cycle.com
      - peer0.montadora-c.car-lifes-cycle.com
      - peer0.montadora-d.car-lifes-cycle.com
      - peer0.concessionaria-f.car-lifes-cycle.com
      - peer0.concessionaria-g.car-lifes-cycle.com
      - peer0.mecanica-k.car-lifes-cycle.com
      - peer0.mecanica-l.car-lifes-cycle.com
      - peer0.financiadora-r.car-lifes-cycle.com
    certificateAuthorities:
      - ca.detrans.car-lifes-cycle.com
peers:
  peer0.detrans.car-lifes-cycle.com:
    url: grpc://localhost:7021
  peer0.gov.car-lifes-cycle.com:
    url: grpc://localhost:7041
```

Fonte: elaborado pelo autor.

Para exemplificar o uso, suponha que se quer executar o método *CreatePerson* do Quadro 6. Para isso o tipo de transação será *submit*, pois muda o estado do livro-razão visto que se está adicionando uma nova pessoa. O nome do canal é *person-channel*, o nome do contrato é *person* e por fim o método é *CreatePerson*. Como se observa na Figura 9 A, o método recebe o CPF, nome, data de nascimento e nome da mãe nessa ordem, então assim será passado no corpo da requisição.

Figura 9 – Exemplo de requisição na API

The screenshot displays an API client interface with two panels, A and B. Panel A shows the 'Body' tab of a PUT request to the endpoint `{{URL_3000}}/submit/person-channel/person/CreatePerson`. The body is a JSON array: `[["111.222.333-44", "Yuri Hartmann", "06/07/2001", "marilda"]]`. Panel B shows the 'Headers' tab for the same request, with 'X-API-Key' set to 'gov'.

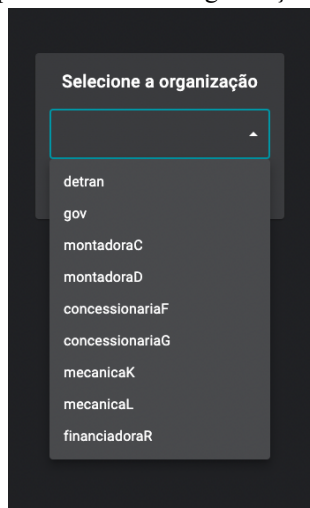
Fonte: elaborado pelo autor.

E para executar esse método, como foi explicado na seção 3.2.4, precisa ser uma organização autorizada, no caso apenas a organização `gov` é aceita para esse método. Passa-se no cabeçalho da requisição através da chave `X-API-Key`, como se observa na Figura 9 B, o nome da organização que será usada para executar. Com isso, ao final tem-se o retorno da pessoa criada, caso tudo ocorra com sucesso ou um erro com a mensagem do que aconteceu de errado.

3.2.7 FRONTEND

Para a construção do *frontend* foi utilizado a biblioteca React que ajuda na construção de interfaces web. O *frontend* foi construído de maneira que atendesse qualquer organização, sendo uma única interface igual para todos apenas para demonstração do ciclo de vida do carro. Para isso ao entrar na interface há a opção de se autenticar como uma das organizações da rede *blockchain* que foi criada como apresentado na Figura 10.

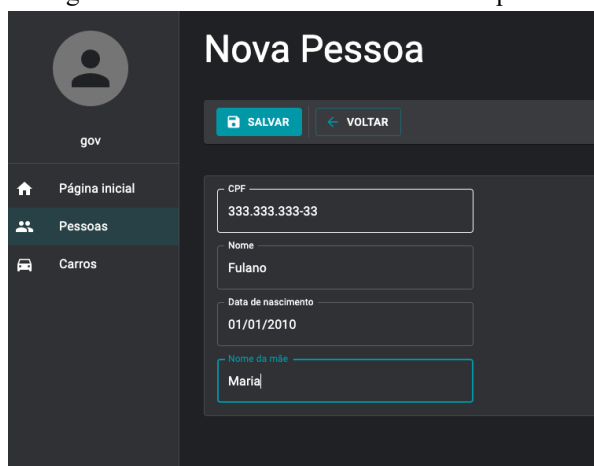
Figura 10 – Campo para selecionar a organização que deseja autenticar



Fonte: elaborado pelo autor.

