Logotipo

Descrição gerada automaticamente

Universidade Federal do Rio de Janeiro

Escola Politécnica

MBA em Big Data, Business Intelligence e Business Analytics

(MB3B)

**TÍTULO DA MONOGRAFIA**

Autor:

**\_\_\_\_\_\_\_\_\_\_\_\_ GIULIANO MUNIZ RIBEIRO \_\_\_ \_**

Nome do Aluno e titulação se houver

Orientador:

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

Manoel Villas Boas Junior, M. Sc.

Coorientador:

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

Edilberto Strauss, Ph. D.

Examinador:

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

Claudio Luiz Latta de Souza, M. Sc.

Examinador:

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

Vinicius Drumond Gonzaga, M. Sc.

Examinador:

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

Vinicius Teixeira do Nascimento, M. Sc.

**Rio de Janeiro, junho de 2023**

**Declaração de Autoria e de Direitos**

Eu, **Giuliano Muniz Ribeiro** CPF 024.972.137-63, autor da monografia ***PORTABILIDADE NUMÉRICA EM UMA PLATAFORMA BLOCKCHAIN***, subscrevo para os devidos fins, asseguintes informações:

1. O autor declara que o trabalho apresentado na defesa da monografia do curso de Pós-Graduação, Especialização MBA em Big Data, Business Intelligence e Business Analytics da Escola Politécnica da UFRJ é de sua autoria, sendo original em forma e conteúdo.
2. Excetuam-se do item 1 eventuais transcrições de texto, figuras, tabelas, conceitos e idéias, que identifiquem claramente a fonte original, explicitando as autorizações obtidas dos respectivos proprietários, quando necessárias.
3. O autor permite que a UFRJ, por um prazo indeterminado, efetue em qualquer mídia de divulgação, a publicação do trabalho acadêmico em sua totalidade, ou em parte. Essa autorização não envolve ônus de qualquer natureza à UFRJ, ou aos seus representantes.
4. O autor declara, ainda, ter a capacidade jurídica para a prática do presente ato, assim como ter conhecimento do teor da presente Declaração, estando ciente das sanções e punições legais, no que tange a cópia parcial, ou total, de obra intelectual, o que se configura como violação do direito autoral previsto no Código Penal Brasileiro no art.184 e art.299, bem como na Lei 9.610.
5. O autor é o único responsável pelo conteúdo apresentado nos trabalhos acadêmicos publicados, não cabendo à UFRJ, aos seus representantes, ou ao(s) orientador(es), qualquer responsabilização/ indenização nesse sentido.
6. Por ser verdade, firmo a presente declaração.

Rio de Janeiro, 03 de junho de 2023.

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

GIULIANO MUNIZ RIBEIRO

**UNIVERSIDADE FEDERAL DO RIO DE JANEIRO**

Av. Athos da Silveira, 149 - Centro de Tecnologia, Bloco H, sala - 212,

Cidade Universitária Rio de Janeiro – RJ - CEP 21949-900.

Este exemplar é de propriedade Escola Politécnica da Universidade Federal do Rio de Janeiro, que poderá incluí-lo em base de dados, armazenar em computador, microfilmar ou adotar qualquer forma de arquivamento.

Permitida a menção, reprodução parcial ou integral e a transmissão entre bibliotecas deste trabalho, sem modificação de seu texto, em qualquer meio que esteja ou venha a ser fixado, para pesquisa acadêmica, comentários e citações, desde que sem finalidade comercial e que seja feita a referência bibliográfica completa.

Os conceitos expressos neste trabalho são de responsabilidade do(s) autor(es).

**DEDICATÓRIA**

**(Opcional).**

**AGRADECIMENTO**

Se não estiver inspirado, aqui está uma sugestão:

“Dedico este trabalho ao povo brasileiro que contribuiu de forma significativa à minha formação e estada nesta Universidade. Este projeto é uma pequena forma de retribuir o investimento e confiança em mim depositados.”

**RESUMO**

Introdução:

Conforme definida pela ANATEL (Agência Nacional de Telecomunicações), a portabilidade numérica é um serviço que permite que os usuários dos serviços de telefonia móvel ou fixa mantenham seu número de telefone ao mudar de uma operadora para outra. Esse processo envolve a transferência de informações sensíveis, como número de telefone e os dados de identificação do usuário, entre as operadoras envolvidas (ANATEL 2022).

Além disso, sua implementação e regulamentação define uma empresa independente das operadoras, chamada Entidade Administradora, que é responsável por todos as transações necessárias para que ocorra a portabilidade numérica e ainda atua como a base de dados centralizada, ou seja, até mesmos as operadoras precisam consultá-la para o correto encaminhamento das chamadas telefônicas (ANATEL 2022).

Todos esse elementos motivaram o desenvolvimento deste trabalho e, sob a forma de uma prova de conceito, foi proposta a utilização da tecnologia blockchain como base para desenvolvimento de uma nova forma de implementação da portabilidade numérica.

