Fundação Educacional do Município de Assis
Instituto Municipal de Ensino Superior de Assis
Campus "José Santilli Sobrinho"

RAFAEL DE LIMA RODRIGUES

SISTEMA DE ASSINATURA DE DOCUMENTOS DIGITAIS UTILIZANDO A TECNOLOGIA BLOCKCHAIN

Assis/SP
2020

Fundação Educacional do Município de Assis
Instituto Municipal de Ensino Superior de Assis
Campus "José Santilli Sobrinho"

RAFAEL DE LIMA RODRIGUES

SISTEMA DE ASSINATURA DE DOCUMENTOS DIGITAIS UTILIZANDO A TECNOLOGIA BLOCKCHAIN

Projeto de pesquisa apresentado ao curso de Análise e Desenvolvimento de Sistema do Instituto Municipal de Ensino Superior de Assis – IMESA e a Fundação Educacional do Município de Assis – FEMA, como requisito parcial à obtenção do Certificado de Conclusão.

Orientando(a): Rafael de Lima Rodrigues
Orientador(a): Célio Desiró

Assis/SP
2020

FICHA CATALOGRÁFICA

RODRIGUES, Rafael de Lima.
Sistema de assinatura de documentos digitais utilizando a tecnologia Blockchain/ Rafael de Lima Rodrigues. Fundação Educacional do Município de Assis –FEMA – Assis, 2020.
66p.

1. Blockchain. 2. Documentos digitais.

CDD:
Biblioteca da FEMA

SISTEMA DE ASSINATURA DE DOCUMENTOS DIGITAIS UTILIZANDO A TECNOLOGIA BLOCKCHAIN

RAFAEL DE LIMA RODRIGUES

Trabalho de Conclusão de Curso apresentado ao Instituto Municipal de Ensino Superior de Assis, como requisito do Curso de Graduação, avaliado pela seguinte comissão examinadora:

Orientador: Célio Desiró

Examinador: Guilherme de Cleva Farto

Assis/SP
2020

RESUMO

Com o crescimento do uso tecnologia em todas as áreas da sociedade, as pessoas passam a criar relações com outras pessoas em pontos do mundo cada vez mais distantes. Porem quando falamos em relações contratuais jurídicas ainda encontramos uma certa dificuldade na inclusão da tecnologia, pois o mundo jurídico ainda se vale de métodos muitos arcaicos para dar validade uma assinatura ou a veracidade de um documento. Atualmente a forma mais utilizada é método de assinatura física com o reconhecimento em cartório desse documento. As tecnologias existentes ainda encontram grande resistência em sua implementação, pelo fato do alto custo, ou algumas vezes pelas vulnerabilidades existentes nesses sistemas. A utilização da tecnologia *Blockchain* tem potencial de revolucionar o ramo da identificação digital, seja por sua capacidade de universalização dos dados, ou por sua possibilidade de implementação a um custo abaixo das tecnologias existentes. Diante disso, o presente trabalho tem como objeto implementar uma aplicação que permite a assinatura de um documento de forma digital, e mantendo seus dados salvos em uma rede Blockchain. A pesquisa irá utilizar uma a linguagem Java juntamente com os frameworks Spring e o Hyperledger Fabric e a ferramenta IBM Blockchain Platform para Visual Studio Code. Ao final foi implementado uma aplicação Web que permite criar, assinar e validar um documento assinado de forma digital, bem como a visualização do histórico das transações realizadas a partir do documento mantido na rede blockchain.

Palavras-chave: Blockchain; Hyperledger Fabric; IBM Blockchain Platform; Documentos digitais.

ABSTRACT

With the growth in the use of technology in all areas of society, people are creating relationships with other people in increasingly distant parts of the world. But when we talk about legal contractual relations, we still find a certain difficulty in the inclusion of technology, because the legal world still uses archaic methods to give validity to a signature or the veracity of a document. Currently the most used form is the method of physical signature with the recognition in registry of this document. The existing technologies still find a large resistance in their implementation, due to the high cost, or sometimes due to the vulnerabilities existing in these systems. The use of Blockchain technology has the potential to revolutionize the branch of digital identification, either due to its capacity of data universalization, or due to its possibility of implementation at a cost below the existing technologies. Therefore, this work aims to implement an application that allows the signature of a document in a digital way, and keeping your data saved in a Blockchain network. The research will use a Java language together with Spring frameworks and Hyperledger Fabric and the IBM Blockchain Platform tool for Visual Studio Code. At the end it was implemented a Web application that allows the creation, signing and validation of a signed document in a digital way, as well as the visualization of the history of the transactions performed from the document maintained in the blockchain network.

Keywords: Blockchain; Hyperledger Fabric; IBM Blockchain Platform; Digital Documents

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

1. INTRODUÇÃO

Atualmente, a tecnologia está presente em nosso dia-a-dia a todo momento, em tudo que fazemos, seja enviar uma mensagem por meio de um aplicativo no *smartphone,* carros autônomos que estacionam sozinhos em uma vaga na rua, quando compramos algo pela internet, entre outras coisas.

Esse avanço tecnológico vem permitindo que as pessoas terem acesso a informações sobre tudo o que acontece no mundo em tempo real, e se conectar e relacionar com outras pessoas mesmo com enormes distancias entre elas. Entretanto, algumas áreas ainda estão começando a entrar nesse novo mundo, como exemplo podemos citar o mundo do direito.

Nos últimos anos grandes mudanças foram feitas, sendo o exemplo mais significativo o fim dos processos físicos, os quais foram passaram a ser substituídos pelos processos digitais, onde os processos já existentes foram digitalizados e nos novos já nascem digitais.

Porém, quando falamos de assinatura de um contrato por meios digitais, seja ele de compra e venda de um bem, ou de uma prestação serviços encontramos uma certa dificuldade para que esses documentos possam ser protegidos pelas regras direito, e as pessoas possam se sentir respaldadas para o caso de um imprevisto.

Um dos aspectos mais relevantes para o Direito concerne à validade do documento eletrônico, pois, uma vez assinado o documento, não mais poderá este ser alterado, sem que a assinatura eletrônica seja invalidada (BEHRENS, 2005).

Um exemplo dessas dificuldades, é para dizermos com absoluta convicção que aquelas pessoas que estão celebrando o contrato, que estejam comprando vendendo ou produto, realmente existem, e realmente consentem com este ato.

A Medida Provisória 2.200-2 de 24 de agosto de 2001, a estabelece a Infraestrutura de Chaves Públicas Brasileira – ICP-Brasil como um modelo seguro para assinatura de documentos oficiais ou particulares digitais por meio de identidades digitais (BRASIL, 2001).

Porém, todas essas formas de identificação digital ainda possuem certas vulnerabilidades, seja no momento de garantir identificação da pessoa titular da identidade digital no momento da assinatura do documento, bem como ao garantir a veracidade dos documentos assinados por esse titular.

Segundo Menke (2017):

> *Uma das vulnerabilidades das assinaturas eletrônicas que se baseiam em senhas, é que, na maioria dos casos, o próprio titular da assinatura é quem define a sua senha e a armazena no sistema de seu interlocutor (por exemplo, um site de comércio eletrônico). Isso faz com que um fraudador possa criar uma identificação falsa, valendo-se de dados de terceiro que ignora completamente o ato ilícito praticado. Esta prática é comum na criação de contas de e-mail falsas.*

Para Junior (2001), quando se fala em autenticidade de um documento não nos referimos à impossibilidade de sua reprodução, mas à sua inalterabilidade, ou conformidade com o que pretendeu o autor.

Diante disso, a utilização de tecnologia *blockchain* se mostra uma ótima ferramenta para garantir a integridade dos dados do documento assinado digitalmente, pois a partir do seu princípio de imutabilidade os dados estariam protegidos de qualquer alteração.

O blockchain vem atraindo a atenção dos bancos devido ao seu potencial para agilizar as complexas e extensas redes de pagamentos e liquidação da indústria, e ao fazer assim, simultaneamente minimize riscos e despesas (THE INSTITUTE OF INTERNATIONAL FINANCE, 2015).

2. DESENVOLVIMENTO DO TRABALHO

2.1. PROBLEMATÁICA

Modelos de assinaturas digitais se utilizam de par chaves, sendo uma privada e outro pública. A privada fica no poder do titular da identidade, e é utilizada para assinar o documento digitalmente, e a pública é utilizada para verificar a autenticidade da assinatura.

Em grande parte dessas assinaturas, os dados são armazenados de forma centralizada, ou em alguns casos descentralizada e protegidas com criptografias, como MD5 ou SHA256. Porém, como mencionado por Guelfi (2011), essas formas de criptografias tem sua vulnerabilidade comprovada por meio de um ataque com supercomputadores, e no caso do MD5 até mesmo um *notebook* de 1.6 Ghz pode realizar esse ataque.

Segundo o *Identity Theft Resource Center* (2017) apenas nos Estados Unidos mais de 170 milhões de registros ficaram expostos através de ataques de *hackers*. Quando levado a uma escala global é quase que impossível calcular a quantidade de dados que foram expostos. Desta forma, se faz necessário, a utilização de algum método ou tecnologia, que garanta de forma substancial a invulnerabilidade e inalterabilidade das informações e transações realizadas.

Nesse cenário, a aplicação do *Blockchain* para o tratamento das vulnerabilidades que existem das identidades digitais aparece como uma alternativa inovadora e muito promissora (BATISTA; DIAS; SILVA, 2018),.

2.2. OBJETIVO

Neste contexto, este trabalho tem como objetivo, abordar o conceito de base de dados distribuídas do *Blockchain* e suas vantagens, e desenvolver uma aplicação Web que demonstre a funcionalidade da criptografia *Blockchain* na proteção dos dados de um documento e na criação de uma identidade digital desde documento bem como das partes que assinam este documento.

Permitindo assim, inserir dentro do mundo jurídico documentos digitais, que possuam a mesma validade jurídica que os documentos físicos, seja perante as partes, ou quando for necessário requerer a execução desses documentos perante o judiciário.

2.3. PÚBLICO ALVO

O presente trabalho tem como público alvo profissionais do direito ou áreas afins, que necessitam realizar assinaturas de documentos de forma digital, bem como usuários comuns que tenha essa necessidade de assinar algum documento de uma forma segura e acessível.

2.4. JUSTIFICATIVA

Esse estudo tem uma grande importância, tendo em vista que cada vez mais, a sociedade anseia por ferramentas que possibilitem que as pessoas de diversas partes do mundo possam criar relações de forma prática, rápida e confiável, e que também diminuam os custos dessas relações.

A criação de uma ferramenta que garanta a segurança dos dados das pessoas e das coisas poderá possibilitar um crescimento exponencial das relações internacionais, seja na compra e venda e de produtos, assinatura de contratos, ou qualquer outra forma que se necessite a autenticação de documentos e identificação de pessoas.

2.5. ESTRUTURA DE DESENVOLVIMENTO DO TRABALHO

O presente trabalho será composto da seguinte estrutura:

Capítulo 1 – Introdução: No primeiro capítulo será feita uma contextualização a área que será objeto de estudo desse trabalho.

Capítulo 2 – Desenvolvimento do Trabalho: Neste capítulo será abordado pontos como a problemática, os objetivos do trabalho, qual o público alvo, e a justificativa para o desenvolvimento do presente trabalho.

Capítulo 3 – Fundamentação Teórica: Neste terceiro capítulo será realização uma análise conceitual da tecnologia blockchain, como seu surgimento, tipos existentes e conceito de estudo realizados anteriormente.

Capítulo 4 – Método de Desenvolvimento – Neste quarto capítulo será apresentada as ferramentas utilizadas para o desenvolvimento do presente trabalho.

Capítulo 5 – Diagramas – O quinto capítulo será composto dos diagramas levantados durante o processo de desenvolvimento do trabalho.

Capítulo 6 – Conclusões – Neste último capítulo serão apresentadas as vantagens e os resultados obtidos como presente trabalho

3. FUNDAMENTAÇÃO TEORICA

Neste capítulo será feita uma introdução sobre a tecnologia *Blockchain,* e alguns conceitos fundamentais dessa tecnologia que serão utilizados no desenvolvimento deste trabalho.

3.1. BLOCKCHAIN

Em 2008, Satoshi Nakamoto surgiu com um design de uma tecnologia que revolucionou o campo da pesquisa na comunidade de sistemas distribuídos. Ele apresentou um design de processo de troca de dinheiro *peer-to-peer* que foi compartilhado e distribuído (GUPTA; SADOGHI, 2018).

> *Uma Blockchain é uma cadeia de blocos em ordem cronológica, que são resolvidos através de Proof-of-Work (prova de trabalho em tradução literal). O encadeamento é feito adicionando o hash do bloco anterior para o bloco atual, o hash do atual bloco para o próximo bloco e assim por diante. Consecutivos blocos aninhados garantem que as transações tenham uma ordem cronológica, portanto, uma transação não pode ser alterada sem alterar seu bloco anterior e todos os seguintes blocos* (AITZHAN; SVETINOVIC, 2018).

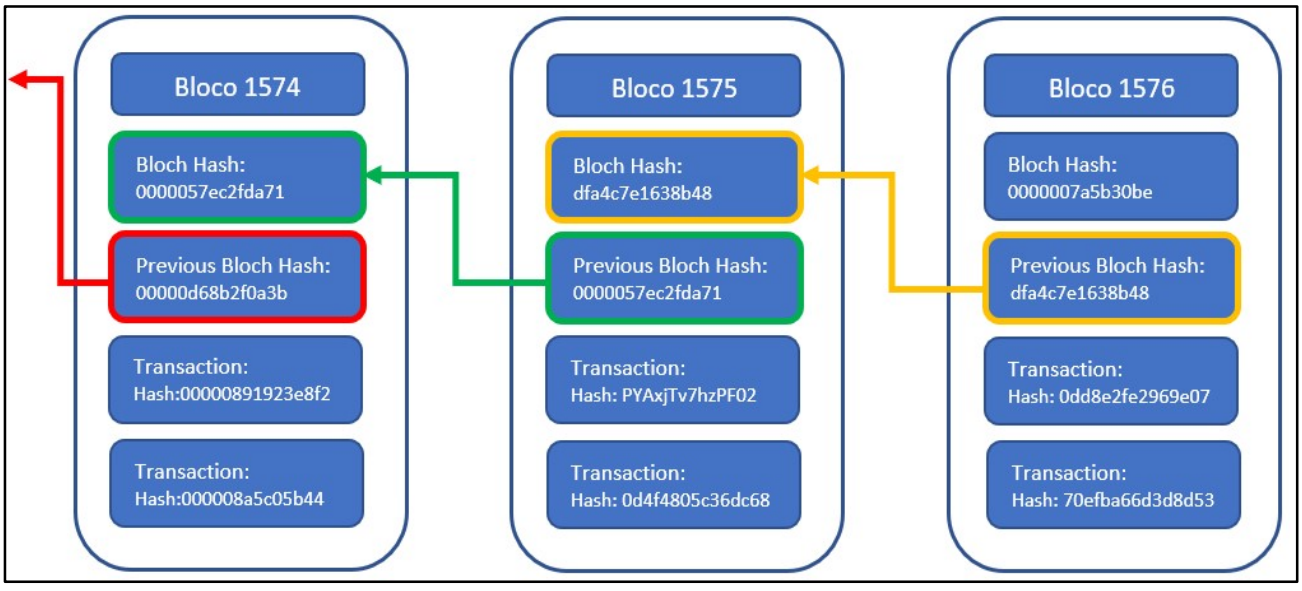

Figura 1: Representação de uma cadeia *Blockchain* (VITORINO, 2018)

A tecnologia *Blockchain* é um facilitador que torna muitos serviços de processos e transações mais transparentes, descentralizados, democráticos e seguros sem a que exista a necessidade um terceiro intermediário (RIVERA et al., 2017).

Muito embora, o modelo de *blockchain* apresentado por Nakamoto, tenha seus dados públicos, ou seja, qualquer pessoa pode consultar as transações realizadas, existem outras formas de blockchain. Atualmente podemos classificar as *blockchains* em públicas ou privadas.

3.1.1. Públicas

O modelo de *blockchain* publica talvez seja a forma mais conhecida dessa tecnologia, pois é modelo que deu origem esse sistema de armazenamento de dados. O modelo proposto por Nakamoto é um livro razão público onde qualquer indivíduo pode participar e ler os dados da rede.

Podemos chamar de publica uma *blockchain* se cada participante puder lê-la e usá-la para realizar transações, mas também puder participar do processo de criação do consenso. Dessa forma não existe um registrador central, e não há a necessidade terceiros intermediários durante o processo (GUEGAN, 2017).

Esses tipos de sistemas são baseados em criptoeconomia, pois os participantes são estimulados a processarem as transações em troca de uma compensação econômica. Os participantes realizam o processamento das transações em suas máquinas em troca recebem criptomoedas, é o caso por exemplo do Bitcoin, ou dos *tokens* da Ethereum (GUEGAN, 2017).

3.1.1. Privadas

Por outro lado, *blockchains* privados permitem que apenas *nodes* (nós) façam parte do processo de consenso, e somente alguns subconjuntos desses *nodes* podem gerar o próximo bloco. Esse tipo de sistema pode ser utilizado em organizações bancarias que permitem que seus clientes participem do consenso, enquanto os funcionários do banco somente confirmam os resultados criando o bloco (GUPTA; SADOGHI, 2018).

Nesse tipo de blockchain não há a ideia de sistema de recompensa aos participantes do bloco que realizam a validação das transações. Além disso, existem uma limitação de quem pode participar da rede, ou seja, aquele que desejar participar deverá primeiro obter uma permissão da rede, que é geralmente emitida por outro grupo de nodes.

Uma vantagem que uma rede privada tem sobre a pública é a questão da escalabilidade. Afinal, uma rede pública possui um algoritmo de consenso de prova de trabalho (*Proof-to-Work*), haverá a busca pelo consenso proposto que no caso seria um padrão no *hash* de identificação do bloco – por exemplo, o *hash* de cada bloco deve possuir os 5 primeiros caracteres igual zero – dessa forma os usuários que contribuem para a rede irão realizar uma “mineração”, seja irão ficar gerando *hash* diferentes até de se alcance o consenso (AITZHAN; SVETINOVIC, 2018).

Após essa mineração a validação do novo bloco passa pôr uma verificação de toda a rede, de forma a comparar se *hash* anterior bloco anterior contido novo bloco é valido. Com isso, quanto mais a rede cresce, mais tempo e poder de computação são necessários para essa validação (GREVE et al., 2018).

Vale salientar que a questão da escalabilidade é um ponto que leva sempre a muitos questionamentos do sistema de blockchain, afinal ela tem com princípios a imutabilidade dos dados, e o fato que eles nunca serão apagados. O que com o passar do tempo, principalmente nas blockchain publicas, pode gerar uma grande dificuldade ao validar uma transação, haja vista que nesse tipo de blockchain é necessário verificar toda a rede para saber se aquele novo bloco pertence ao consenso e que nenhum dado da rede foi alterado.