Após a autenticação na interface chega-se no sistema principal em que se pode listar as pessoas e os carros, e em cada listagem temos ações que podem ser feitas. Pode-se ver as pessoas cadastradas e adicionar novas pessoas através do formulário próprio (Figura 11), lembrando que o formulário para inserir pessoa é exibido para qualquer organização autenticada. Porém ao salvar, caso não tenha permissão para fazer a ação, uma mensagem de erro será exibida – neste caso é preciso estar autenticado como `gov` para poder salvar uma nova pessoa.

Figura 11 – Formulário de adicionar nova pessoa



Fonte: elaborado pelo autor.

Todas as organizações têm acesso para ver os carros que estão cadastrados na rede, como visto na Figura 12. Para inserir um novo carro, apenas organizações com permissão irão conseguir inserir.

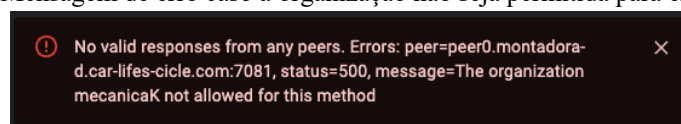
Figura 12 – Listagem de carros da rede

Ações	ID do chassi	Marca	Modelo	Cor	CPF do dono	Concessionaria	Ano	Financiado por
	028191f9-3a9...	montadoraC	model 1	blue	333 333 333-33	Já possui um dono!	2023	+
	02899732-9a9...	montadoraC	model 1	blue	-		2023	+
	0546828c-8d2...	montadoraC	model 1	blue	-		2023	+

Fonte: elaborado pelo autor.

Esse mesmo comportamento acontece com os menus, opções e formulários, pois sempre irão mostrar para todos, mas no momento de executar a transação na *blockchain* irá bloquear caso não seja de uma organização permitida. Como se observa na Figura 13, a *mecanicaK* tentou adicionar um carro, porém ela não tem permissão para executar esse método, então recebe a mensagem de erro.

Figura 13 – Mensagem de erro caso a organização não seja permitida para executar a ação



Fonte: elaborado pelo autor.

4 RESULTADOS

A construção do protótipo, desde seu objetivo até sua implementação, trouxe aprendizados sobre a rede Hyperledger Fabric. Um desses aprendizados foi a sua modularidade. A modularidade do Hyperledger Fabric oferece aos desenvolvedores a capacidade de personalizar o sistema de acordo com os requisitos específicos de seus casos de uso. Os diferentes módulos podem ser adaptados e configurados para atender às necessidades individuais, o que torna mais fácil a implementação de soluções personalizadas. Assim para a área automotiva que foi escolhida para o protótipo, foi possível implementar e desenhar a arquitetura e os dados para funcionar com a *blockchain* do Hyperledger Fabric.

A implementação da API de comunicação com a *blockchain*, descrita na seção 3.2.6, se mostrou muito eficaz para interagir com a *blockchain* através de uma comunicação mais simples e comum. A comunicação padrão da rede Hyperledger Fabric é apenas por gRPC, que detém uma autenticação complexa, mas ao se desenvolver uma API a autenticação foi simplificada a apenas um cabeçalho passado por parâmetro na requisição, o que facilitou muito o desenvolvimento do *frontend* e ajudou muito nos testes e execuções dos métodos dos contratos inteligentes.

A implementação do *frontend*, descrito na seção 3.2.7, trouxe uma facilidade para interagir com o sistema de forma simples e objetiva. Desta maneira se pode autenticar com qualquer organização da rede *blockchain* e ter acesso completo a todas as funcionalidades do sistema, podendo inclusive executar as transações não permitidas e ver o erro de forma a garantir que a autorização de execução está conforme o esperado.

Dessa maneira, a escolha de utilizar o Hyperledger Fabric garantiu que se pudesse construir de forma objetiva e sem maiores obstáculos o protótipo descrito, utilizando seu potencial para resolver os problemas dispostos. Assim como a construção da API e do *frontend* ajudou a montar um cenário de utilização para ver e alterar os dados com objetivo de observar o ciclo da vida de um carro.