Por ter sido desenvolvido sob a forma de uma prova de conceito, foi necessária a exploração de outros temas fundamentais, como a infraestrutura computacional da plataforma Azure (Microsoft Azure 2023), a utilização de containers através do framework Docker (Docker e Docker Compose 2023) e a utilização da plataforma que implementa a tecnologia blockchain de forma privada, Hyperleder Fabric (Hyperledger 2023). Cada um desses temas foi abordado com o intuito de viabilizar e explorar ao máximo essa prova de conceito.

A conclusão não poderia ser nada menos do que a apresentação das evidências de que é possível ter uma estrutura descentralizada, confiável, que garanta os dados imutáveis e que traga transparência ao processo.

Palavras-Chave:

Hyperleder Fabric, ANATEL, entidade administradora, descentralização, containers.

**ABSTRACT**

The number portability is a service that allows users of mobile or fixed-line telephony services to keep their phone number when moving to a different service provider. This process involves the transfer of sensitive information, such as phone numbers and user identification data, between the involved service providers. Additionally, its implementation and regulation by ANATEL defines an independent company from the telecommunication operators (Carriers), called the Administrative Entity, which is responsible for all necessary transactions for number portability and also works as the centralized database, meaning even the telecom operators need to consult it for the correct routing of phone calls (ANATEL 2022).

All of these elements motivated the development of this work and, as a proof of concept (POC), the use of blockchain technology was proposed as a basis for the implementation of a new proposal of number portability (ANATEL 2022).

Once it was developed as a proof of concept, it was necessary to explore other fundamental topics, such as the computational infrastructure of the Azure platform ((Microsoft Azure 2023), the use of containers through the Docker framework (Docker e Docker Compose 2023), and the use of the platform that implements blockchain technology in a permissioned and private, Hyperledger Fabric (Hyperledger 2023). Each of these topics was addressed with the aim of making this proof of concept viable and exploring it to the fullest extent possible.

The conclusion could be nothing less than the presentation of evidence that it is possible to have a decentralized, reliable structure that ensures immutable data and brings transparency to the process.

Keywords: Blockchain, Hyperleder Fabric, ANATEL, Number Portability, Decentralization, containers.

**SIGLAS**

|  |  |
| --- | --- |
| ANATEL | Agência Nacional de Telecomunicações |
| CMMS | Computerized Maintenance Management System |
| UFRJ | Universidade Federal do Rio de Janeiro |
|  |  |

**LISTA DE FIGURAS**

[Figura 1.1 - Processo de portabilidade adaptado de (CURIONI e Gil 2009) 7](#_Toc135175526)

[Figura 1.2 - Exemplo de consulta sobre histórico de portabilidade. 8](#_Toc135175527)

[Figura 1.3 - Fluxo de uma solicitação de portabilidade numérica de um usuário. 9](#_Toc135175528)

[Figura 2.1 - Quando e qual tipo blockchain deve ser usado (M. Belotti 2019) . 11](#_Toc135175529)

[Figura 2.2 – Relação entre a inovação e a complexidade (Iansiti e Lakhani 2017). 12](#_Toc135175530)

[Figura 2.3 - Fluxo das transações da plataforma Hyperledger Fabric. (Androulaki, et al. 2018) 13](#_Toc135175531)

[Figura 3.1 - Painel de controle da plataforma Azure. 14](#_Toc135175532)

[Figura 3.2 - Acesso SSH à VM WebApp 15](#_Toc135175533)

[Figura 3.3 – Utilização da aplicação FileZilla Client. 15](#_Toc135175534)

[Figura 3.4 - Pré-requisitos da plataforma Hyperledger Fabric. 16](#_Toc135175535)

[Figura 3.5 - Visualização da rede blockchain utilizada neste trabalho. 17](#_Toc135175536)

[Figura 3.6 -Aplicação CURL usada para clonar o repositório fabric-samples. 20](#_Toc135175537)

[Figura 3.7 - Script de instalação do Hyperledger Fabric 20](#_Toc135175538)

[Figura 3.8 - Script network.sh e suas opções. 21](#_Toc135175539)

[Figura 3.9 - Script network.sh com a opção de inicialização da rede 21](#_Toc135175540)

[Figura 3.10 - comando fabric-ca-client e os prinicipais parâmetro e opções. 22](#_Toc135175541)

[Figura 3.11 - Conteúdo ao arquivo cert.perm - chave pública da Org1. 23](#_Toc135175542)

[Figura 3.12 - Conteúdo ao arquivo localhost-7054-ca-org1.pem - chave privada da Org1 23](#_Toc135175543)

[Figura 3.13 - Perfil de conexão de cliente (Client Connection Profile - CCP) 28](#_Toc135175544)

[Figura 3.14 - Resposta do comando mostrando os containers criados para os nós. 29](#_Toc135175545)

[Figura 3.15 - Bloco gênese criado como mychannel.block 30](#_Toc135175546)

[Figura 3.16 - Estado do canal **mychannel** 31](#_Toc135175547)

[Figura 3.17 - Comando peer channel Join executado para Org1. 32](#_Toc135175548)

[Figura 3.18 - Comando peer channel join executado para Org2. 32](#_Toc135175549)

[Figura 3.19- Comando peer channel fetch executado em Org1. 33](#_Toc135175550)

[Figura 3.20 - Comando peer channel fetch executado em Org2. 33](#_Toc135175551)