Dessa forma, uma *blockchain* privada pode atender melhor os interesses corporativos ou privados, onde cada participante tem uma função especifica definida podendo inclusive se organizar em grupos. Nessa categoria de *blockchain* podemos citar o *Hyperledger Fabric* (GREVE et al., 2018).

3.2. ALGORITMO DE CONSENSO

Um ponto chave em todos os tipos de *blockchains* são os chamados algoritmos de consenso. O consenso é o principal componente utilizado para manter a consistência de um *blockchain*.

Pode-se dizer que o algoritmo de consenso é uma solução Problema dos Generais Bizantinos (PGB) que consiste no dilema de atingir um consenso entre os usuários que no caso seriam os generais, os quais possuem um objetivo em comum. Dentre os generais podem existir traidores os quais trabalham contra o processo. A ideia é que os generais leias consigam atingir seus objetos sem que os traidores consigam interrompê-los (ALIAGA; HENRIQUES, 2017).

Em outras palavras o algoritmo de consenso é um algoritmo que busca verificar que aquela transação que está sendo enviada ao bloco está de acordo com a regra definida pelos usuários ou membros do bloco, para que somente depois seja inserido na rede.

Existem diferentes formas de consenso, que podem variar a partir da finalidade para a qual foi criada determinado *blockchain,* e de acordo com tipo de *blockchain*, que pode ser pública ou privada.

No modelo de *blockchain* proposto por Nakamoto e bem como em outras *blockchain* publicas o algoritmo de consenso utilizado é o *Proof-of-Work,* onde qualquer nó pode participar do protocolo de procurar acrescentar blocos, de forma a aumentar a *blockchain.* Nesse modelo é resolvido um enigma criptográfico, que é chamado de mineração pela dificuldade computacional do processo (REBELLO et al., 2019).

Nos casos de blockchain privadas são utilizados outros modelos de consenso, pois essa modelo tem em geral um número restrito de participantes, além de suas funções na rede terem níveis diferentes.

As blockchain privadas utilizam consensos com algoritmo tolerância de falha de parada ou bizantino. Nos *blockchain* privados com tolerância a falhas, apenas os nós autorizados, denominados validadores, executam um protocolo de consenso para acrescentar blocos e estender a corrente de blocos, que apresenta a mesma evolução em todos que participam do consenso (REBELLO et al., 2019).

3.3. HYPERLEDGER FABRIC

O *Hyperledger Fabric* é um framework de blockchain de código aberto desenvolvido para redes privadas de negócio ou de permissão e é hospedada dentro projeto *Hyperledger* o qual é mantido pela *The Linux Foundation.*

Diferente de outras tecnologias, o *Hyperledger Fabric* permite a utilização de consenso modular e serviço de associação, dessa forma algoritmos de consenso e verificação de identidades são *plug-and-play*, o que resulta em uma arquitetura blockchain universal compatível com a maioria dos modelos de negócios existentes.

O *Hyperledger Fabric* se utilizada de serviços de contêineres para hospedar os *smart contract* que compõem a lógica de aplicação do sistema, o que é um dos pontos da modularidade dessa plataforma. Dessa forma os contratos podem ser escritos em linguagem de programação padrão, e não possuem acesso direto ao *ledger* (livro-razão da blockchain, onde estão os escritos os blocos com suas transações), pelo fato desses *smart contract* serem executados de forma isolada nos contêineres (HYPERLEDGER, 2020).

Para essa plataforma alcance possa alcançar seu objeto, alguns componentes ou recursos são essenciais. A seguir uma prevê descrição de cada um deles.

3.3.1. Assets

Os *Assets* (Ativos) podem ser definidos como o objeto da transação, seja ele tangível como imóveis, hardware, ou intangíveis como contratos e propriedade intelectual.

No *Hyperledger Fabric* os ativos serão representados sempre por uma chave e um valor, tendo suas alterações estados registradas no *ledger* (HYPERLEDGER, 2020).

3.3.1. Chaincode

O *chaincode* é um software onde será definido um ou mais ativos, as transações para modificá-los, pode se dizer assim que ele é a lógica de negócio da aplicação. Esse software pode ser comparado apenas para fins de compreensão do conceito as classes na programação orientada a objetos. Assim como projeto Ethereum, na plataforma blockchain da IBM a ser utilizada neste trabalho, os *chaincodes* tem a denominação de *smart contract.*

Dentro dos *chaincodes* estarão as regras para ler ou alterar os pares de valores-chaves dos ativos ou de outras informações do banco de dados (HYPERLEDGER, 2020).

3.3.1. Ledger

O *ledger* ou razão é o registro sequencial e resistente a violação de todas transações de estado no *Fabric.* O Estado de transação é o resultado das invocações dos *chaincodes* submetidas pelas partes participantes. Cada transação resulta em um conjunto de pares do tipo chaves-valores de ativos que são confirmados no razão, podem ser um criação, atualização ou exclusão (HYPERLEDGER, 2020).

O ledger é composto por cadeia de blocos que armazenam os registros imutáveis de forma sequenciada, além de um banco de dados de estado o qual mantem o estado atual do ledger (HYPERLEDGER, 2020).

3.3.1. Peer nodes

Os peers não fundamentais na rede pois hospedam o ledger e contratos inteligentes. Eles executam o *chaincode*, acessam os dados do *ledger*, endossam as transações e faz uma interface com a aplicação. Alguns desses pares podem endossar outros peers, ou endossantes. Os *chaincodes* podem criar uma política de endosso, a qual irá definir as condições necessárias para uma transação de endosso válida (MAHESHWARI, 2018).

3.3.1. Channel

Channel ou canais são uma rede de estrutura lógica formada por uma coleção de *peers*. Isso permite um grupo de peers criar uma legder de transação separado (MAHESHWARI, 2018).

Através dos canais pode se criar uma transação privada, onde apenas os participantes daquele canal podem ter acesso, mesmo que dentro da rede exista outros participantes, dessa forma uma rede blockchain do *Hyperledger Fabric* pode ter um ou mais canais.

3.3.1. Organizations

Organizations ou organizações são os membros da rede. Uma rede do Hyperledger Fabric é composta por peers de propriedade e contribuição de diferentes organizações (MAHESHWARI, 2018).

A rede somente existe porque as organizações contribuem com seus recursos individuais para uma rede coletiva. Cada peer tem uma identidade a qual foi atribuída por Provedor de Serviços de Membros (*Membership Service Provider* – MSP) de sua própria organização (MAHESHWARI, 2018).

3.3.1. Membership Services Provider (MSP)

O Fabric é uma rede autorizada, sendo assim seus participantes precisam provar de alguma maneira sua identidade para os demais participantes da rede, para que assim possa realizar transações.

As identidades são emitidas pelas Autoridades de Certificação gerando uma chave pública e uma chave privada que formam um par de chaves que pode ser utilizado para provar a identidade (HYPERLEDGER, 2020).

Em um sistema de identidades digitais com um par de chaves pública e privada, a chave pública é utilizada para identificação, e pode ser compartilhada, por outro lado uma chave privada nunca pode ser compartilhada, pois ela é utilizada para a assinatura. Sendo assim, é necessário um mecanismo que possa provar a validade dessa identidade, no Hyperledger Fabric esse mecanismo é o MSP.

O MSP em um Ordering Service (Serviço de encomenda) contém a chave pública do peer que é utilizada para verificar se a assinatura contida na transação é válida.

Assim, o MSP é o mecanismo que permite que essa identidade seja confiável e reconhecida pelo resto da rede, sem nunca revelar a chave privada do membro (HYPERLEDGER, 2020).

Na figura 2 temos um exemplo de como funciona o MSP. Neste caso os cartões que uma pessoa possui em sua carteira são as identidades que ela possui, já o MSP seria a lista de cartões aceitas um estabelecimento.

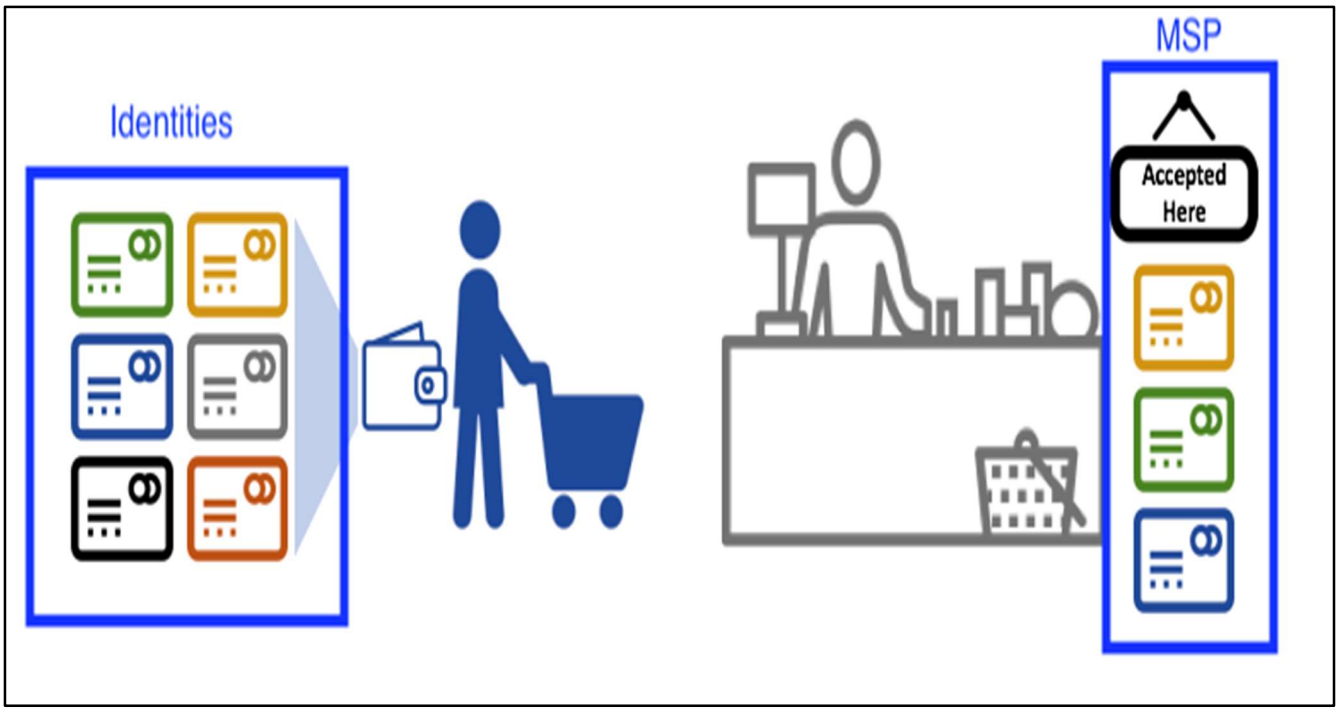

Figura 2: Modelo de exemplificado de um MSP (HYPERLEDGER, 2020)

Com isso podemos concluir que uma identidade que não está listada no MSP não pode realizar uma transação nesse bloco.

3.3.1. Ordering Service

Em *blockchain* que não existe permissão, onde qualquer nó do bloco pode participar do processo de consenso, sendo as transações ordenadas e agrupadas em blocos, utiliza-se um algoritmo probabilístico de consenso para garantir a consistência do ledger. Entretanto, registros divergentes podem ser aceitos em algum momento.

Diferente dos *blockchains* que não exigem permissão, o *Hyperledger Fabric* possui um algoritmo determinístico de consenso que garante que qualquer bloco validado por um peer está correto.

O *Ordering Service* ou serviço de encomenda, como o próprio nome já diz, é um serviço que empacota as transações em blocos a serem entregues aos peers de um determinando canal, garantindo assim a entrega de uma transação na rede. Esse serviço se comunica com os peers e os endossa.

A figura 3 demonstra a funcionalidade de um *Ordering Service* do momento em que uma transação é recebida pelo serviço até empacotamento da transação no bloco.

A primeira função de um nó ordenador é empacotar as atualizações do *ledger* propostas. Na representação acima A1 envia uma transação T1 que será endossada pelo E1 e E2 para o ordenador O1. De forma paralela a aplicação A2 envia a transação T2 que será endossada pelo E1 do ordenador O1. Em seguida O1 empacota a transação T1 da aplicação A1 e a transação T2 da aplicação A2 juntamente com outras transações de outras aplicações na rede para o bloco B2. No bloco B2 temos as transações na seguinte ordem: T1, T2, T3, T4, T5 e T6. Essa ordem pode não ser exatamente a mesma ordem que essas transações chegaram ao ordenador (HYPERLEDGER, 2020).

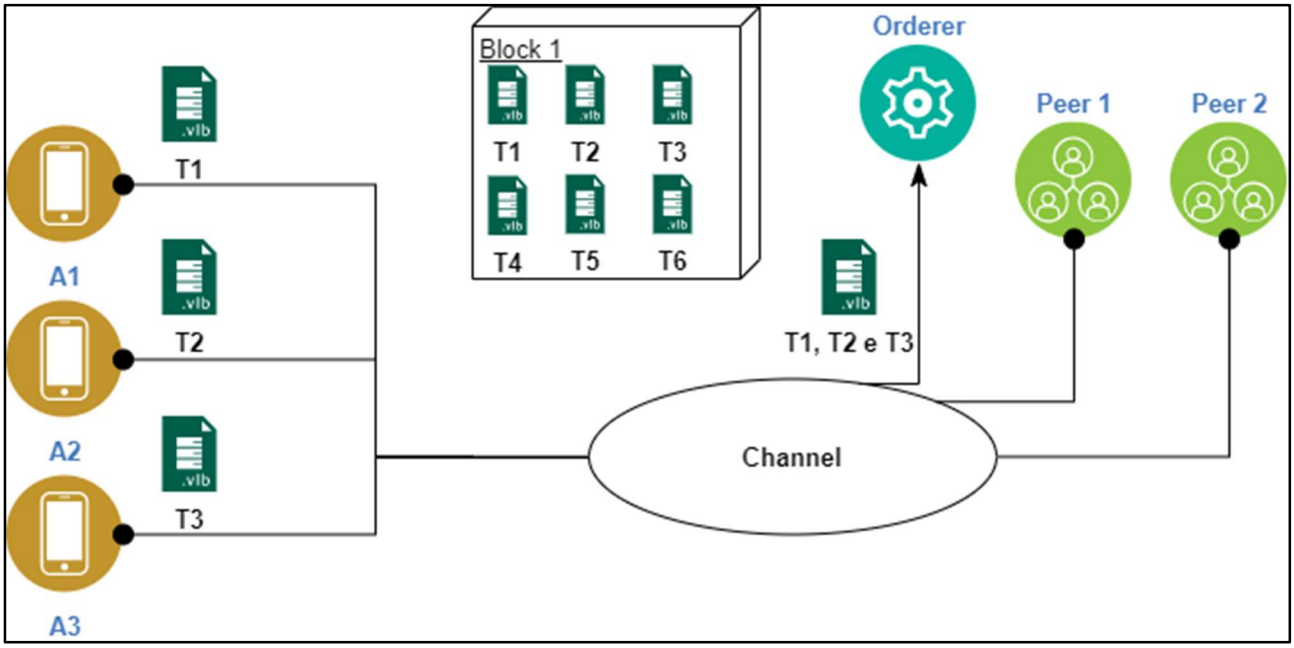

Figura 3: Encomenda e empacotamento de transações baseado em HYPERLEDGER (2020)

Conforme demonstra a representação na figura 4, as transações são validadas de cada vez pelos *peers,* para somente em seguinte serem comitadas e salvas no bloco a ser criado.

No caso de uma transação não ser validada, o bloco ainda será incluído no ledger, porem essa transação terá uma informação de invalida e não poderá ter seu estado alterado.

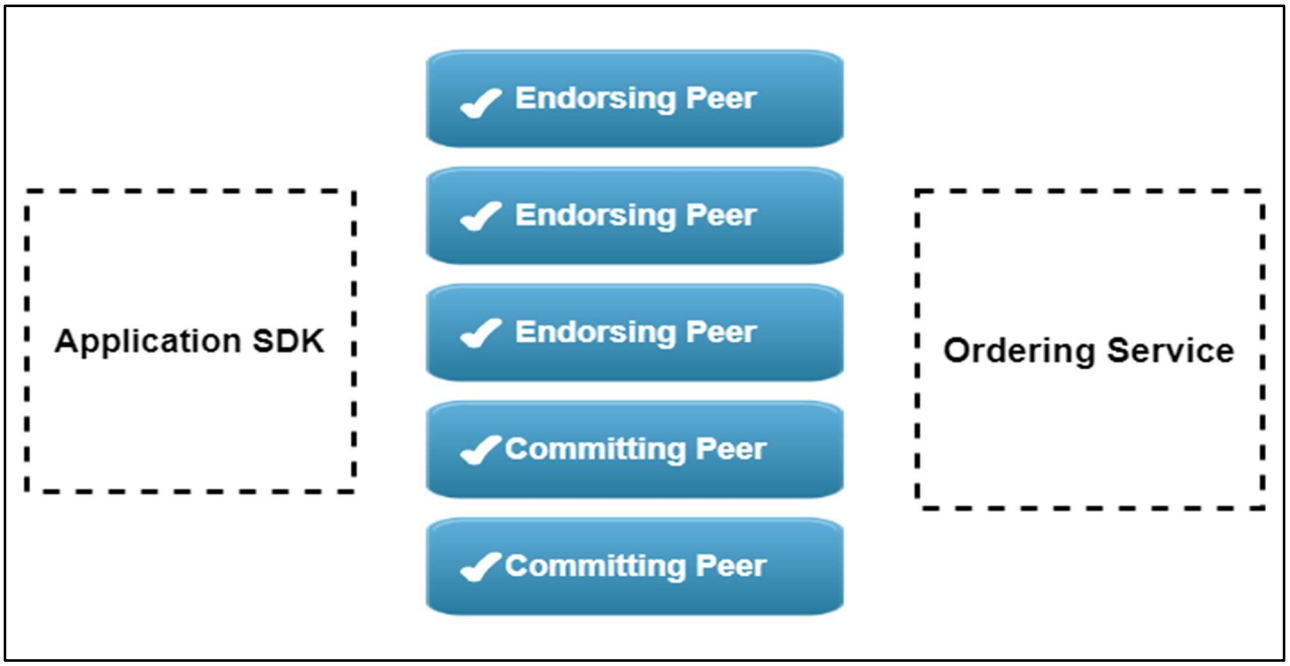

Figura 4: transações validadas uma a uma pelos *peers* de um bloco baseado em PALANIVEL (2018)

A seguir na figura 5 temos uma representação de rede *Hyperledger Fabric* com duas organizações participando. A transação se inicia com aplicação cliente que envia a requisição de transação. A organização a qual essa aplicação é membro recebe a requisição, que em seguida será endossada pelos *peers* antes de ser inserida no *ledger.*