4.1 COMPARATIVO DOS TRABALHOS CORRELATOS

Na apresentação do Quadro 8 é possível observar a comparação dos trabalhos correlatos. De uma forma ampla todos os correlatos utilizam o conceito de *blockchain* e contratos inteligentes, porém aplicados de formas diferentes e para resolver problemas distintos.

Quadro 8 - Comparativo dos trabalhos correlatos

Trabalhos Correlatos Características	Menezes (2020)	Junqueira (2020)	Abreu (2020)	Protótipo
Entidades que gerencia	Documentos de cartórios	Dados pessoais de saúde	Diplomas do ensino superior	Ciclo de vida do carro
Rede de <i>blockchain</i>	<i>Ethereum</i>	Hyperledger Fabric	<i>Ethereum</i>	Hyperledger Fabric
Tem custo para fazer transações	Sim	Não	Sim	Não
Modo de participação dos pares	Público	Privado	Público	Privado
Interface	Aplicação Web	Aplicativo Móvel	Aplicação Web	Aplicação Web
Linguagem de programação do contrato inteligente	Solidity	Golang	Solidity	Typescript

Fonte: elaborado pelo autor.

Como se observa no Quadro 8, a entidade que cada projeto gerencia é diferente e mostra que o uso de *blockchain* para resolver problemas da sociedade pode ser bem amplo e atinge várias áreas. A rede *blockchain* utilizada em cada projeto ficou dependente totalmente do problema a ser resolvido. Quando se fala de algo que pode ser público e de acesso a todos, ou seja, qualquer um pode interagir com a rede, usa-se uma rede *blockchain* pública, e quando os dados e as interações precisam ser autenticados e limitados a uma lista de organizações, utiliza-se rede *blockchain* privada como o Hyperledger Fabric.

Quando comparado o resultado obtido pelo protótipo em comparação com o de Menezes (2020), observa-se que a solução dele se mostrou viável, porém dependia da demanda da população e do reconhecimento dos órgãos públicos. Quando se pensa no protótipo construído também se tem essa dependência, porém com uma grande diferença, a rede *blockchain* utilizada. O trabalho de Menezes faz uso da rede Ethereum, que é uma rede *blockchain* pública que todos têm acesso, o que deixaria os órgãos públicos sem uma autoridade sobre as transações e os contratos inteligentes. Já no protótipo usa-se a rede Hyperledger Fabric que além dos órgãos públicos terem acesso, eles também seriam os responsáveis por incluir mais organizações na rede, fazendo o papel de regulador das organizações, além de se poder ter métodos dentro dos contratos inteligentes que apenas eles teriam autoridade para executar. Logo a solução se mostra mais aplicável e capaz de ser aceita e implementada pelos órgãos públicos.

No trabalho correlato de Junqueira (2020), se observa que um de seus objetivos foi escolher entre a rede *blockchain* Ethereum e Hyperledger Fabric. Ele acaba selecionando o Hyperledger Fabric pois possui a característica de autenticação. Essa característica foi muito explorada no protótipo pois além de apenas organizações que possuem acesso à rede poderem participar da rede, também criaram os contratos inteligentes. A cada método do contrato inteligente pode-se definir a lista de organizações aceitas e assim dar segurança que certas ações só poderiam ser executadas por determinadas organizações, o que mostrou que a escolha de usar o Hyperledger Fabric em vez de uma rede pública como a da Ethereum se mostrou assertiva.

Quando se vê da perspectiva do trabalho de Abreu (2020), sua solução se mostra muito boa e viável, pois trata de verificar os diplomas de uma forma centralizada, porém com os benefícios da *blockchain*. Em nenhum momento precisa da intervenção dos órgãos públicos, funcionando muito bem para o uso de universidades. Essa mesma característica de funcionar sem a interação dos órgãos públicos não se aplicaria ao presente protótipo, pois se trata de cuidar do ciclo de vida de um carro e esse tem forte vínculo ao estado, ou seja, tem burocracias a serem seguidas e monitoradas pelos órgãos públicos. Assim, o trabalho de Abreu conseguiu extrair muito bem o que uma *blockchain* pública tem a oferecer e mostrou que cada problema leva a uma escolha de uma rede *blockchain* privada ou pública.