[Figura 3.21 - Confirmação do nó Org1MSP incluído como nó âncora. 34](#_Toc135175552)

[Figura 3.22 - Confirmação do nó Org2MSP incluído como nó âncora. 34](#_Toc135175553)

[Figura 3.23 - Revisão da rede blockchain utilizada neste trabalho. 35](#_Toc135175554)

[Figura 3.24 - Empacotamento e definição do Package ID 37](#_Toc135175555)

[Figura 3.25 - Instalação do chaincode em peer0.org1.example.com 38](#_Toc135175556)

[Figura 3.26 – Comando peer lifecycle chaincode approveformyorg em **org1.** 39](#_Toc135175557)

[Figura 3.27 - Comando peer lifecycle chaincode approveformyorg em **org2**. 39](#_Toc135175558)

[Figura 3.28 - Confirmação das definições do chaincode. 40](file:///E:\MBA\TCC\app\myApp\Docs\Modelo%20MB3B%20v.13.Giuliano%20Muniz%20Ribeiro.rev01.docx#_Toc135175559)

[Figura 3.29 - Integrações entre o cliente, acesso externo e a rede fabric-test. 40](#_Toc135175560)

[Figura 3.30 - Estrutura dos dados. 41](#_Toc135175561)

[Figura 3.31 - Código appCreate.js e a criação da carteira (wallet) 42](#_Toc135175562)

[Figura 3.32 - Certificados dos usuários admin e appUser na carteira (wallet). 42](#_Toc135175563)

[Figura 3.33 - Criação do gateway para conexão ao nó e ao canal. 42](#_Toc135175564)

[Figura 3.34 - Criação de um novo usuário. 43](#_Toc135175565)

[Figura 3.35 - Resposta à execução de appCreate.js e appQuery.js. 43](#_Toc135175566)

[Figura 3.36 - Estrutura do apiServerNP.js e sua APIs. 44](#_Toc135175567)

[Figura 3.37 – Código da função de Invocação da API InitLeadger. 45](#_Toc135175568)

[Figura 3.38 - Código da função de Invocação da API Create 46](#_Toc135175569)

[Figura 3.39 - Código da função de Invocação da API Transfer. 47](#_Toc135175570)

[Figura 3.40 - Código da função de Invocação da API Query 47](#_Toc135175571)

[Figura 3.41 - Código da função de Invocação da API QueryAll 48](file:///E:\MBA\TCC\app\myApp\Docs\Modelo%20MB3B%20v.13.Giuliano%20Muniz%20Ribeiro.rev01.docx#_Toc135175572)

[Figura 3.42 - Arquivo AssetTransfer.js com todas os Chaincodes. 49](#_Toc135175573)

[Figura 3.43 - Chaincode InitLedger. 49](#_Toc135175574)

[Figura 3.44 - Chaincode CreateAsset. 50](#_Toc135175575)

[Figura 3.45 - Chaincode ReadAsset. 50](#_Toc135175576)

[Figura 3.46 – Chaincode UpdateAsset. 50](#_Toc135175577)

[Figura 3.47 - Chaincode DeleteAsset. 51](#_Toc135175578)

[Figura 3.48 - Chaincode AssetExists. 51](#_Toc135175579)

[Figura 3.49 - Chaincode TransferAsset. 51](#_Toc135175580)

[Figura 3.50 - Chaincode GetAllAssets. 52](#_Toc135175581)

[Figura 1.1.1 - Código da função de Invocação da API QueryAll 65](file:///E:\MBA\TCC\app\myApp\Docs\Modelo%20MB3B%20v.13.Giuliano%20Muniz%20Ribeiro.rev01.docx#_Toc135175582)

**LISTA DE TABELAS**

[Tabela 1 - Faixa de endereços IP definido automaticamente pelo Docker. 29](#_Toc133193126)