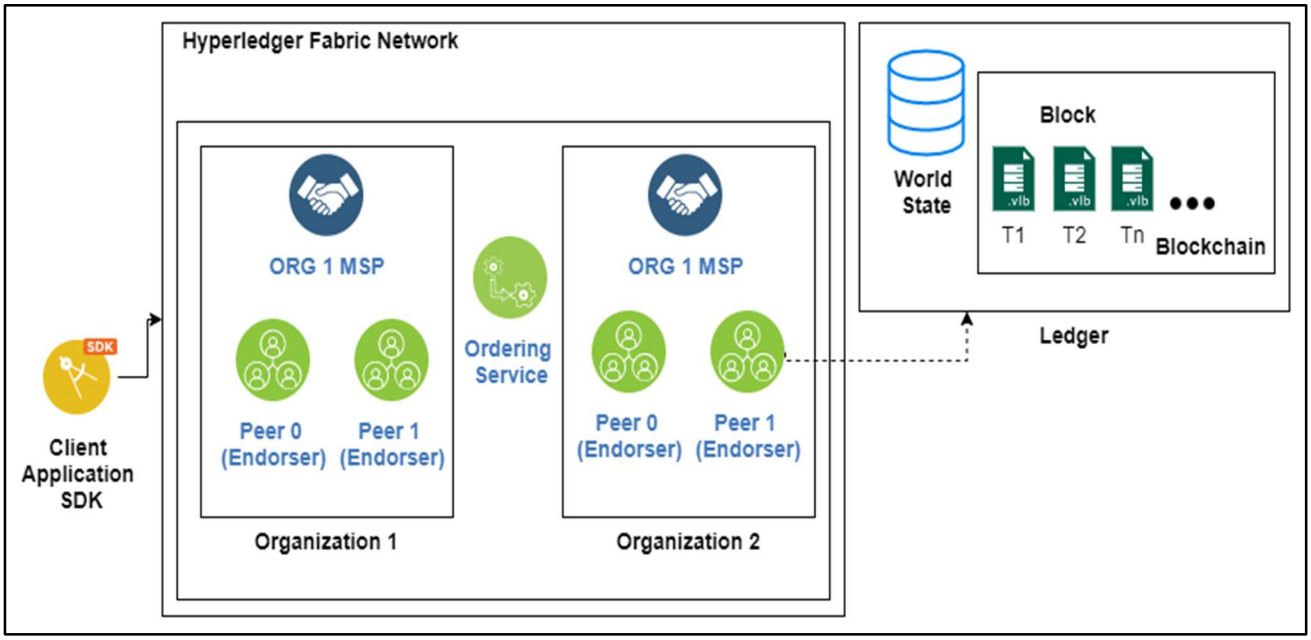

Figura 5: Componentes da rede *Hyperledger Fabric* baseado em MAHESHWARI (2018)

4. MÉTODO DE DESENVOLVIMENTO

Para a elaboração deste trabalho de conclusão de curso serão consultados livros, sites e tutoriais que forneçam informações referentes aos softwares usados para a elaboração do sistema e as exigências legais para validação de documentos digitais.

Na primeira etapa serão levantados requisitos necessários para elaboração do software, e posterior validação desses requisitos, por meio Diagramas de Caso de Uso, Atividades, Sequência, Classe e Entidade Relacionamento.

Na segunda etapa será desenvolvido o software com base na documentação elaborada na primeira etapa.

4.1. JAVA

Java foi concebida por James Gosling, Patrick Naughton, Chris Warth, Ed Frank e Mike Sheridan na Sun Microsystems, em 1991. No início, a linguagem se chamava “Oak”, mas foi renomeada como “Java” em 1995 (SCHILDT, 2015).

Ao final de 1998, foi lançada a Plataforma Java 2 juntamente com sua implementação criada pela Sun, a *Java 2 Platform, Standard Edition Software Development Kit* (J2SE SDK), a qual foi sendo aprimorada nos anos seguintes (WINDER; ROBERTS, 2009). No ano de 2004, houve uma evolução crítica e significativa com o lançamento pela Sun da Plataforma Java 2 na forma da J2SE v5.0, que incluiu tipos genéricos, anotações e vários outros recursos altamente desejáveis(WINDER; ROBERTS, 2009).

Java é uma linguagem Orienta a Objeto. Atualmente, o desenvolvimento orientado a objeto é amplamente utilizado, pois esse tipo de linguagem resolve diversos problemas que aparecem durante o desenvolvimento de softwares, fornecendo soluções variáveis e práticas.

A característica mais marcante dessa linguagem é que os programas criados nela não são compilados em código nativo da plataforma. Eles são compilados para um *bytecode*, que é executado por uma máquina virtual. Isso permite que os desenvolvedores criem seus programas uma única vez e depois executar em qualquer plataforma (BRITO, 2014)

Segundo Winder e Graham (2009):

> *O ambiente de execução Java não só fornece um motor de execução para a linguagem como proporciona um ambiente de processamento seguro. Sempre que um código Java é carregado verifica-se se ele não foi adultera- do, se o código compilado tem consistência interna e com relação ao padrão Java. Além disso, cada programa Java é associado a um gerenciador de segurança que, durante a execução, verifica se o programa tem permissão*
> *para acessar os recursos, como arquivos e conexões de rede, que ele tentar usar.*

4.1.1. Spring MVC

Spring não é apenas um único framework, mas sim um conjunto de ferramentas que resolvem diversas situações no cotidiano do programador, auxiliam na criação de aplicações Java de forma simples e flexível (AFONSO, 2017).

O Spring MVC *(Model, View, Controller)*, ajuda no desenvolvimento de aplicações web de forma mais robusta, e flexível com uma separação de responsabilidade no tratamento da requisição.

O *Model* é responsável pelo é onde a regra de negócio da aplicação fica, é o coração da aplicação. Nesse pacote é onde vamos calcular, processar a informação recebida e chegar ao resultado esperado. O *View,* é por onde o usuário interage com a aplicação. É por onde ele envia suas requisições e por onde ele recebe os resultados solicitados. Por fim o *Controller,* como o próprio nome já diz, ele controla as requisições recebidas, melhor dizendo, ele recebe essa requisição e direciona para onde ele deve ir, onde ela será processada. O *Controller,* faz a ligação entre o que o usuário solicita e quem deve processá-la.

O presente trabalho utiliza o Spring MVC para o desenvolvimento da aplicação lado cliente, responsável enviar as transações para rede *blockchain Hyperledger*.

4.2. VISUAL STUDIO CODE

No desenvolvimento será utilizado a Visual Studio Code com a linguagem de programação Java na construção de regras de negócio, acesso a dados seguindo o padrão Orientação a Objetos e *Design Patters*.

O Visual Studio Code é um leve, mas poderoso editor de código fonte para desktop e está disponível para Windows, macOS e Linux. Possui suporte embutido para JavaScript, TypeScript e Nodes.js e com um rico ecossistema com extensão para outras linguagem como C++, C#, Java, Python, PHP, Go (MICROSOFT, 2020).

4.3. IBM BLOCKCHAIN PLATFORM EXTENSION FOR VISUAL STUDIO CODE

A extensão IBM Blockchain Platform Extension é uma plataforma que auxilia os desenvolvedores a criar, testar, e depurar *smart contract* (contrato inteligentes) além de conectar-se a ambientes Hyperledger Fabric e criar aplicativos que transacionam em sua rede *blockchain* (IBM BLOCKCHAIN, 2020).

A extensão contém uma instancia pré-configurada do Hyperledger Fabric (atualmente versão 1.4) para qual a extensão extrai e usa as imagens apropriadas do Docker (IBM BLOCKCHAIN, 2020).

A interface da plataforma é dívida em quatro blocos que são *Smart Contract, Local Fabric Enviroments, Fabric Gateways e Fabric Wallets.*

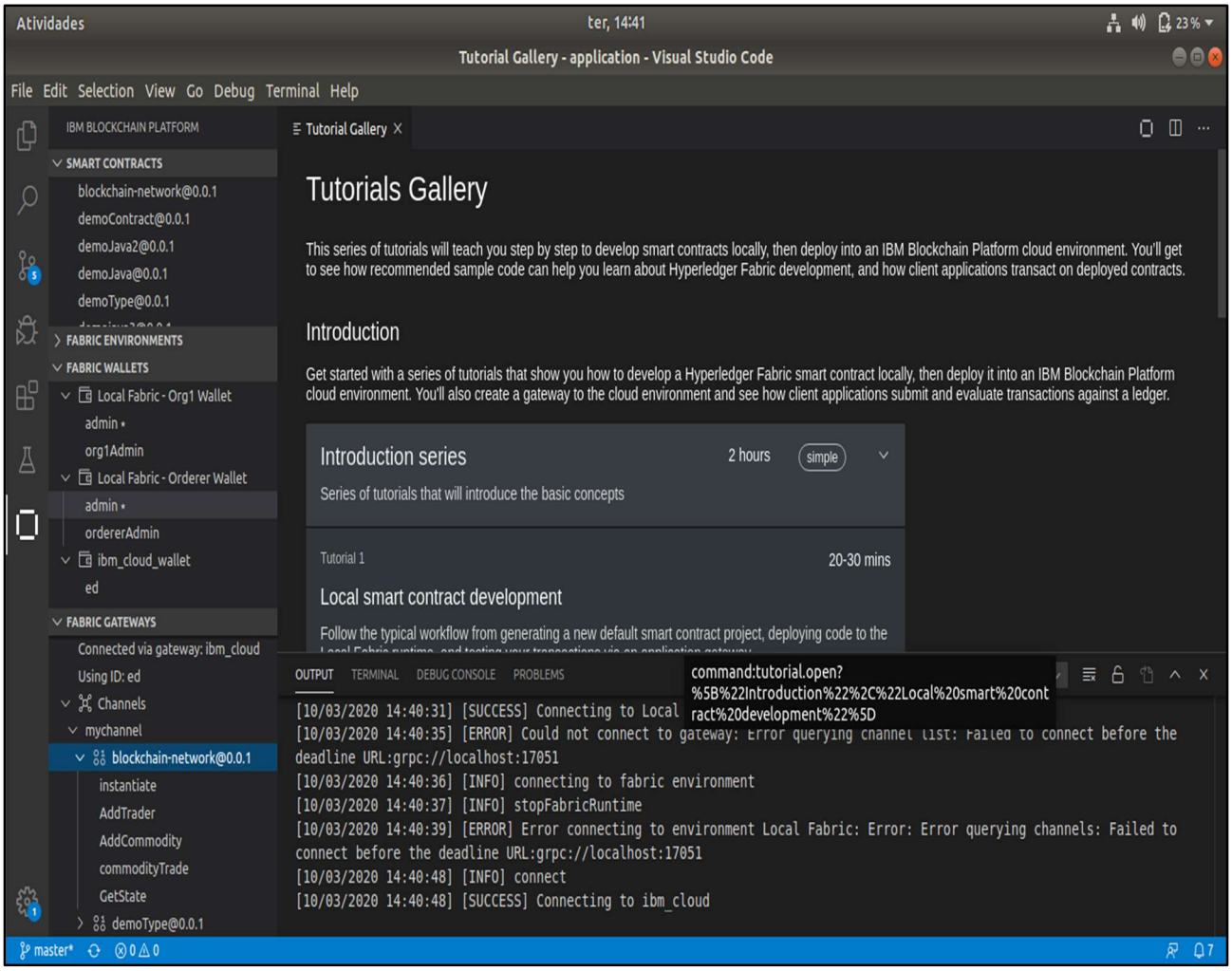

Figura 6: Visão geral da IBM Blockchain Platform Extension for Visual Studio Code

4.3.1. Smart Contract

A sessão smart contract é onde irá constar os smart contract gerados através do projeto que está sendo desenvolvido. Ele possui três funções, a primeira é Create New Project que permite criar um projeto baseado no Hyperledger Fabric. A segunda é Import Package, que permite importar um projeto já existente como por exemplo do rum repositório GitHub. A Terceira e última é Package Open Project, que permite empacotar o smart contract para que seja enviado para uma aplicação em nuvem como o IBM Cloud.

O código a seguir demonstra um modelo de *smart contract* com os métodos criar, ler, atualizar e excluir. Cada método deve possuir a anotação *Transaction* para informar o framework que esse método é uma transação do *smart contract*.

```
@Default
public class MyAssetContract implements ContractInterface {
   public  MyAssetContract() { }

@Transaction()
   public boolean myAssetExists(Context ctx, String myAssetId) {
      byte[] buffer = ctx.getStub().getState(myAssetId);
      return (buffer != null && buffer.length > 0);
   }

   @Transaction()
   public void createMyAsset(Context ctx, String myAssetId, String value) {
      boolean exists = myAssetExists(ctx,myAssetId);
      if (exists) {
         throw new RuntimeException("The asset "+myAssetId+" already exists");
      }
      MyAsset asset = new MyAsset();
      asset.setValue(value);
      ctx.getStub().putState(myAssetId, asset.toJSONString().getBytes(UTF_8));
   }

   @Transaction()
   public MyAsset readMyAsset(Context ctx, String myAssetId) {
      boolean exists = myAssetExists(ctx,myAssetId);
      if (!exists) {
         throw new RuntimeException("The asset "+myAssetId+" does not exist");
      }

      MyAsset newAsset = MyAsset.fromJSONString(new String(ctx.getStub().getState(myAssetId),UTF_8));
      return newAsset;
   }

   @Transaction()
   public void updateMyAsset(Context ctx, String myAssetId, String newValue) {
```

```
        boolean exists = myAssetExists(ctx,myAssetId);
        if (!exists) {
            throw new RuntimeException("The asset "+myAssetId+" does not exist");
        }
        MyAsset asset = new MyAsset();
        asset.setValue(newValue);

        ctx.getStub().putState(myAssetId, asset.toJSONString().getBytes(UTF_8));
    }

    @Transaction()
    public void deleteMyAsset(Context ctx, String myAssetId) {
        boolean exists = myAssetExists(ctx,myAssetId);
        if (!exists) {
            throw new RuntimeException("The asset "+myAssetId+" does not exist");
        }
        ctx.getStub().delState(myAssetId);
    }
}
```

Os métodos recebem sempre argumento tipo *Context*, uma *String* que representa um identificador, além dos métodos de criar e atualizar que possui um argumento String que representa o valor o objeto da transação.

O contexto possui duas funções. A primeira, permite que um desenvolvedor defina e mantenha variáveis de usuários através das invocações de transação dentro de uma *smart contract*. A segunda função permite o acesso a uma ampla variedade Fabric APIs que permitem um desenvolvedor realizar operações relacionadas ao processamento detalhado transações (HYPERLEDGER, 2020).

Os *smarts contract* sempre sua descrição por padrão com o seguinte design: "*nome-blockchain@vesão_do_blockchain*". O nome é livre escolha do programador, já a versão de forma inicial será "0.0.1". Nos casos de atualização de um blockchain já instanciado o

programador irá gerar um novo bloco mantendo-se o nome, apenas alterando a versão, no caso de “0.0.1” passa a ser “0.0.2” por exemplo.

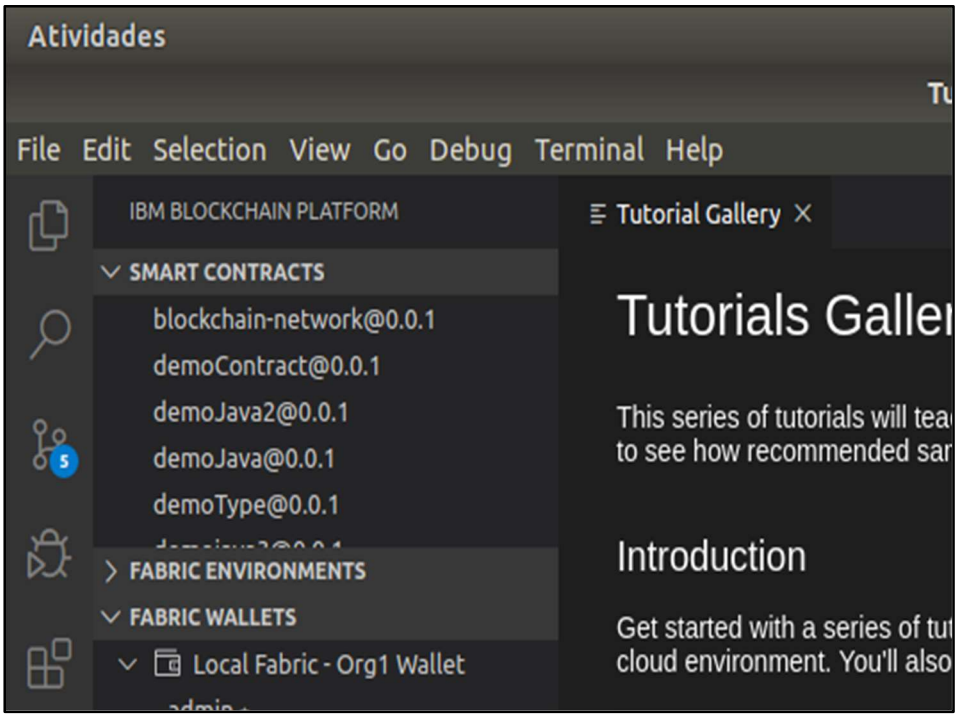

Figura 7: Visão da seção Smart Contract

4.3.1. Local Fabric Enviroments

O *Local Fabric Enviroments* contém uma instancia pré-configurada do Hyperledger Fabric, a qual extrairá e usará automaticamente as imagens corretas do Docker. É uma rede pré-configurada que possui uma organização e um ponto de canal.

Essa seção possui as funções de instalar, instanciar registrar e registar identidades. Para instalar ou instanciar é necessário um arquivo JSON que irá descrever como se conectar ao nó Hyperledger Fabric.

4.3.1. Fabric Gateways

O Fabric Gateway é a ferramenta responsável por executar as funcionalidades do contrato inteligente, ou seja, após a instanciação dos contratos, as funções como criar transação,

consultar uma transação, ou ainda quaisquer outras função atribuídas ao contrato ficaram disponíveis nesse painel para testes, sendo os resultados exibidos no Terminal.

Essa ferramenta permite ao programador realizar uma análise dos resultados obtidos com o contrato inteligente antes de começar a realizar as transações em um servidor em nuvem evitando assim gasto com processamentos desnecessários durante a fase de testes da aplicação.

Na figura 8 temos um blockchain com uma função *instantiate* que irá instanciar o *smart contract* em uma nuvem. Uma função *AddTrader*, que irá adicionar uma identificação de um Trader ou comerciante na blockchain. Em seguida a função *AddCommodity*, que irá adicionar o objeto da transação, que pode ser um algo tangível ou intangível. A próxima função é a *commodityTrade*, que a função que executa a transação. Por fim, temos a função *GetState*, que retorna o estado do objeto, ou seja, sua descrição, e quem possui a propriedade atualmente desse objeto.