Com a escolha da *blockchain*, o custo de transação vem como consequência da escolha. Numa rede pública, os nós que processam ganham uma recompensa por ajudar a rede. Enquanto numa rede privada, como cada organização precisa necessariamente ter ao menos um nó rodando, a rede sempre terá participantes para executar as transações. Lembrando que os custo aqui citado é em relação a adicionar uma transação na rede e não aos custos computacionais.

A linguagem de programação também está diretamente ligada a rede *blockchain* escolhida, já que depende da rede ter o suporte para as linguagens. A interface por sua vez pode ser livre e vai muito de como a interação será mais bem explorada, mas tanto aplicativos móveis como aplicações Webs tem o objetivo de apresentar os dados e fazer as interações com a rede de forma simples.

4.2 TESTE DE VELOCIDADE

Para se conseguir avaliar se a rede *blockchain* do Hyperledger Fabric consegue atender uma boa velocidade e capacidade de lidar com requisições, se conduziu testes de performance para conseguir metrificar sua capacidade.

4.2.1 CENÁRIOS DOS TESTES

Os cenários escolhidos para fazer esses testes levaram em conta o número de entidades, que no caso do protótipo é o número de carros registrados na *blockchain*. O segundo fator foi o número de requisições simultâneas e por fim se selecionou algumas transações do contrato inteligente com características diferentes.

Para a quantidade de carros dentro da *blockchain* se escolheu os seguintes números, 1.000, 10.000, 100.000 e 1.000.000, ou seja, de mil a um milhão multiplicando por 10. Assim com essa diferença conseguiu-se tirar informações de se o volume de carros dentro da *blockchain* afeta a velocidade das transações.

Quanto ao segundo fator a ser levado em conta nos testes foi de requisições simultâneas e para isso se selecionou as quantidades de 1 e 30 requisições simultâneas que trazem informações relevantes quanto a capacidade de lidar com várias requisições concorrentes. Lembrando que as 30 requisições simultâneas tratam de 30 diferentes chamadas para a rede Hyperledger Fabric e não um método que cria 30 carros em uma única transação.

Por fim, o último fator foi a seleção de quais transações foram executadas e para isso se selecionou a transação de adicionar um carro à rede *blockchain*, que executa a parte de validação de dados e inserção de um novo registro na rede, de acordo com o caso de uso “1. Inserir um carro novo”. A outra transação foi a de ler os dados de um carro da *blockchain*, que executa a parte de leitura de um dado na rede sem alteração de qualquer dado. A última transação foi a de adicionar uma manutenção em um veículo, que executa a leitura de um dado, seu processamento e por fim a atualização do dado na rede, de acordo com o caso de uso “8. Fazer manutenção no carro”.

O ambiente para executar os testes foi um servidor provisionado na nuvem, com 16 núcleos de processamento, 32 GB de memória RAM e 30 GB de SDD. A rede *blockchain* foi executada em ambiente de virtualização usando-se a plataforma Docker, que é a técnica de virtualização em containers. Todas as organizações foram executadas com 1 nó e a rede com apenas 1 nó ordenador.

O tempo total da transação calculou-se levando em conta todas as etapas do fluxo de transação, seção 2.3.3, pois esse é o tempo que leva para se ter o dado com a garantia que foi processado e está registrado no livro razão.

4.2.2 RESULTADOS DOS TESTES

Após a execução dos testes, foram obtidos os seguintes resultados conforme a Tabela 1 e a Tabela 2:

Tabela 1 – Resultados de velocidade para adicionar um carro

Número de carros na rede	(a) Executando 1 requisição (milissegundos)	(b) Executando 30 requisições simultâneas (milissegundos)	Aumento de tempo entre volume de requisições simultâneas (b) / (a)
1.000	2180	3387	+ 55,3%
10.000	2182	3419	+ 56,6%
100.000	2179	3418	+ 56,8%
1.000.000	2179	3377	+ 54,9%

Fonte: elaborado pelo autor.

Tabela 2 – Resultados de velocidade para ler os dados de um carro

Número de carros na rede	(a) Executando 1 requisição (milissegundos)	(b) Executando 30 requisições simultâneas (milissegundos)	Diferença entre volume de requisições simultâneas (b) / (a)
1.000	2163	2619	+ 21%
10.000	2165	2589	+ 19,5%
100.000	2156	2632	+ 22%
1.000.000	2186	2553	+ 16,7%

Fonte: elaborado pelo autor.