**Sumário**

Capítulo 1: Introdução 1

1.1 - Tema 2

1.2 - Justificativa 3

1.3 - Objetivos 4

1.4 - Delimitação 6

1.5 - Metodologia 6

1.5.1 – Identificação da natureza do problema de portabilidade numérica 7

1.5.2 – Definição da plataforma de implementação da tecnologia blockchain 9

Capítulo 2: Embasamento Teórico 10

Capítulo 3: Propostas Tecnológicas 13

3.1 - Infraestrutura Computacional 13

3.2 - Plataforma Hyperledger Fabric 16

3.2.1 - Elementos da rede Blockchain 16

3.2.2 - Conceitos do Hyperledger Fabric 16

3.2.3 - Instalação do Hyperledger Fabric 18

3.2.4 - Instalação dos exemplos, executáveis e imagens Docker 19

3.2.5 - Inicializando a rede e os componentes do Hyperledger Fabric 20

3.2.5.1 - Criando o canal 28

3.2.5.2 - Criando o bloco de gênese 29

3.2.5.3 - Inserir os nós no canal 31

3.2.5.4 - Selecionando os nós âncora (anchor peer) 32

3.3 - Smart Contracts 34

3.3.1 - Smart Contracts no Hyperledger Fabric 34

3.3.2 - Packing ID - Empacotamento 35

3.3.3 - Instalação do chaincode em cada nó 36

3.3.4 - Aprovação das definições do chaincode 37

3.3.5 - Confirmação da definição do chaincode 38

3.3.6 - Invocação do chaincode 39

3.3.6.1 - Invocação por Aplicativo Cliente – Cliente-App 41

3.3.6.2 - Invocação por API Server 43

3.3.6.2.1 - API initLedger 44

3.3.6.2.2 - API Create 45

3.3.6.2.3 - API Transfer 46

3.3.6.2.4 - API Query 47

3.3.6.2.5 - API QueryAll 47

3.4 - Chaincodes 48

3.4.1 - Chaincode InitLedger 49

3.4.2 - Chaincode CreateAsset 49

3.4.3 - Chaincode ReadAsset 50

3.4.4 - Chaincode UpdateAsset 50

3.4.5 - Chaincode DeleteAsset 50

3.4.6 - Chaincode AssetExists 51

3.4.7 - Chaincode TransferAsset 51

3.4.8 - Chaincode GetAllAssets 51

Capítulo 4: Resultados Obtidos 52

Referências Bibliográficas 55

Bibliografia 55

# Introdução

Portabilidade numérica é um mecanismo de transferência de um número de telefone de uma Prestadora Doadora para uma Prestadora Receptora a partir da solicitação de um usuário e é descrita pela Agência Nacional de Telecomunicações (ANATEL) da seguinte maneira:

“A portabilidade possibilita ao cliente dos serviços de telefonia fixa (Serviço Telefônico Fixo Comutado - STFC) e móvel (Serviço Móvel Pessoal - SMP) manter o seu número de telefone (Código de Acesso de Usuário e Códigos não Geográficos), independentemente da prestadora de serviço a qual esteja vinculado” (Anatel 2023).

Esse serviço é regulamentado pelo Regulamento Geral de Portabilidade aprovado pela Resolução 460/2007 (Anatel 2023) que define também a Entidade Administradora, que centraliza e gerencia uma base de dados de referência da portabilidade numérica contendo todas as informações necessárias para o correto encaminhamento de chamadas e mensagens, que permita a qualquer tempo realizar auditorias e obter relatórios, e exige ainda que esta entidade administradora a neutralidade com relação às prestadoras de serviços de telecomunicações.

Esta descrição do serviço de portabilidade numérica aponta para o problema que será tratado neste trabalho que é a proposta de substituição de um modelo que depende de uma entidade para garantir a confiabilidade das transações de portabilidade numérica por um modelo descentralizado e que garanta essa confiabilidade entre as operadoras de telecomunicações, os usuários dos serviços de telecomunicações e a agência reguladora ANATEL.

Qual solução tecnológica pode ser a opção capaz de substituir esse modelo centralizado, onde as transações dependem da Entidade Administradora e que também apresente a confiabilidade necessária frente aos desafios das transações de portabilidade numérica?

A resposta para esta pergunta virá pela proposta de descentralização e manutenção da confiabilidade das transações que são obtidas através da tecnologia blockchain. Esta tecnologia utiliza um conceito oriundo de contabilidade empresarial chamado livro razão (*ledger*, em inglês) para implementar uma estrutura que garanta que os resultados das transações serão sequenciados por ordem de execução, serão imutáveis e tratados em uma rede distribuída de entidades que possuem altíssmo grau de desconfiança entre si.

Para sustentação desta proposta, este trabalho foi desenvolvido através de uma prova de conceito (POC – Proof of Concept) que será apresentado com mais detalhes no capítulo 1.5 – Metodologia.

Três grandes temas que direcionam este trabalho são discutidos mais detalhadamente no capítulo 2 – Embasamento Teórico, que são a análise da portabilidade numérica, identificação de sua natureza e modelagem dos dados necessários nestas transações, a discussão sobre a infraestrutura computacional e de desenvolvimento de software que implementa essa modelagem de dados e suas regras de negócio e, finalmente, a solução proposta para este problema de portabilidade numérica através da tecnologia Blockchain.

No capítulo 3 – Propostas Tecnológicas, será apresentada a forma como esta prova de conceito foi implementada e os detalhes da infraestrutura e ambiente de desenvolvimento. É o capítulo onde os detalhes técnicos serão apresentados desde a utilização da plataforma Azure (Microsoft Azure 2023), do serviço Docker (Docker e Docker Compose 2023) para utilização da plataforma que implementa a tecnologia blockchain de forma permissionada e privada, Hyperleder Fabric (Hyperledger 2023)

No Capítulo 4 – Resultados obtidos ....

No capiítulo 5 – conclusão e trabalhos futuros : estrtura mais simples nesse projeto.

## Tema

A prova de conceito aplicada na resolução do problema aborda quatro grandes áreas do conhecimento que são abordadas nos próximos capítulos.

Em telecomunicações, o tema de portabilidade numérica é bastante abrangente e possui diversas maneiras para sua implementação conforme apresentado na norma RFC 3482 (Foster, McGarry e Yu 2003). A Resolução 460/2007 da ANATEL (Anatel 2023) define a forma como este serviço deve ser implementado no Brasil.

Existe ainda uma pequena análise sobre a utilização de servidores em nuvem e os critérios adotados para escolha da plataforma da Azure da Microsoft (Azure 2023).