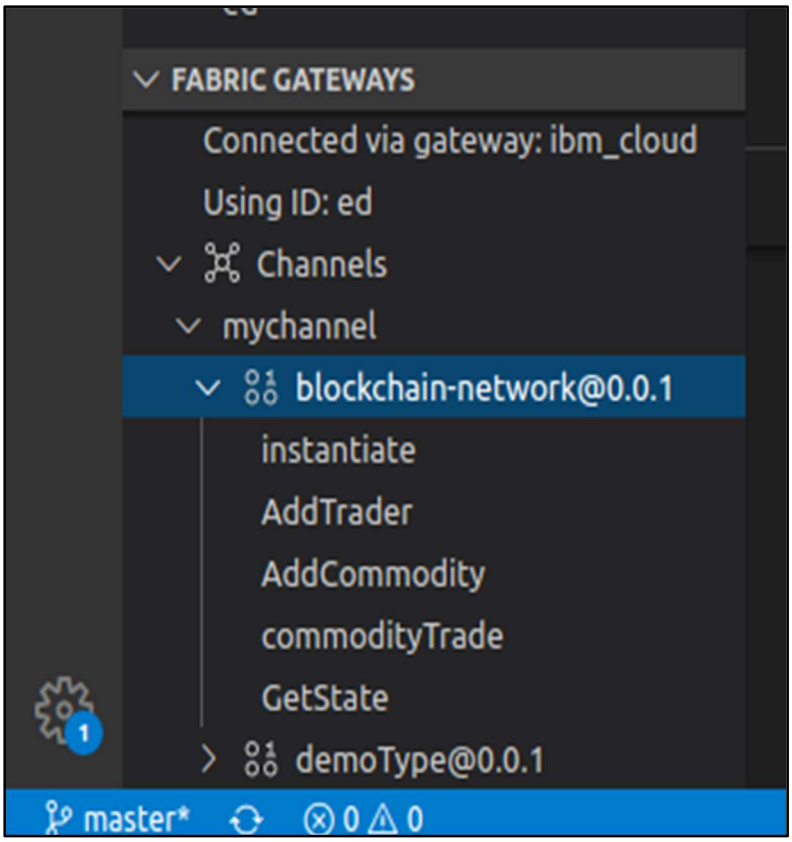

Figura 8: Visão da seção Fabric Gateways

4.3.2. Fabric Wallets

O *Fabric Wallets* contém as carteiras com as credenciais dos participantes da rede, como uma carteira da ou das Organizações que participam da rede, as *Orderers Wallet*, que são responsáveis por receber e empacotar as transações, bem como uma *Peers Wallet*, que fazem execução os *smarts contract,* no exemplo abaixo é identificado como *ibm_cloud_wallet.*

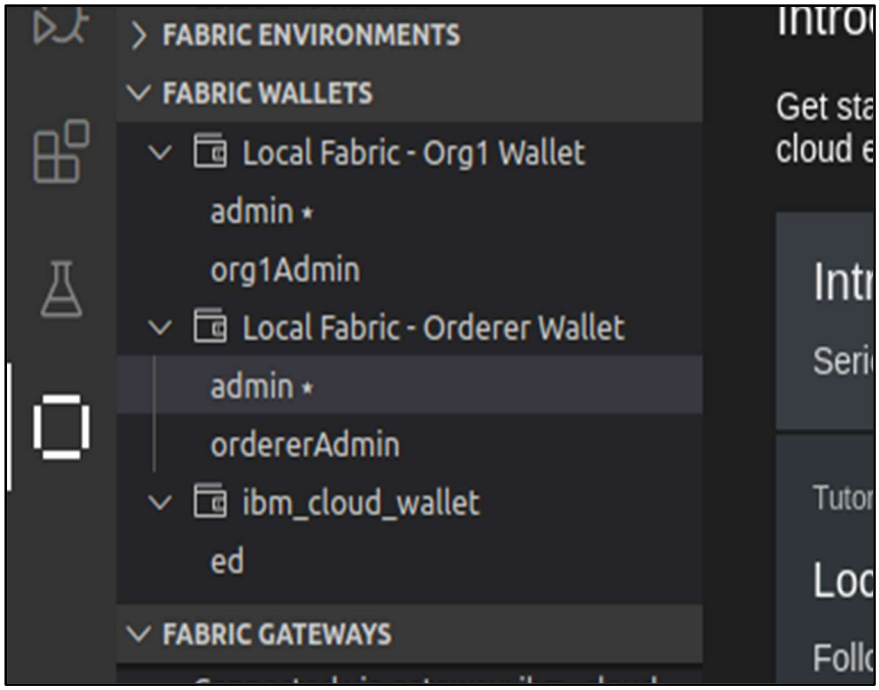

Figura 9: Visão da seção Fabric Wallets

4.4. IBM BLOCKCHAIN PLATFORM

A IBM Blockchain Platform fornece uma oferta BaaS (blockchain como serviço) de pilha gerenciada e completa que permite implementar componentes de blockchain em ambientes diversos. Pode ser implementada tanto no IBM Cloud bem como nas próprias instalações ou ainda em qualquer plataforma de contêiner Kubernetes (Introdução à IBM Blockchain Platform). No presente trabalho a implementação será realizada na própria IBM Cloud onde será armazenado os blocos e transações gerados pelo projeto.

A interface do IBM Blockchain Platform é algo bem intuitivo, onde o desenvolvedor pode der uma visão mais geral de rede blockchain, com os *Peers* e a descrição da organização

a qual eles pertencem, as Autoridades Certificadoras e os *Ordering Service,* existente na rede.

Através do *Dashboard* é possível navegar e obter informações mais detalhadas da rede, como os canais, *smart contract*, *wallet* e *organization* existentes na rede conforme demonstra a figura 10.

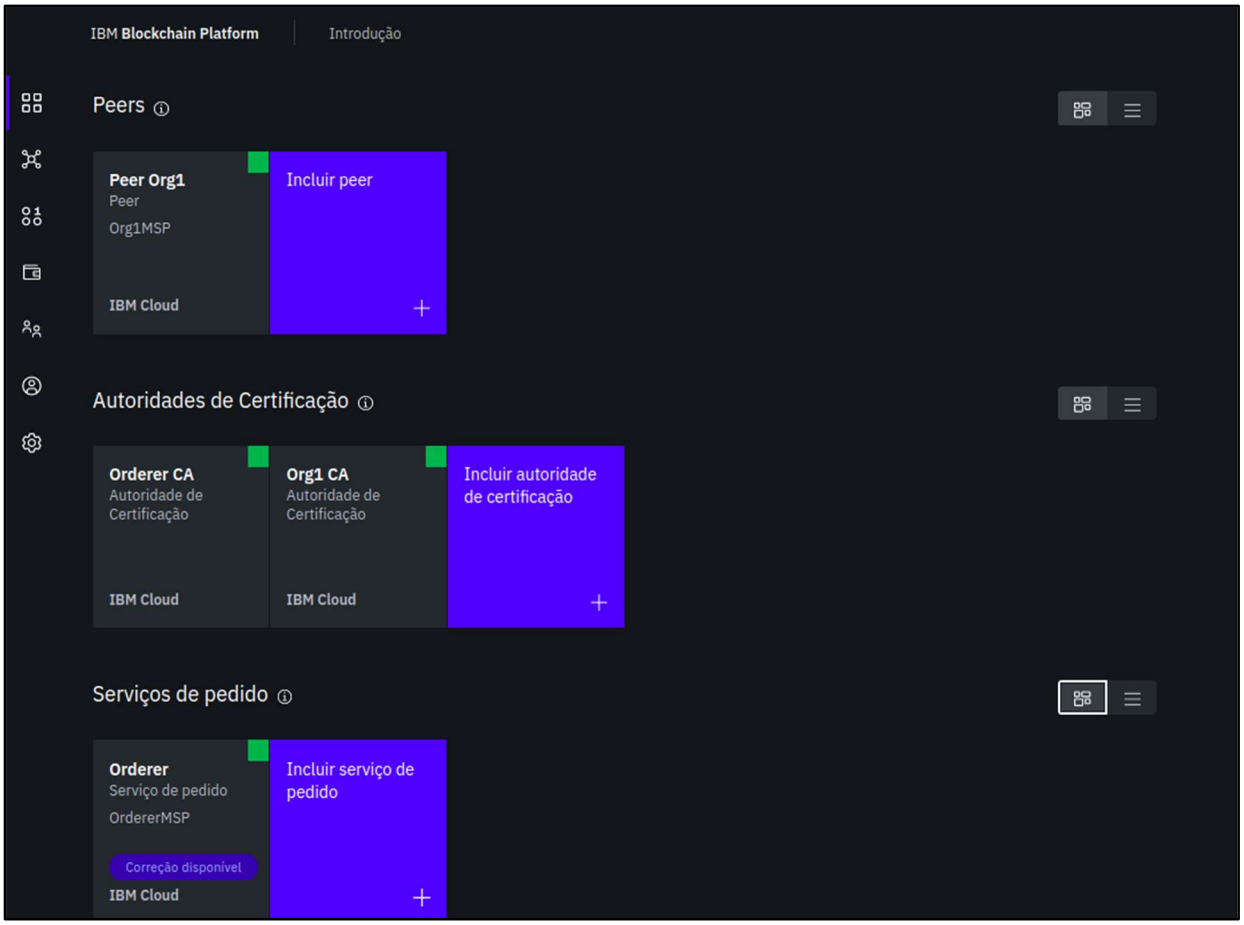

Figura 10: Visão Geral do IBM Blockchain Platform

Na opção canais é possível consultar os canais existentes além dos blocos criados por meio desde canal (figura 11). Além disso a plataforma oferece uma prática opção para criar um canal ou se conectar a canal pré-existente.

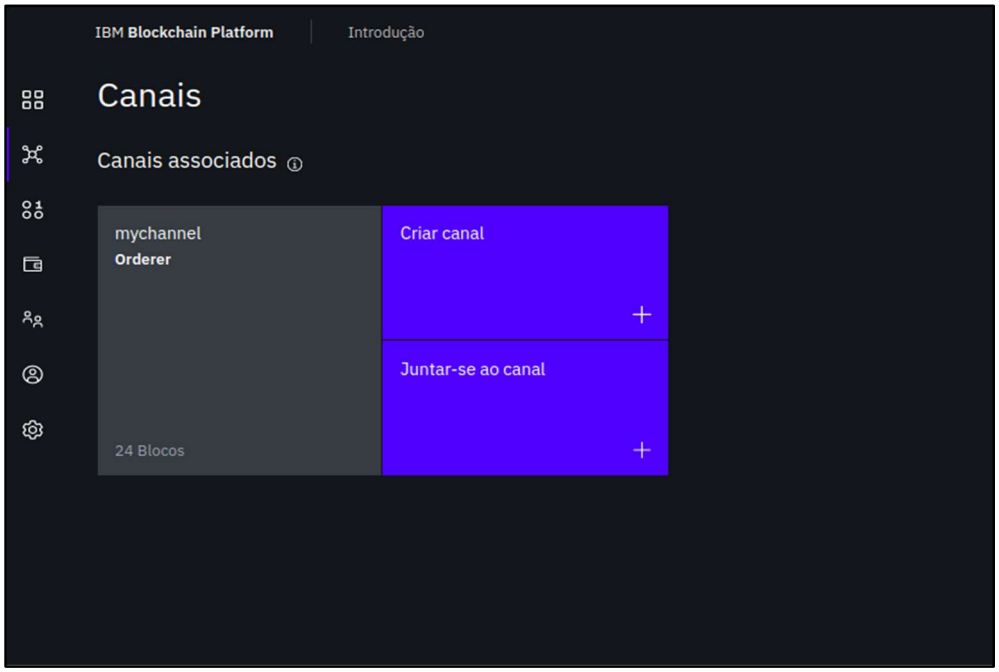

Figura 11: Visão dos canais associados

Ao clicar sobre o canal existe o administrador terá uma visão de várias informações do canal, nome do Serviço de Pedido, versão do recurso do aplicativo, do serviço de pedido e do canal, além das informações sobre os blocos já criados (figura12).

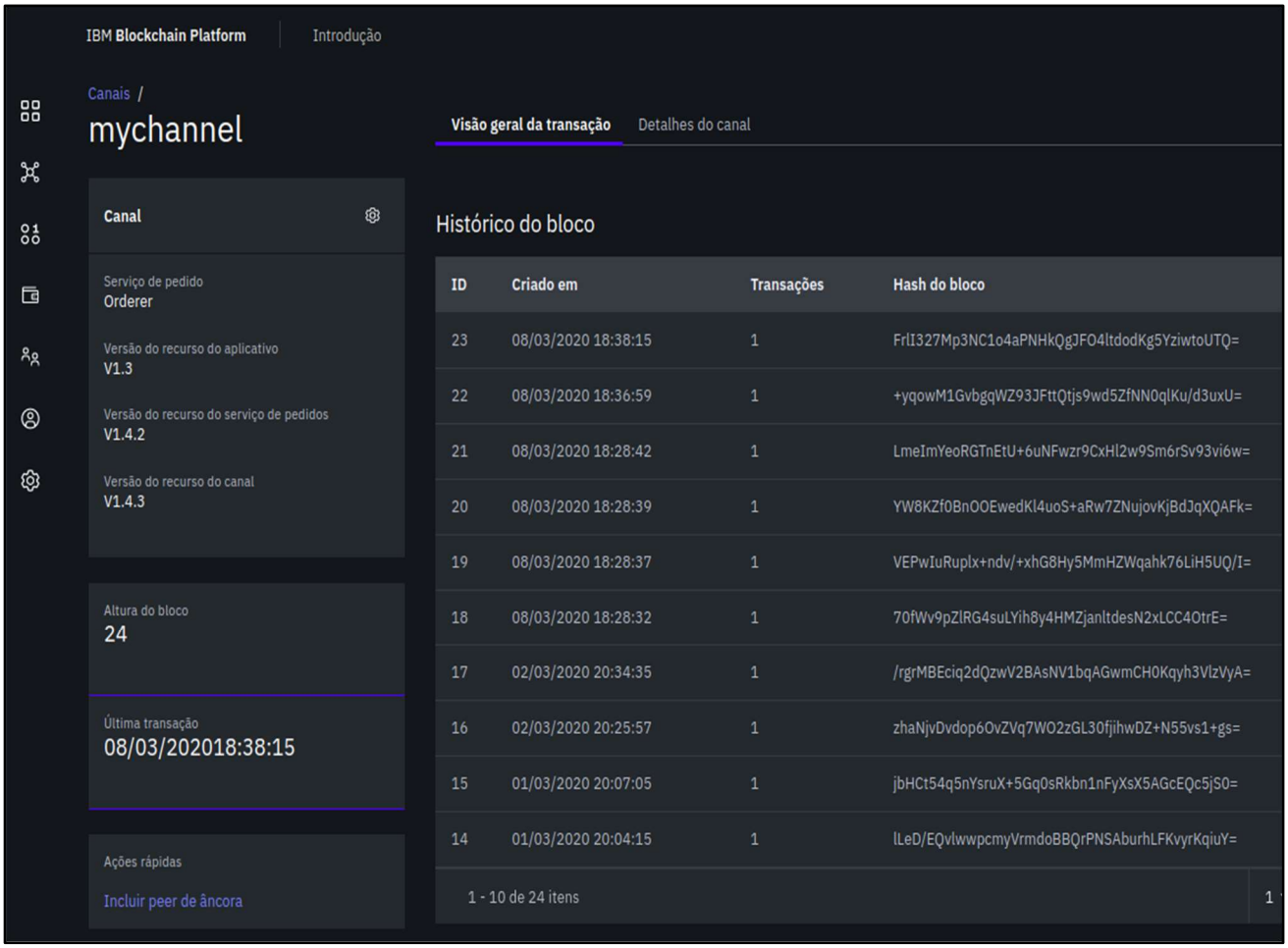

Figura 12: Visão detalhada de um canal

Na opção *smart contract* o administrador tem um a visão dos contratos instalados e instanciados na rede, bem como as opções de instalar e instanciar um contrato (figura 13). Uma outra opção é a de gerar um SDK o qual será utilizado para conectar a aplicação cliente com a rede blockchain.

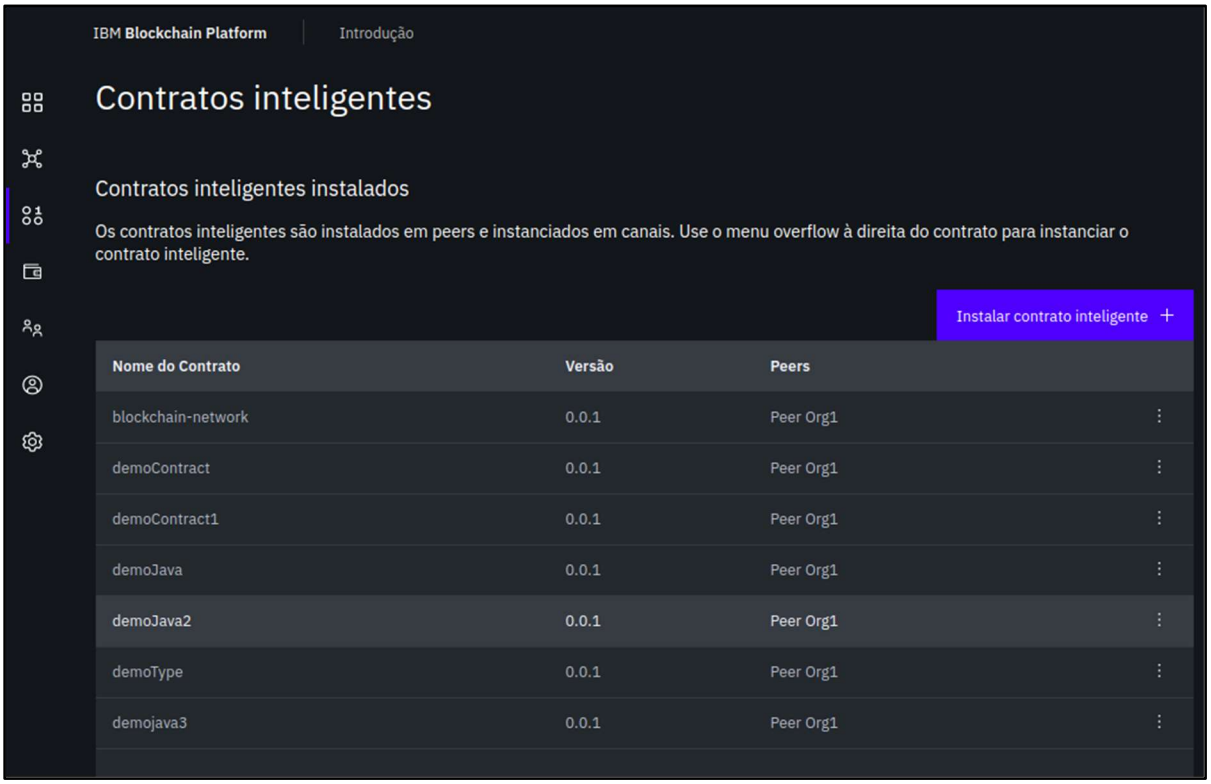

Figura 13: Visão dos contratos instalados na rede

Na Opção *wallet* (figura 14) é possível ter uma visão geral das identidades contidas na rede. Como Administrador, Autoridade Certificadora, Organizações. Nesta tela também é possível criar identidades que irão compor a estrutura da rede.