Como se observa na Tabela 1, o tempo para adicionar um carro na *blockchain* em ambos os cenários de requisições simultâneas e em todos os cenários de números de carros na rede, foi praticamente igual. Apenas quando comparada a execução de 1 versus 30 requisições simultâneas, se observa um aumento expressivo de mais de 50%. Logo nota-se que transações simultâneas afetam sim a velocidade que as transações são executadas.

Quando se analisa a Tabela 2, o tempo para ler os dados de um carro em todos os cenários de quantidade de carros na rede e nos dois cenários de requisições simultâneas se manteve muito consistente e quando se compara a execução de 1 versus 30 requisições simultâneas teve um aumento entre 16% e 22%. Assim nota-se um leve aumento para execuções simultâneas.

Ao se comparar o tempo de adicionar um carro e ler os dados de um carro com 1 requisição simultânea, em ambos os casos e em todos os cenários de quantidade de carros na rede, o tempo foi muito próximo. Logo o tempo de registrar um dado ou ler um dado da rede, pode-se considerar que é linear. Porém quando se compara os dois métodos executados com 30 requisições simultâneas, se nota que o acréscimo para adicionar um carro fica em torno de mais 50% de tempo enquanto para ler um carro fica por volta de 20%, ou seja, em um alto volume de requisições, a velocidade fica mais afetada para adicionar dados do que ler dados da rede.

Um detalhe que se notou é que independentemente do método executado e da quantidade de carros na rede, o tempo foi pouco afetado. Com isso pode-se prever que se a rede tivesse 10.000.000 ou até mais carros o tempo continuaria praticamente o mesmo, o que mostra uma robustez da rede para grande quantidade de dados.

4.2.3 LIMITAÇÃO ENCONTRADA

Como descrito na seção 4.2.1, um dos métodos que se selecionou foi o de adicionar uma manutenção em um veículo, porém esse não foi apresentado nos resultados pois encontrou-se uma limitação significativa na rede. Quando executados os testes para esse método, ao fazer apenas uma requisição simultânea não foram encontrados problemas para adicionar a manutenção. Porém ao executar 2 ou mais requisições simultâneas, um erro acontecia ao executar o contrato inteligente. Este erro se referia à tentativa de atualização de um mesmo registro por várias transações ao mesmo tempo, ou seja, ao tentar atualizar um carro na rede com sua nova manutenção, se acontecasse 2 ou mais atualizações simultâneas a rede retornava erro e não conseguia salvar o carro com a nova manutenção.

Ao encontrar essa limitação da rede, a hipótese é que a rede não é adequada para altas taxas de execução de atualizações de registro, e sim adequada para leitura e inserção de novos registros. Nesse cenário de adicionar uma manutenção no carro, para que não acontecasse esse erro o caminho seria criar outro contrato inteligente que apenas cuidasse das manutenções e assim cada nova manutenção em um carro seria inserido um novo registro na rede e não a atualização de um registro já existente.

5 CONCLUSÕES

O objetivo principal do protótipo era construir uma rede *blockchain* onde se pudesse salvar os dados do carro e controlar o seu ciclo de vida. Mais especificamente, de ter uma forma de rastrear todos os dados de um carro de forma organizada e descentralizada. Este objetivo foi alcançado com sucesso, pois como foi mostrado conseguiu-se construir os contratos inteligentes e fazer uma interface para visualizar esse ciclo de informações. Ao final sempre se tem o histórico de tudo que aconteceu com o carro e a garantia que as informações só serão alteradas por organizações que tenham a permissão para executar.

Um dos objetivos específicos era identificar e modelar o ciclo de vida de carros, o que foi alcançado parcialmente. As ressalvas dizem respeito a que muitos dos processos burocráticos que existem atualmente estão ligados ao fato que as informações não estão de forma centralizada e que demandam de processos específicos para cada situação e região do país. Então o modelo do ciclo de vida de carros foi desenhado a partir do que existe atualmente, porém com modificações para se adequar ao protótipo.