A instalação e configuração da plataforma Hyperledger Fabric e o desenvolvimento dos códigos que compõem os contratos inteligentes (*smart contracts*) abre também um outro tema que envolve o caráter mais técnico sobre utilização de containers (Docker Inc. s.d.) e a instalação e utilização de Javascript (OpenJS JavaScript 2023).

A solução final baseada na plataforma blockchain para o problema de portabilidade numérica será tratado como uma prova de conceito através da plataforma Hyperledger Fabric e a implementação dos contratos inteligentes, que são a transcrição das regras de negócio da portabilidade para a liguagem de programação, será através do Javascript.

## Justificativa

Do ponto de vista do usuário, o número de telefone celular se tornou o seu principal ativo digital e também uma identidade que ele precisa que seja mantida, uma vez que suas transações digitais irão depender disso. Essa é uma justificativa para que os usuários não fiquem vinculados a uma operadora e que sejam capazes de execer sua opção de escolha em trocar entre as operadora de telefonia.

Por outro lado, o ambiente comercial entre as operadoras de telecomunicações possui enormes desafios. Dentre eles, além da necessidade de tratamento das informações de milhões usuários, existe a inclusão da informação sobre qual operadora destino a qual o usuário pertence, que caracteriza a portabilidade numérica.

Atualmente é necessário um sistema centralizado de uma empresa isenta e independente das operadoras para armazenamento, tratamento e disponibilização desses dados para viabilizar a portabilidade numérica (Anatel 2023). Essas transações atuam em uma estrutura dos dados de um usuário e criam uma seqüência de eventos de portabilidade que torna a sua história como uma cadeia de eventos imutável e que necessita de transparência para que todas as operadoras utilizem essa informação, sem afetar o serviço ofertado ao usuário.

Esse problema foi resolvido com a tecnologia disponível à epoca em que foram tratados, e que é analisado neste trabalho sob a ótica da tecnologia blockchain.

Inicialmente é necessário analisar a natureza do problema que sugere um bloco de dados de um usuário alterado conforme sua solicitação de mudança entre as operadoras de telefonia. Cada uma dessa operadoras possui o seu objetivo comercial de aumentar sua base de usuários e evitar que migrem para outra operadora concorrente.

Esse fatores justificam a utilização da tecnologia blockchain, uma vez que os princípios básico desta tecnologia são a base distribuída e que cada parte(nó) da rede blockchain tenha acesso à base e à história completa dos blocos. Além diso deve ter a transparência em cada transação e identificação de cada usuário que atue na alteração dos blocos garantindo a irreversibilidade (imutabilidade) para que os blocos alterados sejam imutáveis e que as regras de negócio sejam implementadas em código de programação para, então, seerem executadas sob a demanda dos usuários (Iansiti e Lakhani 2017).

Para que isso seja possível, o blockchain utiliza um conceito contábil chamado de Livro razão, ou *Ledger* em inglês, onde todas as transações realizadas de uma empresa são armazenadas em uma ordem ordem cronológica. Cada informação inserida no livro razão ficará permanentemente e de forma imutável (PIRES​, et al. 2018). Esse conceito de livro razão ( *ledger* ) é fundamental para entendimento da tecnologia blockchain e também a justificativa para que seja utilizada como solução para o problema de portabilidade numérica.

Distributed Ledger Technology (DLT) é a tecnologia de armazenamento e atualização de um livro razão ( *ledger* ) de maneira distribuída e descentralizada. Cada computador representa um nó e possui uma cópia da base de dados, e não existe nenhuma entidade centralizadora que seja responsável pela sua manutenção, atualização e distribuição dessa base de dados. A alteração dessa base é feita de forma independente e existem alguns algoritmos de consenso que permitem a votação entre todos os componentes dessa rede para definição da última versão dessa base (livro razão). Todos esses processos de consenso e manutenção base que representa o livro razão acontece de forma automatizada em todos os nós dessa rede (PIRES​, et al. 2018).

A portabilidade numérica acontece entre um grupo de empresas concorrente e operadoras de telefonia. Isso direciona a análise para uma característica da tecnologia blockchain que é a opção do seu modo de operação público ou privado. Esse modo de operação é classificado conforme permissão de acesso à rede blockchain, no modo público o acesso à rede é aberto e público. Neste caso, qualquer computador pode participar e atuar nessas transações e o processo de consenso é necessário para garantir a confiabilidade. Este modo é adotado, por exemplo, nas negociações da criptomoeda Bitcoin (NAKAMOTO 2008) e que não é objeto de análise deste trabalho. De maneira análoga, se o modo de acesso é definido como privado, significa que as modificações das transações são confiadas apenas a um grupo de computadores.

Com todos esses pontos apresentados, é possível sugerir uma proposta de que a solução do problema de portabilidade numérica seja resolvido através da implementação de uma rede com a tecnologia blockchain e que irá atuar no modo privado.

A validação dessa proposta é feita através de uma prova de conceito (POC) e a forma de implementação, problemas encontrados e os resultados obtidos são o objetivo deste trabalho.