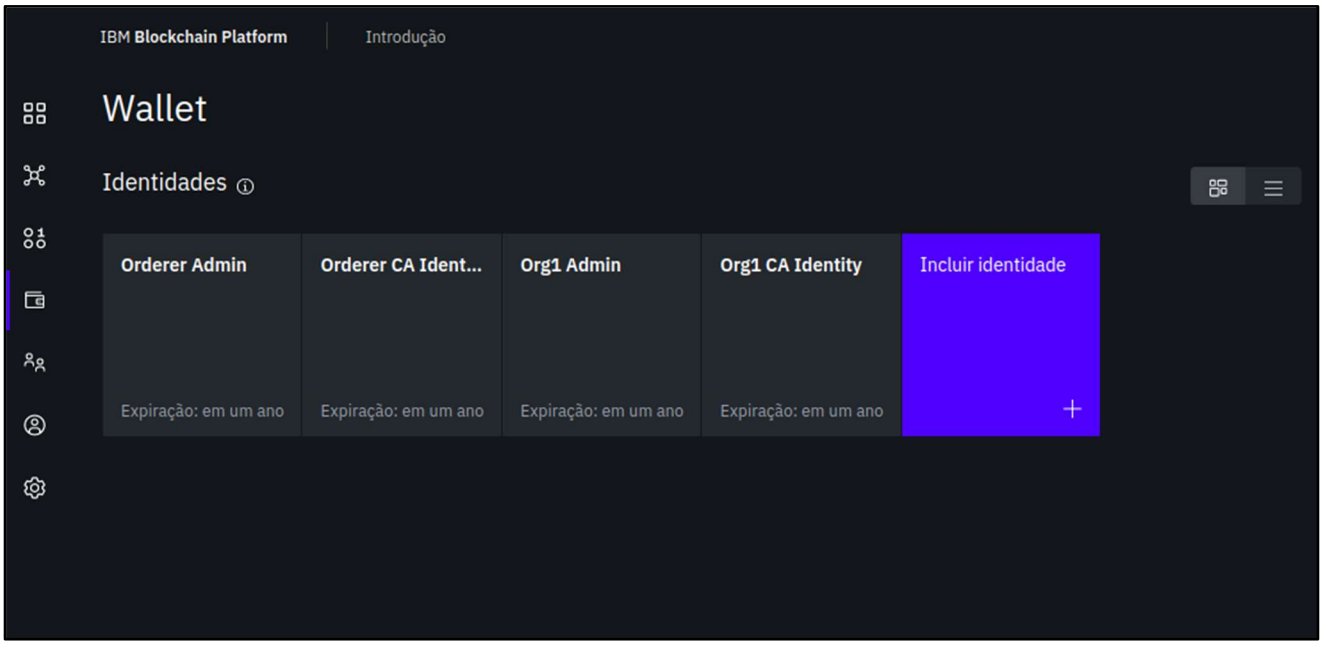

Figura 14: Visão de uma wallet e suas identidades

Por fim temos a opção Organizações (figura 15), que é onde será possível visualizar todas as organizações contidas na rede, e obter informações mais detalhadas sobre essas organizações. Esta opção também permite criar uma definição de MSP ou importar uma já existente.

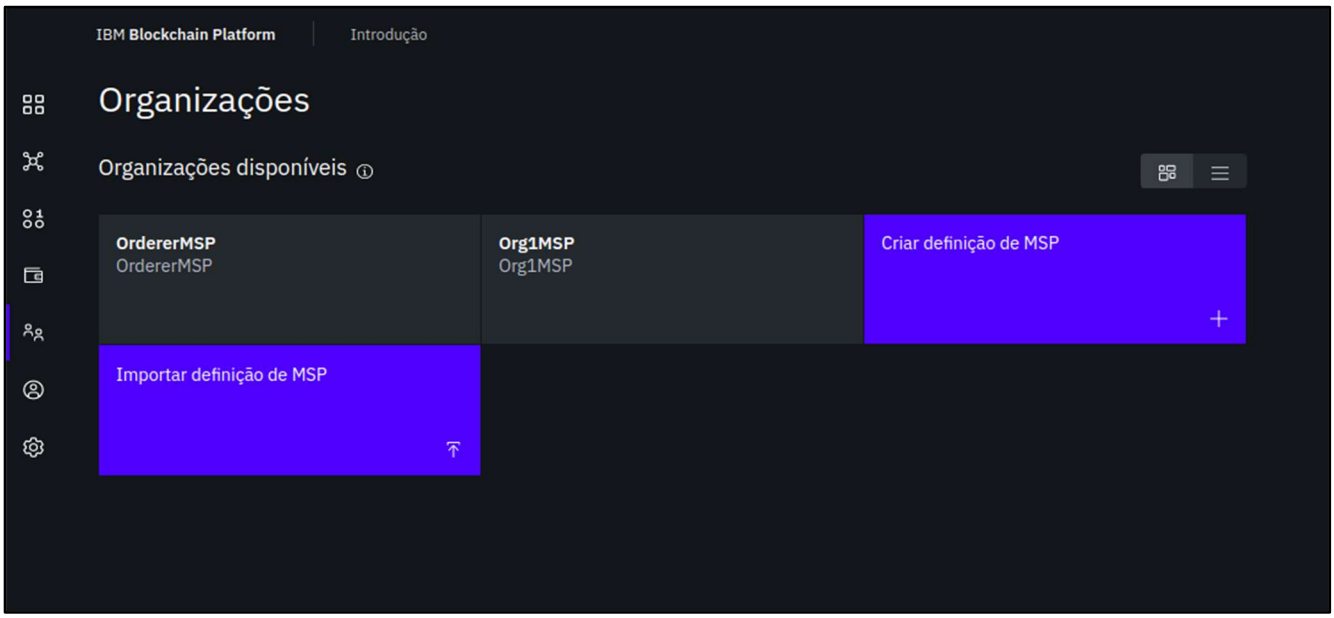

Figura 15: Visão das organizações contidas na rede

5. DIAGRAMA E DESCRIÇÃO DO TRABALHO

Para análise do sistema, que envolve o levantamento e validação de requisitos, além da elaboração de diagramas e para análise de documentação do projeto será utilizado o método de Orientação a Objetos e a modelagem de dados da linguagem UML (*Unified Modeling Language*). Como ferramenta será utilizado a plataforma online disponível no site www.draw.io.

5.1. DESCRIÇÃO DO PROJETO

O presente trabalho tem por objeto desenvolver uma aplicação que demonstrará um software que termine aos usuários criarem e assinarem documentos de forma eletrônica, utilizando como base uma tecnologia blockchain.

Em um primeiro momento será criada uma identidade digital do usuário, que através desta identidade irá se identificar e realizar a assinatura dos documentos. Essa identidade ficar armazenada em uma rede *blockchain.* Posteriormente, quando o usuário já possuir uma identificação, ele poderá criar contratos, enviá-los para outras partes participantes do contrato, para que todos possam assinar os contratos. Os contratos conforme forem sendo assinados, irão receber uma transação de atualização na rede blockchain, informando que um participante assinou o contrato. Parem que alguém possa assinar esse contrato, sua identificação deverá constar dentro do bloco de criação do contrato, ou seja quando o contrato é criado, o usuário criador deverá informar a identificação dos usuários participantes.

5.2. DIAGRAMA DE CASO DE USO

Os casos de uso são uma técnica utilizada para captar os requisitos funcionais de um sistema. Eles descrevem as interações típicas entre os usuários de um sistema e o próprio sistema, fornecendo uma narrativa sobre como o sistema é utilizado (FOWLER, 2004).

Casos de uso possui um ator principal, que solicita ao sistema para que execute um serviço. Esse ator principal é aquele cuja o objetivo o caso de uso está tentando satisfazer e, na maioria das vezes será o iniciador do caso de uso.

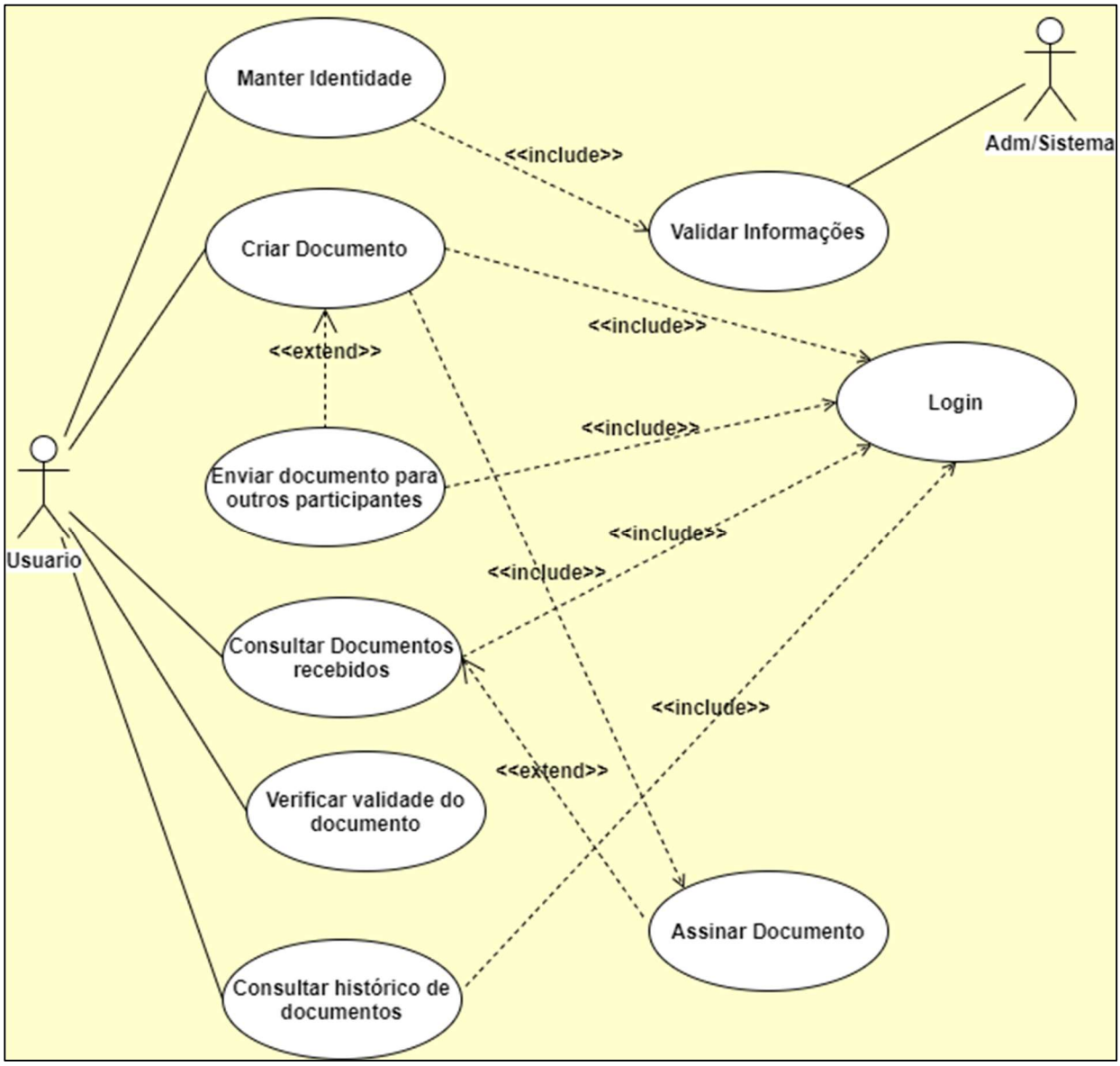

Figura 16: Diagrama de Caso de Uso

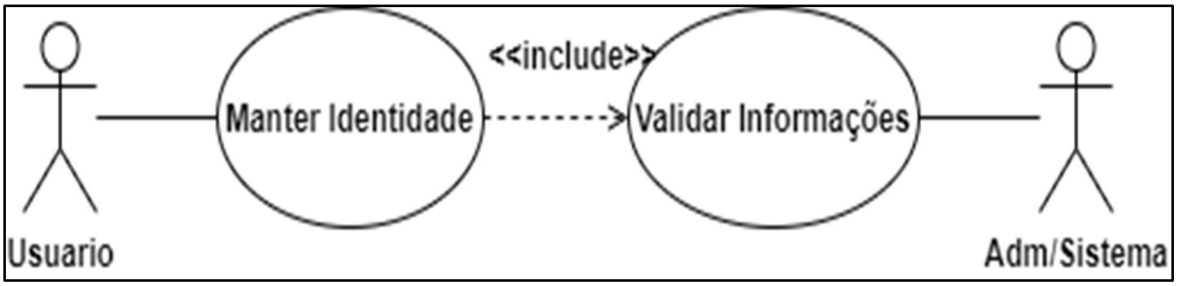

Figura 17: Diagrama de Caso de Uso Manter Identidade

Finalidade/Objetivo	Criar uma identidade digital para o usuário que realizou a atividade
Atores	Usuário e administrador/Sistema
Pré-condição	Não Existe
Evento inicial	O Usuário clica não opção “criar cadastro” na tela inicial
Evento principal	• O aplicativo oferece ao usuário uma interface onde ele consegue inserir seus dados pessoais. • O usuário preenche o formulário disponível. • Os dados serão enviados ao administrador ou a um sistema que irá verificar a veracidade dos dados informados [A1]. • Validado os dados a identidade é criada
Fluxo alternativo	A1 – Os dados informados são inconsistentes:

	• O usuário será informado que a identidade não pode ser criada pois os dados são inconsistentes. • O sistema retorna ao a tela inicial.

Tabela 1: Manter Identidade

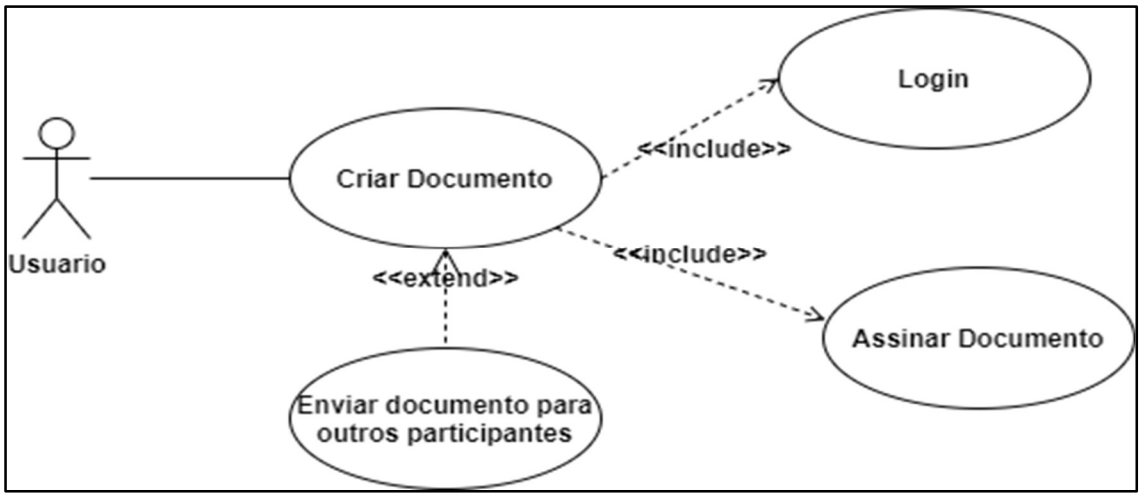

Figura 18: Caso de Uso Criar Documento

Finalidade/Objetivo	Criar um documento assinado digitalmente para o usuário que realizou a atividade
Atores	Usuário
Pré-condição	Login
Evento inicial	O Usuário clica não opção “criar documento” no menu principal
Evento principal	• O aplicativo oferece ao usuário uma interface onde ele consegue redigir o documento.

	• O usuário redige documento e informa quem deverá assinar o documento [A1]. • Após redigir o documento e informar as partes que irão assinar o usuário é clicar em assinar documento.
Fluxo alternativo	A1 – Existindo outras partes no documento: • Após assinatura do documento um link de acesso ao documento será enviado aos demais participantes para que assinem o documento.

Tabela 2: Criar Documento

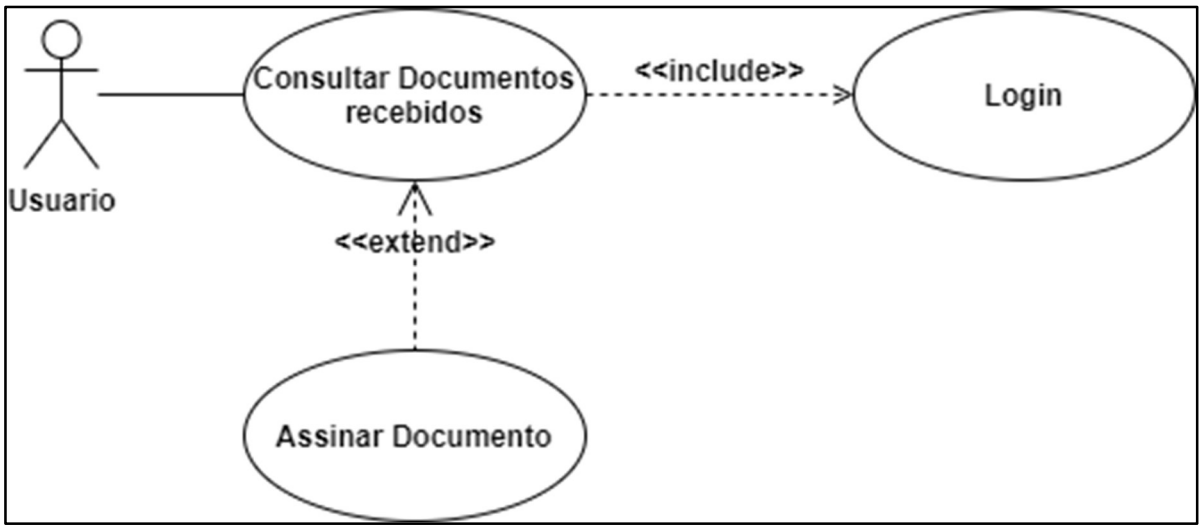

Figura 19: Caso de Uso Consultar Documentos recebidos

Finalidade/Objetivo	Consultar os Documentos recebidos que estão pendentes de assinatura
Atores	Usuário

Pré-condição	Login
Evento inicial	O Usuário clica não opção “Documentos recebidos” no menu principal
Evento principal	• O aplicativo oferece ao usuário uma interface onde ele consegue visualizar todos os documentos recebidos que estão pendentes de assinatura. • Em uma lista de documentos o usuário irá clicar sobre o documento que deseja assinar. • A aplicação irá exibir a integra do documento além das partes que irão ou já assinaram o documento. • O usuário clica em assinar documento [A1] [A2].
Fluxo alternativo	A1 – Recusar documento: • O usuário clica na opção “Recusar”, o documento irá sair da lista pendentes. • O sistema retorna a lista de pendentes. A2 – Cancelar: • O usuário clicar em “cancelar”, a tela com o documento irá fechar. • O documento continua na lista de pendentes • O sistema retorna a lista de pendentes.