Outros dois objetivos que foram propostos eram, criar uma rede *blockchain* privada usando Hyperledger Fabric e construir contratos inteligentes para gerir as regras de negócio envolvendo o carro. Ambos criados e implementados com sucesso. A rede *blockchain* foi construída com as devidas configurações de canais, organizações e contratos inteligentes e conseguiu-se interagir e usufruir da rede *blockchain*. Os contratos inteligentes foram construídos com sucesso e conseguiram atender as regras de negócios que foram implementadas, permitindo validar e processar os dados de forma correta.

O último objetivo específico era avaliar a viabilidade do protótipo para funcionamento no mundo real. Diante do exposto, levando em conta os trabalhos correlatos e os resultados apresentados, conclui-se que se mostra uma solução viável para funcionar no mundo real, atendendo a demanda de unificar os sistemas, para se ter as informações centralizadas e com garantia de estar corretas e ao mesmo tempo uma rede distribuída sem ponto de falhas.

Por fim, os próximos passos para evoluir o protótipo seriam investigar mais a fundo a segregação de canais por entidade, ou seja, avaliar a separação das manutenções, restrições e outros atributos em um canal separado do canal de carros e contratos inteligentes separados para não haver problemas de concorrência como encontrado nos testes. Outro ponto não mapeado pelo protótipo seria a possibilidade de enviar imagens para a rede, pois muitos processos dependem de imagens para se ter o histórico e documentação. Um último ponto seria abranger um espectro maior de veículos e tipos de ciclos que um veículo pode sofrer.

REFERÊNCIAS

- ABREU, Antônio Welligton dos Santos. **Uma abordagem baseada em blockchain para armazenamento e controle de acesso aos dados de certificados de alunos do ensino superior**. 2020. 146 f. Dissertação (Programa de Pós-Graduação em Computação) - Universidade Federal do Ceará, Quixadá.
- ANDROULAKI, Elli et al. Hyperledger fabric: a distributed operating system for permissioned blockchains. In: EUROSYS CONFERENCE, 13, 2018, Porto, Portugal. **Proceedings...** Porto, Portugal; ACM, 2018. Disponível em: <https://www.semanticscholar.org/paper/Hyperledger-fabric%3A-a-distributed-operating-system-Androulaki-Barger/7e00418545f3a8839eecd906dba167d85bd6e509>. Acesso em: 15 abril 2023.
- HYPERLEDGER. **A Blockchain Platform for the Enterprise**. 2023. Disponível em: <https://hyperledger-fabric.readthedocs.io/en/latest/>. Acesso em: 15 abril 2023.
- JUNQUEIRA, Natália Rodrigues. **Concessão de permissão a dados de saúde baseada em contratos inteligentes em plataforma de blockchain**. 2020. 90 folhas. Dissertação (Mestrado em Ciência da Computação) - Universidade Federal de Goiás, Goiânia.
- MENEZES, Leonardo Dias. **Blockchain e cartórios: uma solução viável?**. 2020. 61 f. Dissertação (Mestrado em Ciências) - Universidade de São Paulo, São Paulo.
- MOTTA, Fernando C. P. **O que é burocracia**. São Paulo: Editora Hedra Ltda., 2017.
- NAKAMOTO, Satoshi. **Bitcoin: A Peer-to-Peer Electronic Cash System**. [S.I.], 2009. Disponível em: <https://bitcoin.org/bitcoin.pdf>. Acesso em 18 abr. 2023.
- NARAYANAN, Arvind; BONNEAU, Joseph; FELTEN, Edward; MILLER, Andrew; GOLDFEDER, Steven. **Bitcoin and Cryptocurrency Technologies: A Comprehensive Introduction**. Nova Jersey, Princeton University Press, 2016.
- SWAN, Melanie. **Blockchain: blueprint for a new economy**. Sebastopol: O'Reilly Media, 2015.
- SZABO, Nick. **Formalizing and Securing Relationships on Public Networks**. **Firstmondey**, v. 2, n. 9, 1997. Disponível em: <https://firstmonday.org/ojs/index.php/fm/article/view/548/469>. Acesso em 18 abr. 2023.
- TAPSCOTT, Don; TAPSCOTT, Alex. **Blockchain Revolution: How the Technology Behind Bitcoin Is Changing Money**. New York: Penguin Random House, 2016.