## Objetivos

Como este trabalho foi desenvolvido baseado em uma prova de conceito (POC) e como o tema abrange diferentes áreas do conhecimento, conforme já tratado anteriormente, é possível definir diferentes objetivos que tratem desde a definição de infraestrutura computacional e de desenvolvimento de software, até aos objetivos que focam nas plataformas que implementam a tecnologia blockchain, sua relação com o tema e finalmente a visão holística que a implentação da portabilidade numérica imprime ao trabalho.

O início da análise do tema já define um objetivo elementar que é a identificação da natureza do problema da portabilidade numérica e sua aderência à tecnologia blockchain. Dentro dessa identificação está primordialmente a necessidade de um diagrama de transição de estados que represente o ciclo de portabilidade numérica para que seja implementado conforme a plataforma adotada nesta prova de conceito.

Em seguida, e já com o caráter prático desta prova de conceito, a definição do ambiente computacional onde a prova de conceito foi implementada e que possa ser reproduzida é outro objetivo deste trabalho.

Com esta definição do ambiente virtualizado e de conteriners, um outro objetivo é a definição da infraestrutura computacional para sua implementação e que seja versátil para criação de novas máquinas virtuais conforme a demanda. Um dos critérios utilizados nesta escolha é que o resultado final esteja disponível para uma apresentação remota e que não tenha limitações físicas como por exemplo, utilizar máquinas virtuais em um laptop limita fisicamente a expansão ou criação de novas máquinas virtuais por causa dos recursos limitados desta máquina, portanto é necessário a utilização de ambientes em nuvem que podem atender esta demanda.

Ainda dentro desta definição de infraestrutura computacional, um outro objetivo geral é a definida do ambiente de desenvolvimento que compreende editor de textos aderente à linguagem de programação escolhida e o ambiente de versionamento e atualização do código.

Ao se tratar de objetivos mais específicos, existe a definição da tecnologia para implementação da rede blockchain que atenda a necessidade de ser uma rede privada de processamento distribuído e que tenha a tecnologia DLT (*Distributed Ledger Technology*). A escolha da plataforma Hyperledger Fabric (Hyperledger 2023) atende aos critérios da natureza do problema de portabilidade numérica tratado neste trabalho, e todos os detalhes de sua configuração também são objetivos.

Além disso também são os objetivos, a identificação das entidades necessárias na implementação do Hyperledger Fabric e a vinculação com as entidades do mundo real, como por exemplo as operadoras de telefonia, a Anatel e a entidade administradora.

A autenticação das entidades que irão permitir que a transação de portabilidade numérica seja efetivada é um outro objetivo deste trabalho e a apresentação da relação de confiança entre os envolvidos.

Com todos os objetivos acima definidos e adequadamente alcançados, um outro objetivo é a implementação dos Contratos Inteligentes (Smart Contracts) para que representem o diagrama de estado do ciclo de portabilidade e que tenham as integrações e validações das entidades que compõem a rede blockchain.

Como objetivo final, a validação e apresentação de evidências que comprovem que as transações acontecem com os devidos consensos das entidades e que justifiquem a prova de conceito para o problema de portabilidade numérica.

## Delimitação

Dada a extensa abrangência deste trabaho e também pela complexidade da infraestrutura computacional apresentada, existem alguns fatores que seriam naturalmente considerados como objetivo mas que não são abordados neste trabalho e compõem sua delimitação.

Foi utilizada a plataforma de Cloud Computing, Microsoft Azure (Microsoft Azure 2023) para criação das máquinas virtuais com sistemas operacionais Linux Ubuntu. Como foi utilizada uma licença Azure for Students (Azure 2023) os custos relacionados com os recursos consumidos neste trabalho não estão sendo analisados.

A plataforma de tecnologia Blockchain utilizada é a Hyperledger Fabric (Hyperledger 2023) e o fatores que remetem ao desempenho ou performance desta plataforma ou mesmo da aplicação desenvolvida, não estão como objetivos deste trabalho.

Sobre as autenticações, seus certificados e métodos, serão abordados em um nível de profundidade suficiente para a validação das relações de confiança e comprovação de que as transações tenham o grau de segurança esperados.

Desta forma, o trabalho limitou-se a analisar e documentar as infraestruturas computacional e de desenvolvimento de software escolhidas, assim como a plataforma de implementação da tecnologia blockchain e ainda a análise e documentação da portabilidade numérica.

## Metodologia

Os objetivos e conclusões desse trabalho são baseados em uma prova de conceito sobre aplicação da tecnologia blockchain na resolução do problema de portabilidade numérica.

Para que esta prova de conceito fosse possível, foi necessário dividir o problema em três partes distintas. A raiz do problema é a portabilidade numérica e foi necessário começar a análise pela identificação de sua natureza para que todas as proximas etapas deste trabalho se justifiquem.

Com a natureza do problema identificada, foi necessário compreender a tecnologia blockchain e suas variações para que seja a solução do problema original e em seguida foi necessário identificar quais as plataformas que implementam essa tecnologia da forma esperada para resolução do problema.

A terceira parte de resolução deste problema é de caráter prático e envolveu a definição da infraestrutura computacional e as ferramentas de desenvolvimento de software e testes.