Tabela 3: Documento Recebido

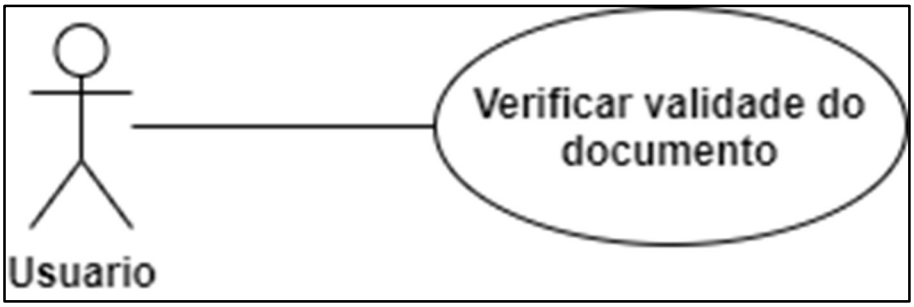

Figura 20: Caso de Uso Verificar validade do documento

Finalidade/Objetivo	Verificar a validade do documento em posse do usuário
Atores	Usuário
Pré-condição	Possuir chave de identificação do documento
Evento inicial	O Usuário clica não opção “Consultar Documento” no menu principal
Evento principal	• O aplicativo oferece ao usuário uma interface onde ele insere a chave de identificação do documento. • O usuário insere a chave de identificação [A1]. • O sistema retorna o documento da chave informada.
Fluxo alternativo	A1 – chave de identificação invalida: • O sistema retorna que a chave informada não possui um documento correspondente. • O sistema retorna ao menu principal.

Tabela 4: Verificar validade de documento

Figura 21: Caso de uso Consultar Histórico

Finalidade/Objetivo	Consultar os Documentos que já foram criados e ou assinados pelo usuário
Atores	Usuário
Pré-condição	Login
Evento inicial	O Usuário clica não opção “Histórico” no menu principal
Evento principal	• O aplicativo oferece ao usuário uma interface onde ele consegue visualizar todos os documentos criados ou assinados pelo usuário. • Em uma lista de documentos o usuário irá clicar sobre o documento que deseja consultar. • A aplicação irá exibir a integra do documento além das partes que irão ou já assinaram o documento [A1].
Fluxo alternativo	A1 – imprimir ou realizar download do documento:

	• Ao visualizar o usuário poderá clicar em imprimir ou realizar download do documento. A2 – Cancelar: • O usuário clicar em “cancelar”, a tela com o documento irá fechar. • O sistema retorna a lista de histórico.

Tabela 5: Consultar Histórico

5.3. DIAGRAMA DE CLASSE

Diagramas de classe descrevem os tipos de objetos presentes em um sistema e os vários tipos de relacionamentos estáticos existentes entre eles. Esses tipos de diagramas também mostram as propriedades e as operações de uma classe e restrições que se aplicam a forma de como os objetos estão conectados (FOWLER, 2004).

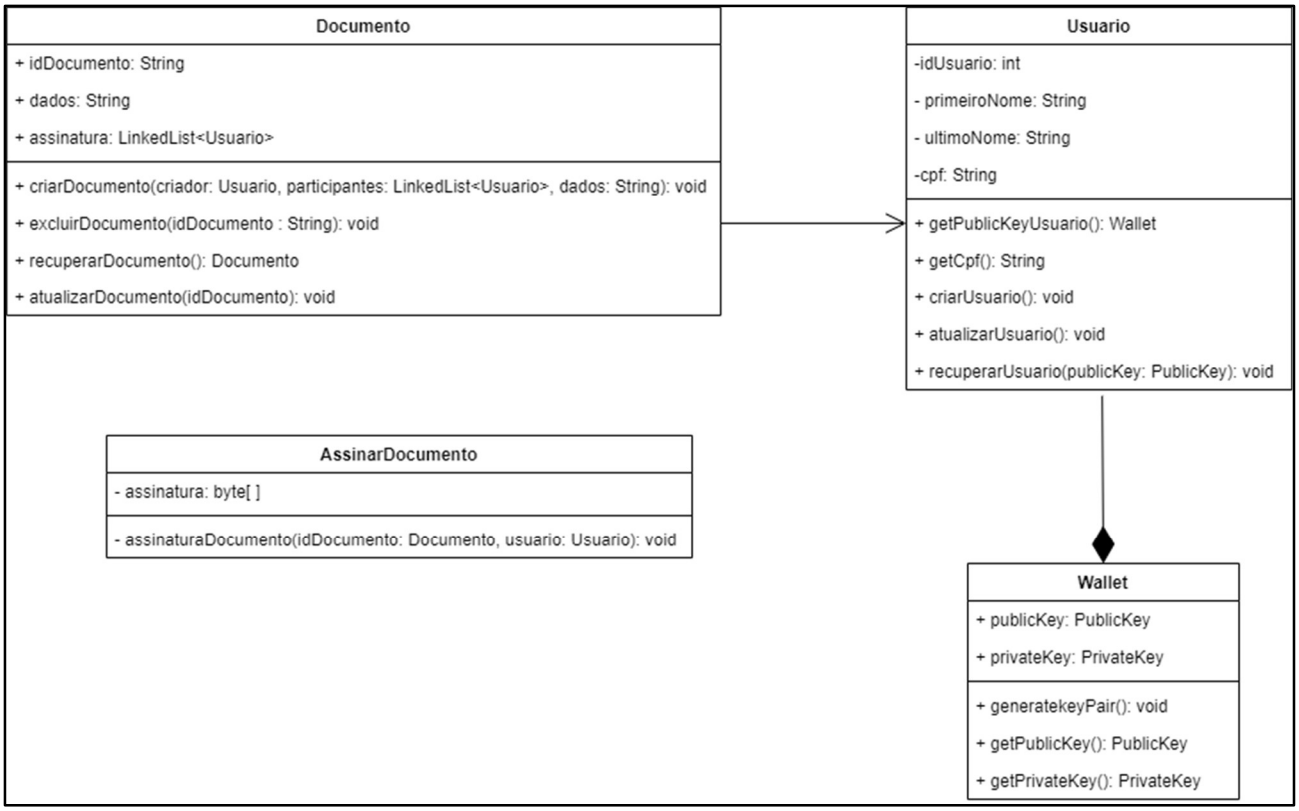

Figura 22: Diagrama de Classe

5.4. DIAGRAMA DE ATIVIDADES

Os diagramas de atividades são uma técnica para descrever logica de procedimento, processo de negócio e fluxo de trabalho. Esse diagrama permite que quem está seguindo o processo escolha a ordem na qual fazer as coisas. Em outras palavras, ele simplesmente determina a s regras essenciais de sequência que se deve seguir (FOWLER, 2004).

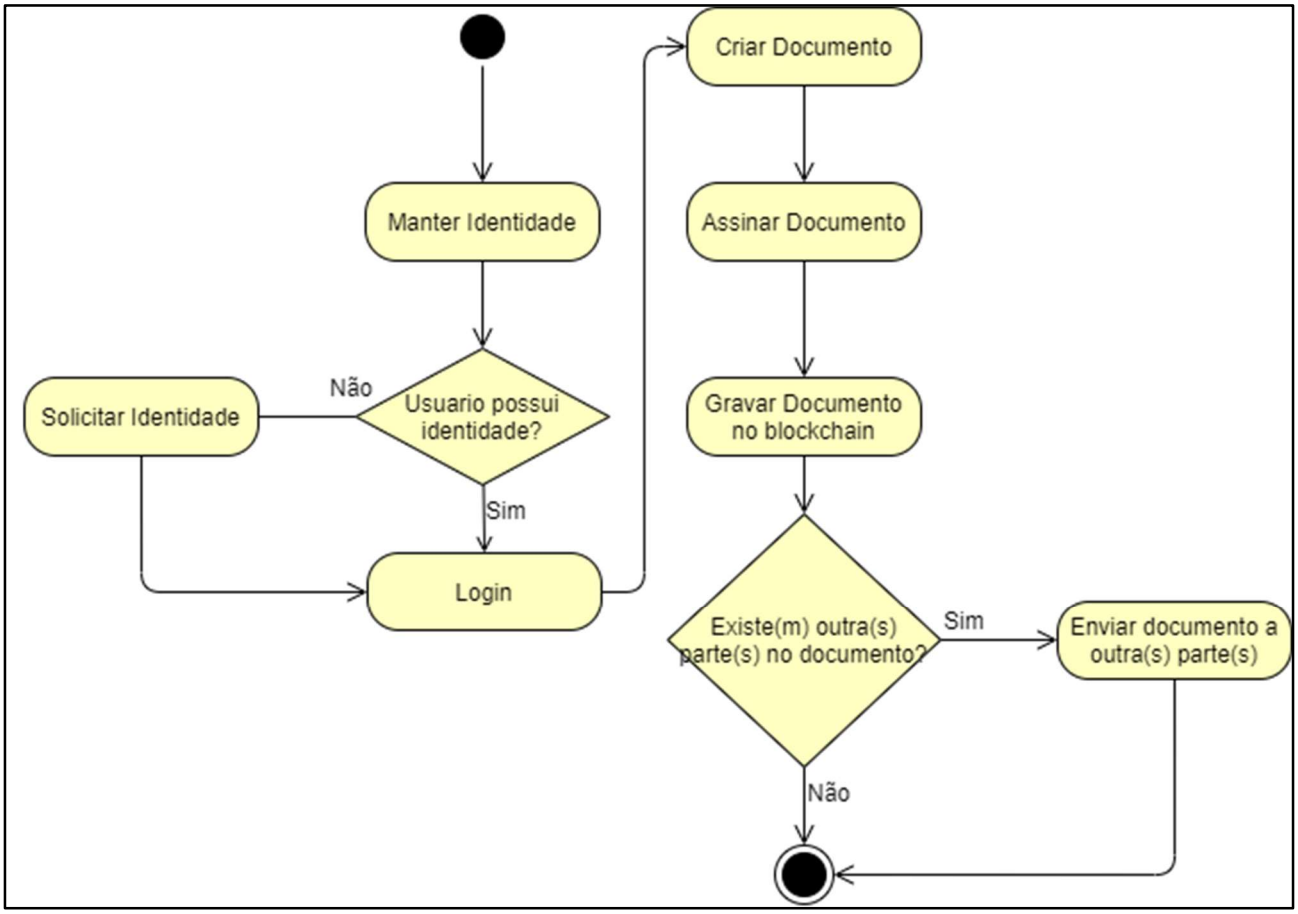

Figura 23: Diagrama de Atividade Criar Documento

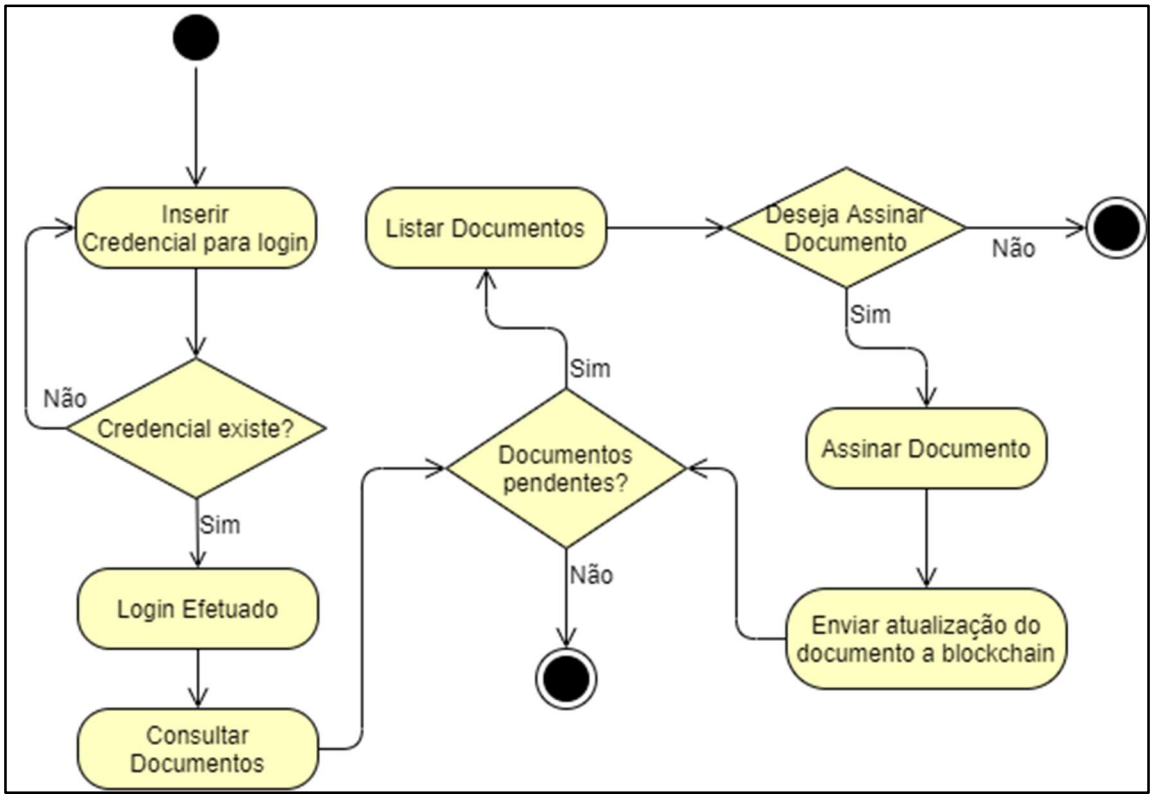

Figura 24: Diagrama de Atividade Assinar documento pendente

5.5. DIAGRAMA DE SEQUÊNCIA

Enquanto casos de uso descrevem como atores externos interagem com um sistema, gerando um evento de sistema pra um sistema, geralmente solicitando uma operação de sistema para tratar um evento. Um diagrama de sequência de sistema é uma figura que mostra, para um cenário específico de caso de uso, os eventos que os atores externos geram, sua ordem e os eventos entre sistemas (LARMAN, 2004).