- WOOD, Gavin. **Ethereum: A secure decentralised generalised transaction ledger**. Ethereum Yellow Paper, 2014.

APÊNDICE A – DESCRIÇÃO DOS CASOS DE USO

Descrições casos de uso	
1. Inserir um carro novo	A montadora envia os dados de um novo carro produzido, então é verificado se já existe um carro com o chassi informado, caso já exista o erro é retornado, é verificado se o ano não é maior que o ano atual, caso for maior o erro é retornado, então adiciona o carro no livro-razão
2. Adquirir um carro novo para venda	A concessionária seleciona o carro que quer comprar zero da montadora, é verificado caso o carro já seja de outra concessionária ou pessoa física, caso já tenha o erro é retornado, então atribui o carro para a concessionária e salva no livro-razão
3. Vender um carro	A concessionária envia os dados do comprador do carro e o carro escolhido, verifica se o carro é da concessionária que está fazendo a venda, caso não o erro é retornado, é verificado se o valor é menor que zero, caso seja o erro é retornado, é verificado se não possui nenhum restrição e nenhum financiamento, caso tenha o erro é retornado, por fim faz a troca do carro da concessionária com a pessoa física e os dados são salvos no livro-razão
4. Pedir para transferir um carro para a própria concessionária	A concessionária envia os dados do carro escolhido, é verificado se não possui nenhum restrição e nenhum financiamento, caso tenha o erro é retornado, verifica se o carro tenha um dono pessoa física, caso não tenha o erro é retornado, por fim cria uma pendência de transferência e salva no livro-razão
5. Fazer licenciamento do carro	O Detran envia os dados do carro para fazer o licenciamento, é verificado se não possui nenhum restrição, caso tenha o erro é retornado, é adicionado a data de vencimento do licenciamento um ano a mais da data atual, por fim é salvo no livro-razão
6. Adicionar restrição ao carro	O Detran envia os dados do carro para inserir uma restrição, a restrição é adicionada com a data atual e por fim salva no livro-razão
7. Remover restrição ao carro	O Detran envia os dados do carro para remover uma restrição, é verificado se a restrição existe, caso não o erro é retornado, é adicionada data de exclusão da restrição e por fim salva no livro-razão

8. Fazer manutenção no carro	A mecânica envia os dados da manutenção do carro, é verificado se não possui nenhuma restrição, caso tenha o erro é retornado, é verificado se a manutenção não tem KM do carro menor que a da última manutenção, caso sim o erro é retornado, por fim a manutenção é adicionado no carro e salvo no livro-razão
9. Trocar de carro com outra pessoa física	O Detran envia os dados da troca do carro, é verificado se a pessoa existe, caso não o erro é retornado, é verificado se não possui nenhuma restrição e nenhum financiamento, caso tenha o erro é retornado, é verificado se o valor da venda não é menor que zero, caso sim o erro é retornado, verifica se o novo dono não é o dono atual, caso sim o erro é retornado, por fim o carro muda de novo, o licenciamento é removido e os dados salvos no livro-razão
10. Aceitar trocar de carro com uma concessionária	O Detran envia os dados de confirmação da troca, a troca do carro é efetivada e as pendências removidas, por fim os dados são salvos no livro-razão
11. Indicar o financiamento para o carro	A Financiadora envia os dados do financiamento do carro, é verificado se já existe um financiamento ativo, caso sim o erro é retornado, e por fim o financiamento é adicionado e salvo no livro-razão
12. Liberar o carro do financiamento	A Financiadora envia os dados do carro para retirar o financiamento, é verificado se a financiadora que está retirando é a mesma que está ativa, caso não o erro é retornado, por fim o financiamento é removido e salvo no livro-razão
13. Cadastrar uma pessoa	O governo envia os dados da pessoa, e verificado se já existe uma pessoa com esse CPF, caso sim o erro é retornado, é validado se a data de nascimento está correta, caso não o erro é retornado, por fim a pessoa é salva no livro-razão
14. Indicar que a pessoa morreu	O governo envia os dados da pessoa, é verificado se a pessoa existe, caso não o erro é retornado, por fim é salvo que a pessoa não está viva no livro-razão