## 1.5.1 – Identificação da natureza do problema de portabilidade numérica

A portabilidade numérica é regulamentada na Resolução 460 de 19 de março de 2007, e substituída pela Resolução 749 de 15 de março de 2022 (Anatel 2023). Nesta resolução são definidos os atores da portabilidade numérica e a arquitetura centralizada de acesso a base de dados chamada de Base de Dados Nacional de Referência da Portabilidade (BDR).



Figura . - Processo de portabilidade adaptado de (CURIONI e Gil 2009)

Esta base de dados centralizada BDR é mantida e atualizada pela Entidade Administradora, que é uma empresa independente e de neutralidade comprovada e precisa garantir que os registros de portabilidade numérica de um usuário devem ser mantidos por 10 anos.

A Entidade Administradora definida pela ANATEL foi a empresa ABR Telecom que disponibiliza a consulta do histórico da portabilidade no site abaixo. Neste exemplo é apresentado o resultado para o número 21-99405.4384 através de consulta ao site <https://consultanumero.abrtelecom.com.br/>



Figura . - Exemplo de consulta sobre histórico de portabilidade.

A partir desta regulamentação da ANATEL para implementação da portabilidade numérica, é possível desenhar o fluxo que atenda os seus requisitos e definir as características necessárias da plataforma que implementa a tecnologia blockchain.

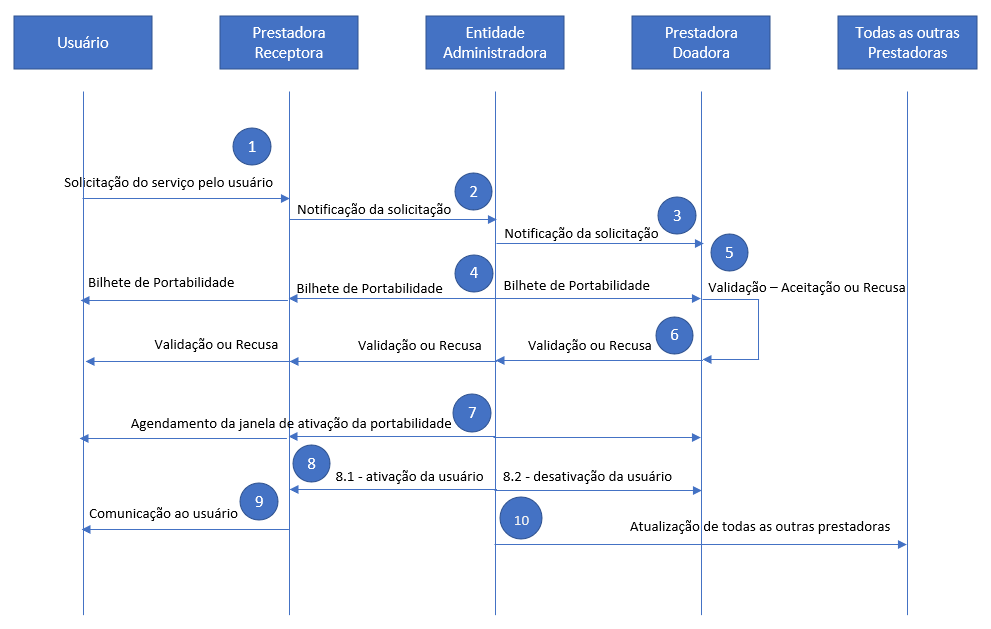


Figura . - Fluxo de uma solicitação de portabilidade numérica de um usuário.

## 1.5.2 – Definição da plataforma de implementação da tecnologia blockchain

Como já apresentado anteriormente, os requisitos necessários da plataforma que implemente a tecnologia blockchain fazem com que a escolha da plataforma Hyperledger Fabric (Hyperledger 2023) atenda a esses critérios.

# Embasamento Teórico

Em telecomunicações, o tema de portabilidade numérica é bastante abrangente e possui diversas opções de implementação. No Brasil a regulamentação da ANATEL (Anatel 2023) define a forma como este serviço foi implementado.

Dois trabalhos foram fundamentais para as conclusões sobre portabilidade numérica e o seu posicionamento dentro da tecnologia Blockchain:

O primeiro trabalho (CURIONI e Gil 2009) apresenta uma visão sobre o conceito e a implementação da portabilidade numérica no Brasil.

O outro trabalho (Moreira, et al. 2009) apresenta uma abordagem mais técnica, apresentando os diversos tipos de solução de portabilidade numérica conforme norma RFC 3482 (Foster, McGarry e Yu 2003) e a opção utilizada no Brasil. Possui diversos exemplos práticos de topologia de rede de telecomunicações que, apesar de não serem o objetivo deste projeto, servem para trazer o tema para o campo prático.

Para entendimento da tecnologia blockchain o artigo “*A Vademecum on Blockchain Technologies: When, Which, and How*” (M. Belotti 2019) apresenta todas as características dessa tecnologia, com ênfase para o modo de permissão de acesso à rede blockchain que são privadas ou públicas. Apresenta ainda a questão fundamental deste trabalho que é “Quando usar a tecnologia blockchain?”. Para auxiliar na tomada de decisão sobre quando e qual tecnologia blockchain utilizar, este artigo apresenta as seguintes questões:

1 – É preciso armazenar e compartilhar dados que podem ser tratados como um livro-razão (ledger)?