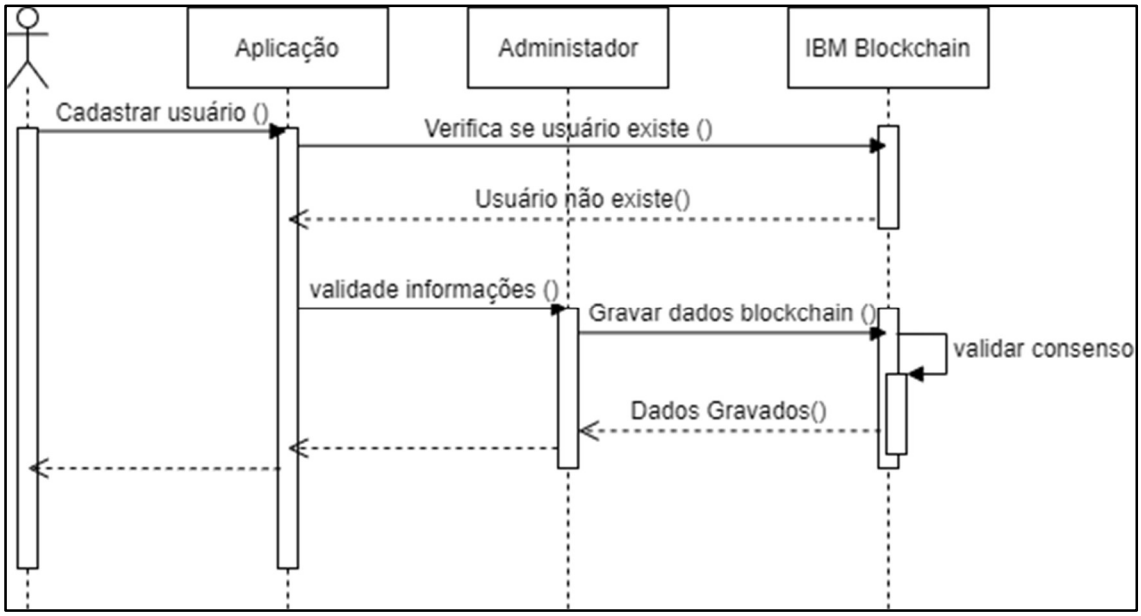

Figura 25: Diagrama de Sequência Criar Identidade

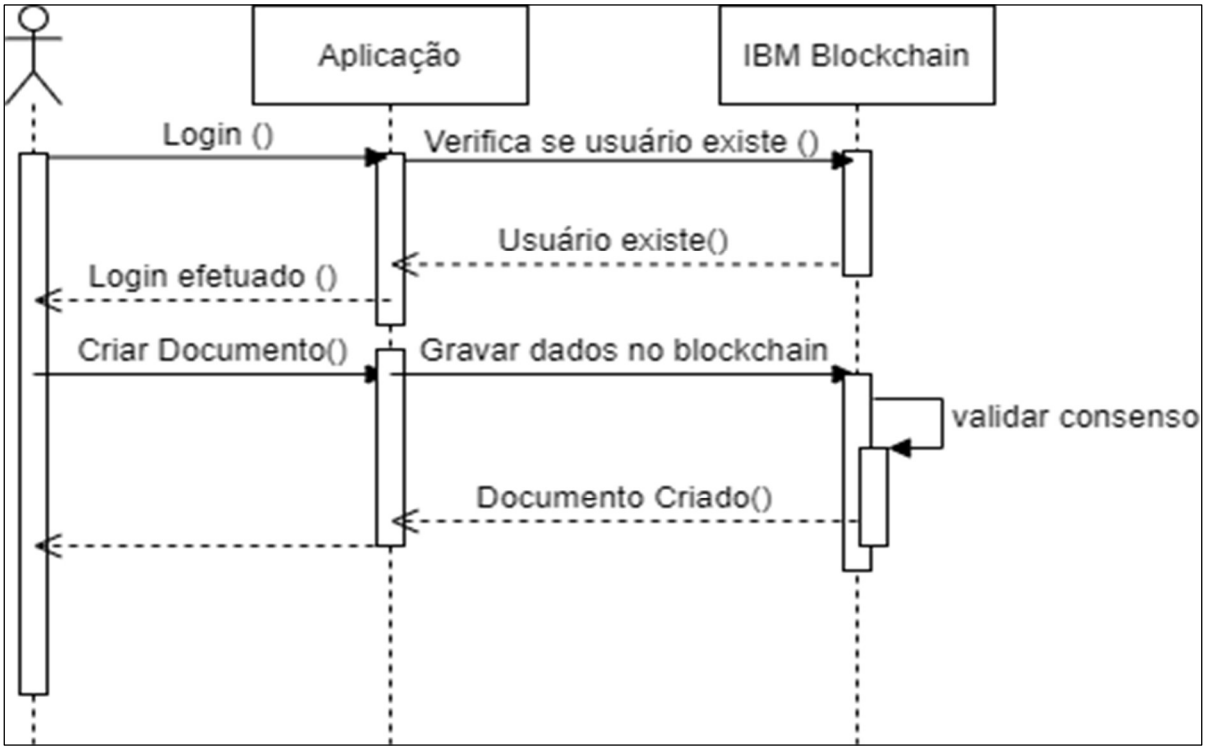

Figura 26: Diagrama de Sequência Criar Documento

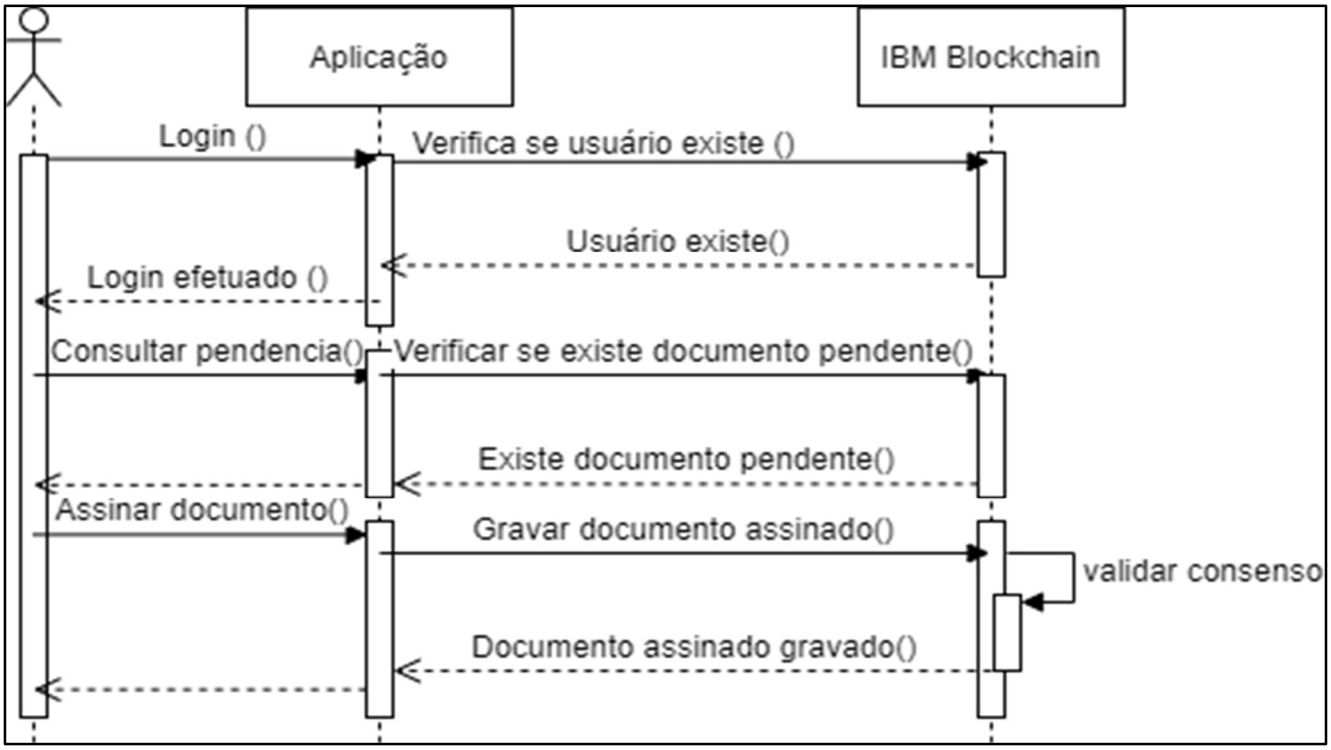

Figura 27: Diagrama de Sequência Assinar Documento Pendente

6. CONCLUSÃO

Ao final do estudo proposto será possível demonstrar de forma clara as vantagens da utilização da tecnologia blockchain na criação de documentos digitais. Permitindo ao usuário criar, assinar, e validar documentos assinados digitalmente.

Com isso, espera-se que esse estudo ajude a criar novas formas de utilização da tecnologia blockchain em nosso cotidiano, permitindo que todas as pessoas, tenham a sua disposição uma ferramenta poderá ser de grande utilidade em suas vidas, tornando mais pratica e barata a relação contratual entre duas ou mais partes, independentes de onde estejam.

6.1. TRABALHOS FUTUROS

Como trabalhos futuros, pode-se considerar uma implementação de uma identidade digital também utilizando uma rede blockchain de forma a dar uma maior segurança não somente nos dados referentes aos documentos criados e assinados, mas também sim na identificação do usuário que está criando esse documento.

Além da identidade digital, há a possibilidade de implementação de sua segunda camada de privacidade além dos *channels* através um recurso do próprio framework do *Hyperledger Fabric* chamado Data Private. Tal recurso que funciona como um sub canal que é implementado como um atributo *array* do objeto que está sendo transacionado na rede, o qual leva a identificação dos membros que pode acessar tal transação.

REFERÊNCIAS

AFONSO, A. **Produtividade no desenvolvimento de Aplicações WEB com Spring Boot**. 3º ed. [s.l.] AlgaWorks Softwares, Treinamentos e Serviços Ltda, 2017.

AITZHAN, N. Z.; SVETINOVIC, D. Security and Privacy in Decentralized Energy Trading Through Multi-Signatures, Blockchain and Anonymous Messaging Streams. **IEEE Transactions on Dependable and Secure Computing**, v. 15, n. 5, p. 840–852, 2018.

BEHRENS, F. **A Assinatura Eletrônica como Requisito de Validade dos Negócios Jurídicos e a Inclusão Digital na Sociedade Brasileira**. [s.l.] ontifícia Universidade Católica do Paraná, 2005.

BRASIL. MEDIDA PROVISÓRIA nº 2.200-2,. . 2001.

BRITO, E. **Java: Entenda para que serve o software e os problemas da sua ausência**. Disponível em: <https://www.techtudo.com.br/dicas-e-tutoriais/noticia/2014/11/java-entenda-para-que-serve-o-software-e-os-problemas-da-sua-ausencia.html>. Acesso em: 8 ago. 2020.

EDUARDO MAEHARA ALIAGA, Y.; AURELIO AMARAL HENRIQUES, M. **Uma comparação de mecanismos de consenso em blockchains**. Campinas: 2017Disponível em: <https://www.fee.unicamp.br/sites/default/files/departamentos/dca/eadca/eadcax/trabalhos/artigo_20_Mecanismos_Consenso_Blockchains_Yoshitomi_Maehara_Prof_Marco_Aurelio.pdf>. Acesso em: 6 ago. 2020

FINANCE, T. I. OF I. **Banking on the Blockchain Reengineering the Financial Architecture**. [s.l: s.n.].

FOWLER, M. **UML ESSENCIAL: UM BREVE GUIA PARA A LINGUAGEM-PADRAO**. [s.l: s.n.].

GREVE, F. et al. **Blockchain e a Revolução do Consenso sob DemandaSimpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC) - Minicursos**. [s.l: s.n.]. Disponível em: <http://143.54.25.88/index.php/sbrcminicursos/article/view/1770>. Acesso em: 8 ago. 2020.

GUEGAN, D. **Public Blockchain versus Private blockhain**. [s.l: s.n.]. Disponível em: <http://centredeconomiesorbonne.univ-paris1.fr/>. Acesso em: 8 ago. 2020.

GUELFI, A. R. **Análise de elementos jurídico-tecnológicos que compõem a assinatura digital certificada digitalmente pela Infra-estrutura da Chaves Públicas do Brasil (ICP-Brasil).**São PauloUniversidade de São Paulo, , 2011. Disponível em: <https://www.teses.usp.br/teses/disponiveis/3/3142/tde-26072007-164132/en.php>

GUPTA, S.; SADOGHI, M. Blockchain Transaction Processing. In: **Encyclopedia of Big Data Technologies**. [s.l.] Springer International Publishing, 2018. p. 1–11.

HYPERLEDGER. **hyperledger-fabricdocs Documentation Release master hyperledger**. [s.l: s.n.]. Disponível em: <https://hyperledger-fabric.readthedocs.io/_/downloads/en/latest/pdf/>. Acesso em: 9 ago. 2020.

IBM BLOCKCHAIN. **IBM Blockchain Platform - Visual Studio Marketplace**. Disponível em: <https://marketplace.visualstudio.com/items?itemName=IBMBlockchain.ibm-blockchain-platform>. Acesso em: 8 ago. 2020.

IDENTITY THEFT RESOURCE CENTER. **DATA BREACH REPORTS**. [s.l: s.n.]. Disponível em: <https://www.idtheftcenter.org/images/breach/2017Breaches/DataBreachReport_2017.pdf>. Acesso em: 6 ago. 2020.

JUNIOR, I. T. G. **O Documento Eletrônico como Meio de Prova no Brasil**São PauloSaraiva, , 2001.

LARMAN, C. **Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development (3rd Edition)**. USA: Prentice Hall PTR, 2004.

MAHESHWARI, S. **Blockchain basics: Hyperledger Fabric – Build Smart. Build Secure. IBM Developer**. Disponível em: <https://developer.ibm.com/articles/blockchain-basics-hyperledger-fabric/>. Acesso em: 6 ago. 2020.

MENKE, F. **No Title**ed. São Paulo: Pontifícia Universidade Católica de São Paulo, , 2017. (Nota técnica).

MICROSOFT. **Documentation for Visual Studio Code**. Disponível em: <https://code.visualstudio.com/docs>. Acesso em: 6 ago. 2020.

OLIVEIRA ABREU BATISTA, A.; REBECCA BASTOS DIAS, E.; BORGES SILVA, M.

Identificação digital baseada em blockchain: Um conceito disruptivo no ciberespaço. Goiânia: Media Lab / UFG, 2018Disponível em: <https://files.cercomp.ufg.br/weby/up/777/o/28_-_Alex_Batista.pdf>. Acesso em: 6 ago. 2020

PALANIVEL, C. **Hyperledger Fabric - Architecture, Components & Transactional Flow | LinkedIn**. Disponível em: <https://www.linkedin.com/pulse/hyperledger-fabric-architecture-components-flow-chandrasekaran/>. Acesso em: 8 ago. 2020.

REBELLO, G. A. F. et al. **Correntes de Blocos: Algoritmos de Consenso e Implementação na Plataforma Hyperledger Fabric**. 38o Jornada de Atualização em Informática (JAI) do XXXIX Congresso da Sociedade Brasileira de Computação (CSBC 2019). **Anais**...2019

RIVERA, R. et al. **How digital identity on blockchain can contribute in a smart city environment**. 2017 International Smart Cities Conference (ISC2). **Anais**...2017

SCHILDT, H. **Java para iniciantes**. [s.l.] Bookman Editora, 2015.

TAM KC. **Examining the Behaviour of Hyperledger Fabric when World State is Tampered**. Disponível em: <https://medium.com/@kctheservant/exploring-the-behaviour-of-hyperledger-fabric-when-world-state-is-tampered-764676fe90f2>. Acesso em: 13 ago. 2020.

VITORINO, J. C. **Blockchain: As principais aplicações na área de Redes e Telecom. – Blog CCNA**. Disponível em: <https://blog.ccna.com.br/2018/09/12/blockchain-as-principais-aplicacoes-na-area-de-redes-e-telecom/>. Acesso em: 12 ago. 2020.

WINDER, R.; ROBERTS, G. **Desenvolvendo software em Java**. Rio de Janeiro: Grupo Gen - LTC, 2009.

APENDICE A - INSTALAÇÃO DE HELLO-WORLD HYPERLEDGER A PARTIR DA IBM BLOCKCHAIN PLATFORM

REQUISITOS

Para desenvolver uma hello-world Hyperledger é necessário que o ambiente de desenvolvimento possuía os seguintes requisitos:

- Sistema Operacional Linux ou Windows (verificar a compatibilidade da versão do Windows com Docker)
- NodeJs versão 8 ou 10.
- Npm versão mais recente
- Docker versão não superior a 18.9.9
- Docker Compose versão mais recente
- Java 8.
- Gradle mais recente
- Visual Code com a extensão IBM Blockchain Platform

INSTALAÇÃO IBM BLOCKCHAIN PLATFORM

Para instalar o IBM Blockchain Platform o desenvolvedor deverá buscar em extensões por IBM Blockchain Platform e clicar em *install*.

Após a instalação da extensão o desenvolvedor deverá verificar se todos os requisitos são preenchidos, para isso ao abrir o Visual Studio Code, o desenvolvedor deverá pressionar F1 e buscar por *IBM Blockchain Platform: View Prerequisites* (figura 28). Será mostrado uma tela com os requisitos informando se eles já estão todos instalados. Caso necessite instalar algum, após a instalação basta clicar em *Check again* para verificar de a instalação ocorreu corretamente.

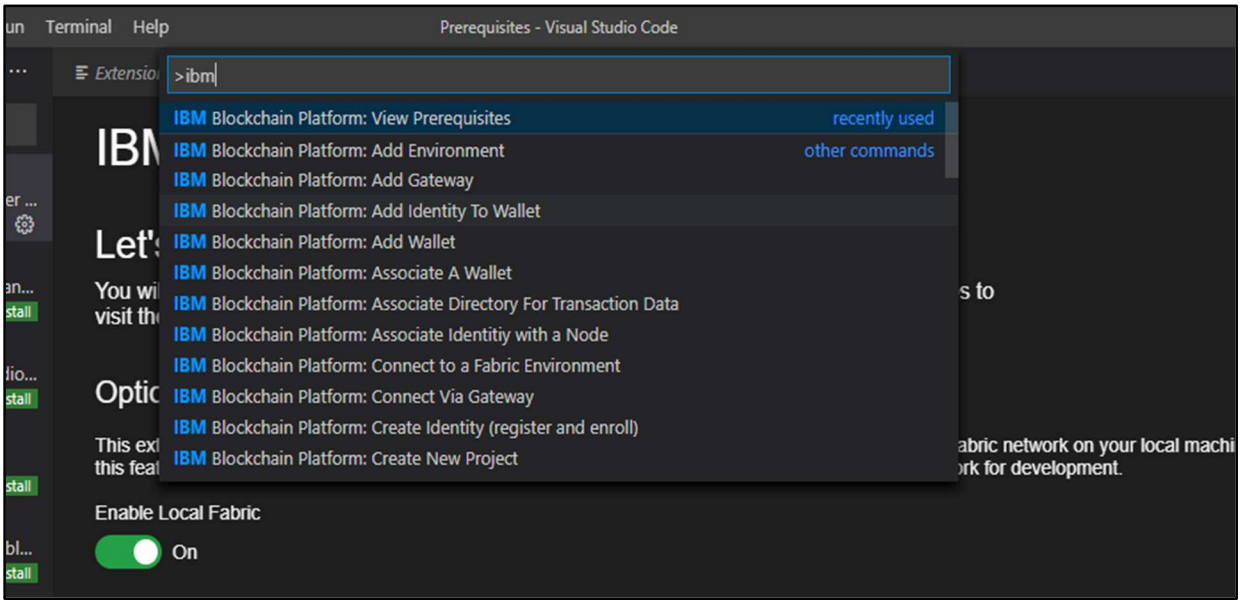

Figura 28: Visão dos pré-requisitos da IBM Blockchain Platform

CRIANDO UM PROJETO

A fim de facilitar o desenvolvimento de um *chaincode*, o plugin IBM Blockchain Platform tem a opção para gerar um projeto com todas as configurações básicas para um chaincode e uma modelo básico de implementação de *chaincode.*

Para criar esse projeto deve clicar no ícone da extensão IBM Blockchain Platform e clicar no menu de ações e escolher a opção Create New Project (figura 29).

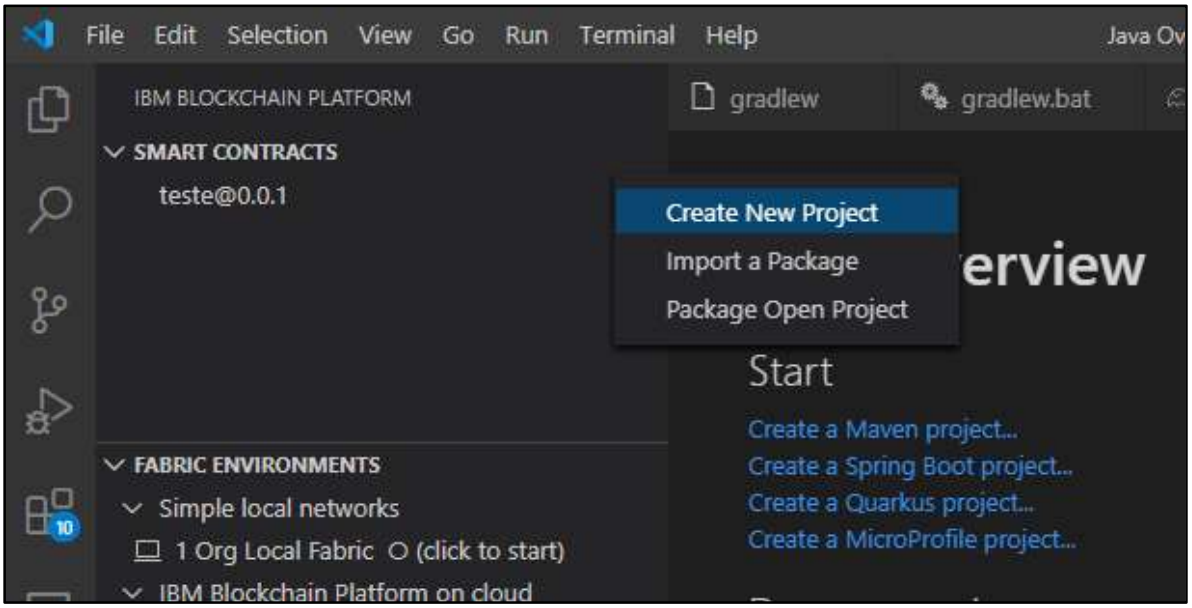

Figura 29: Visão criação de novo projeto

Em seguida escolha o tipo de contrato, Default Contract ou Private Data Contract. Para essa implementação será utilizado o tipo Deault Contract. Após a o tipo será solicitado a escolha da linguagem, sendo as opções Java, JavaScript, TypeScript ou Go. Como o objeto desse trabalho é a implementação de um modelo de blockchain utilizando Java como linguagem, será escolhido a opção Java. Em seguida informar o nome do smart contract.

O projeto possui inicialmente duas classes, a primeira é a *MyAsset* que representa o objeto que será transacionado e possui os métodos *toJSON* e *fromJSON*, que serão utilizados converter a *string* do das recebido para transação em JSON e converter os dados provenientes da rede para *string* (figura 30).

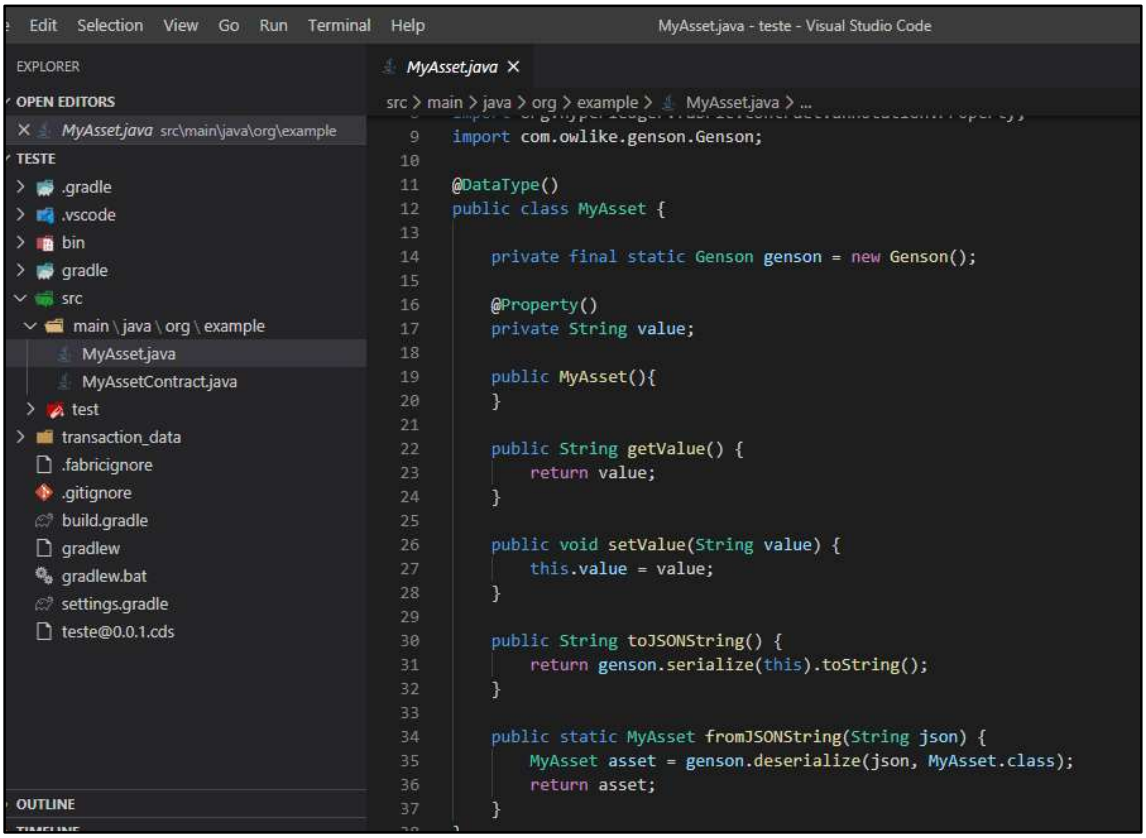

Figura 30: Modelo da classe MyAsset

A segunda é a *MyAssetContract* que possui os métodos *create, read, delete* e *update*. O método *createMyAsset* recebe três argumentos, que são o *Context*, e duas strings, o id e o objeto da transação cada uma delas. Antes de criar a transação ele verifica se aquela transação já existe, em caso negativo a string é transformada em pelo método toJSON da classe MyAsset e transação é confirmada(figura 31).

```
36
37      @Transaction()
38      public void createMyAsset(Context ctx, String myAssetId, String value) {
39          boolean exists = myAssetExists(ctx,myAssetId);
40          if (exists) {
41              throw new RuntimeException("The asset "+myAssetId+" already exists");
42          }
43          MyAsset asset = new MyAsset();
44          asset.setValue(value);
45          ctx.getStub().putState(myAssetId, asset.toJSONString().getBytes(UTF_8));
46      }
47
```

Figura 31: Modelo do Metodo createMyAsset

O método readMyAsset recebe um argumento que é o Id da transação. Caso o Id não corresponda a nenhuma transação uma exceção é lançada, caso corresponda a alguma transação o método retorna as informações da transação (figura 32).

```
47
48      @Transaction()
49      public MyAsset readMyAsset(Context ctx, String myAssetId) {
50          boolean exists = myAssetExists(ctx,myAssetId);
51          if (!exists) {
52              throw new RuntimeException("The asset "+myAssetId+" does not exist");
53          }
54
55          MyAsset newAsset = MyAsset.fromJSONString(new String(ctx.getStub().getState(myAssetId),UTF_8));
56          return newAsset;
57      }
58
```

Figura 32: Modelo da classe readMyAsset

O método delete e update devem receber uma atenção especial, afinal, em um blockchain as informações não são excluídas ou alteradas. O Ledger de cada peer possui duas partes, uma é chamada de world state e outra blockchain. O world state representa o estado atual da transação, enquanto o blockchain possui o log de uma transação em uma estrutura blockchain (TAM KC, 2019). Dessa forma, quando é feito uma exclusão, o que se altera é o estado dessa transação no World State (figura 33).

```
58
59      @Transaction()
60      public void updateMyAsset(Context ctx, String myAssetId, String newValue) {
61          boolean exists = myAssetExists(ctx,myAssetId);
62          if (!exists) {
63              throw new RuntimeException("The asset "+myAssetId+" does not exist");
64          }
65          MyAsset asset = new MyAsset();
66          asset.setValue(newValue);
67
68          ctx.getStub().putState(myAssetId, asset.toJSONString().getBytes(UTF_8));
69      }
70
71      @Transaction()
72      public void deleteMyAsset(Context ctx, String myAssetId) {
73          boolean exists = myAssetExists(ctx,myAssetId);
74          if (!exists) {
75              throw new RuntimeException("The asset "+myAssetId+" does not exist");
76          }
77          ctx.getStub().delState(myAssetId);
78      }
79
80  }
```

Figura 33: modelo com os métodos updateMyAsset e deleteMyAsset

EMPACOTAR, INSTALAR E INSTANCIAR O CHAINCODE

O projeto possui as classes que representam o chaincode, entretanto é necessário inserir o chaincode na rede Hyperledger, para isso o primeiro passo é gerar um arquivo ou um pacote com o código do contrato inteligente. Para isso deve se acessar a extensão da IBM Blockchain Platform instalada no Visual Sudio Code e clicar em no menu de ações do campo Smart Contract e escolher a opção Package Open Project como demonstrado pela figura 34. A plataforma irá requisitar para o desenvolvedor inserir o nome e a versão do chaincode. A versão deve seguir um modelo igual a 0.0.0, no estudo de caso em tela foi informado como versão 0.0.1.

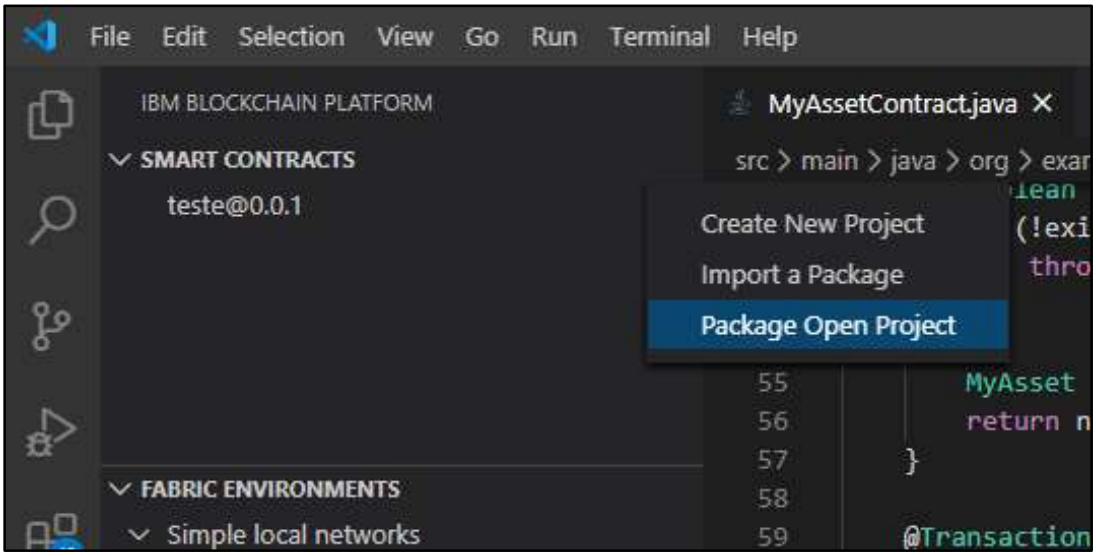

Figura 34: Demonstração da função Package Open Projoject

Após ser gerado o pacote do chaincode deve ser instalado na rede Hyperledger. Porem, antes devemos iniciar uma instancia local da rede. Para isso, na paleta Fabric Enviroments, clicar na instancia da rede que criada por padrão pela IBM Blockchain Platform, no caso de uso em estudo 1 Org Local Fabric. Com isso é carregada as imagens do Docker que compõem a rede Hyperledger.

Com a instancia iniciada será dispobilizada a opção instalação e instanciação. Primeiro clicar em instalação e selecionamos o contrato inteligente, em seguida o mesmo processo na instanciação (FIGURA).

Com o contrato instalado e instanciado o serviço com de blockchain da rede Hyperledger já está pronto para ser consumido. Para isso, utiliza-se o Fabric Gateways para acessar as portas do serviço. Se formos para a paleta de Fabric Gateways e expandirmos a Dropbox Channel vamos encontrar o canal criado pela rede pelo qual serão feitas as transações. Com isso, se expandirmos a Dropbox do canal contido dentro do item Channels teremos os métodos criados anteriormente (figura)

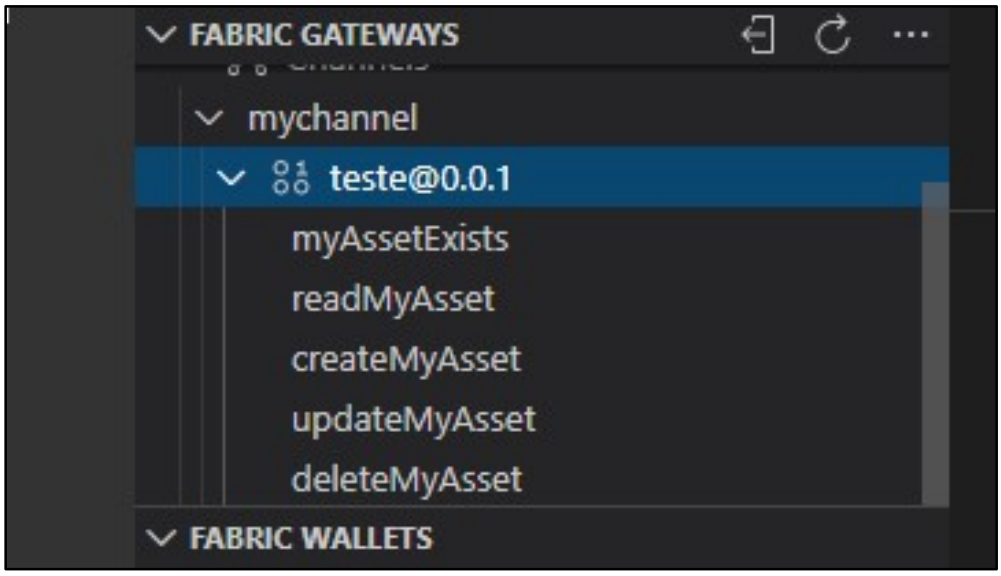

Figura 35: Demonstração dos métodos contidos no Chaincode

Para que possamos transacionar com a instancia da rede local clicamos com o botão direito sobre o método. A plataforma irá disponibilizar duas opções, Evaluate Transaction e Submit Transaction (figura 0). Nos métodos que alteração o estado da rede, como createMyAsset, updateMyAsset e deleteMyAsset, utilizamos a opção Submit Transaction. Por outro lado, quando realizamos apenas consulta como no caso dos métodos myAssetExists e readMyAsset utilizamos o Evaluate Transaction.

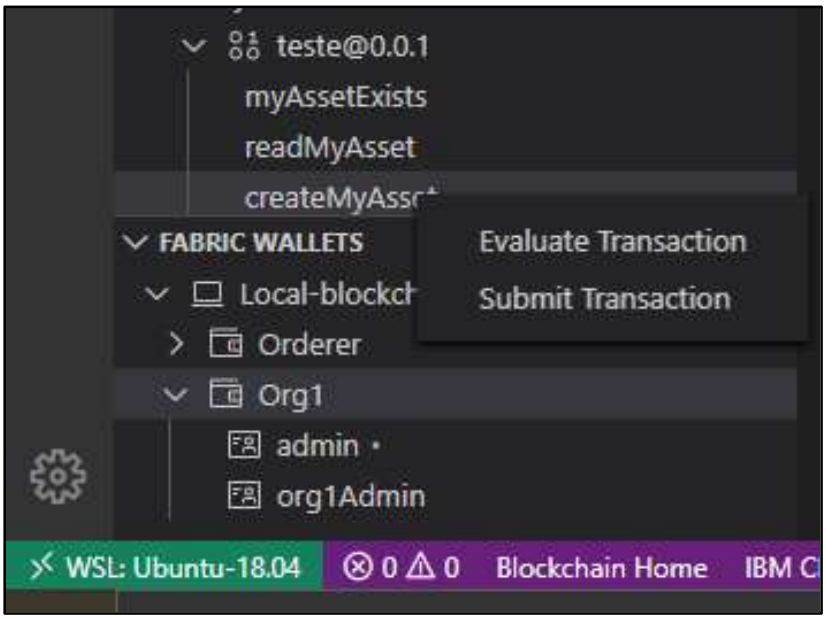

Figura 36: Funções EvaluateTransaction e Submit Transaction

Com isso, conclui-se a demonstração de instalação implementação e consumo de rede Hyperledger local, a fim permitir uma simples uma simples porem completa compreensão do funcionamento de uma rede Blockchain Hyperledger a partir da IBM Blockchain Platform.

APÊNDICE B -REGISTRO DE OPERAÇÕES DE TRANSAÇÕES

Para demonstrar o funcionamento do contrato inteligente instanciado na rede iremos começar pelo método createMyAsset. Ao clicar sobre o referido método e sobre Submit Transaction, a plataforma irá exibir uma caixa de diálogo onde devemos inserir o Id e valor a ser transacionado. No caso em questão O Id será a string “001” e o valor a string “Hello Hyperledger” (figura) e confirmamos a transação.

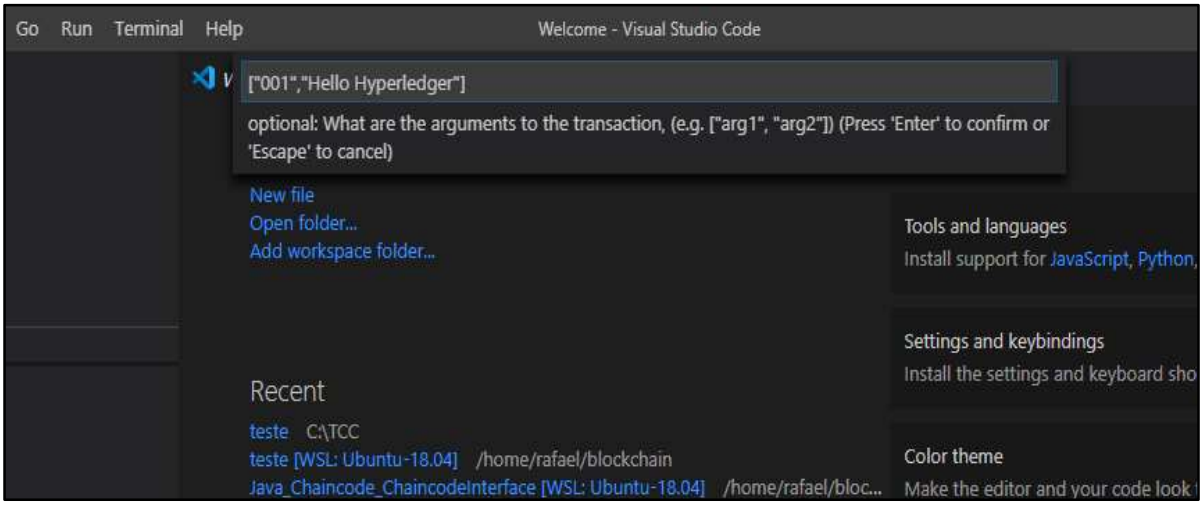

Figura 37: Demonstração de uma transação pelo método createMyAsset

Registro da transação a partir do método createMyAsset:

```
[8/12/2020 11:59:55 PM] [INFO] submitTransaction
[8/13/2020 12:00:29 AM] [INFO] submitting transaction createMyAsset with args 001,H
ello Hyperledger on channel mychannel
[8/13/2020 12:00:32 AM] [SUCCESS] No value returned from createMyAsset
[8/13/2020 12:00:57 AM] [INFO] submitTransaction
```

Registro da transação a partir do método readMyAsset:

```
8/13/2020 12:08:12 AM] [INFO] evaluateTransaction
[8/13/2020 12:08:21 AM] [INFO] evaluating transaction readMyAsset with args 001 on
channel mychannel
```

```
[8/13/2020 12:08:21 AM] [SUCCESS] Returned value from readMyAsset: {"value":"Hello
Hyperledger"}
```

Registro da transação a partir do método updateMyAsset:

```
[8/13/2020 12:21:13 AM] [INFO] submitTransaction
[8/13/2020 12:21:36 AM] [INFO] submitting transaction updateMyAsset with args 001,h
ello-world on channel mychannel
[8/13/2020 12:21:38 AM] [SUCCESS] No value returned from updateMyAsset
```

Registro da transação a partir do método deleteMyAsset:

```
[8/13/2020 12:23:10 AM] [INFO] submitTransaction
[8/13/2020 12:23:15 AM] [INFO] submitting transaction deleteMyAsset with args 001 o
n channel mychannel
[8/13/2020 12:23:18 AM] [SUCCESS] No value returned from deleteMyAsset
```