2 - Existem vários “escritores” em potencial, ou seja, existem várias entidades que irão efetuar transações?

3 – A quem é confiada a manutenção deste livro-razão?

3.a – Uma entidade externa?

3.b - Um grupo de atores seleto?

3.b) Você precisa que o livro razão seja publicamente validado?

3.c - Uma comunidade pública?...

A definição final não vem a partir das respostas diretas à estas perguntas, mas da análise da natureza do problema que se propõe resolver frente a esses questionamentos.

Este artigo estende esses questionamentos sob forma de um diagrama para tomada de decisão, conforme mostrado abaixo.

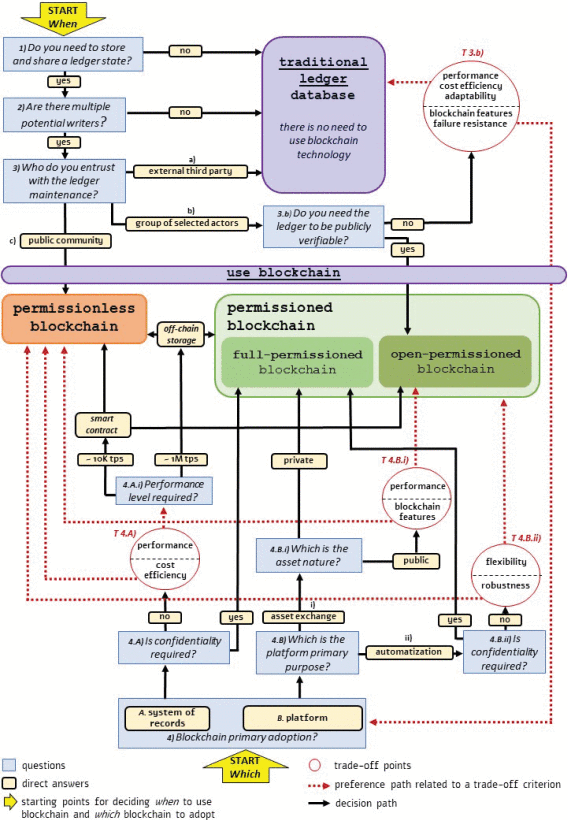
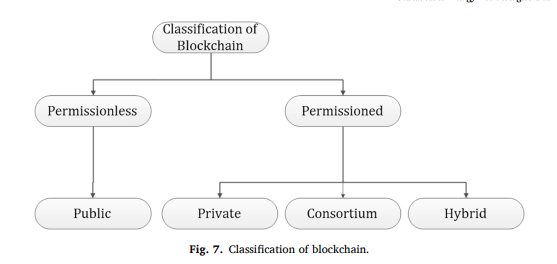


Figura . - Quando e qual tipo blockchain deve ser usado (M. Belotti 2019) .

Quando e qual tipo de blockchain usar em vez de adotar um sistema de banco de dados tradicional. Os círculos vermelhos mostram os pontos de análise entre aspectos fundamentais para os diferentes casos de uso de blockchain. As setas vermelhas indicam a consequência de dar prioridade a um aspecto em detrimento do outro, enquanto as setas pretas relatam respostas a todas as perguntas – que vêm com um pedido – de qualquer pessoa interessada na tecnologia blockchain (M. Belotti 2019).



Um outro artigo utilizado neste trabalho tem como título “The Truth about Blockchain” (Iansiti e Lakhani 2017) e compara a tecnologia Blockchain com outros casos de tecnologias disruptivas, como por exemplo o TCP/IP. Este artigo apresenta os cinco princípios básicos do blockchain que são a base de dados distribuída, as transações ponto a ponto (*peer to peer*), a transparência, a imutabilidade (transações irreversíveis) e a implementação dos contratos sob a lógica computacional.

Existe ainda uma análise de adoção de novas tecnologias, apresentando um gráfico com as quatro fases dessa adoção, onde cada fase é definida pela relação de novidade da aplicação em função da complexidade de tornar essa nova tecnologia ativa e criando valores....

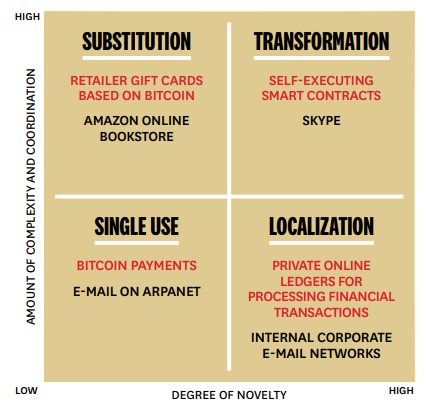


Figura . – Relação entre a inovação e a complexidade (Iansiti e Lakhani 2017).

Uma opção de utilização da plataforma Hyperledger Fabric como solução para o problema de portabilidade numérica é apresentada no artigo “The Design of a Mobile Number Portability System on a Permissioned Private Blockchain Platform” (Krishnaswamy, et al. 2019). Este artigo possui o mesmo posicionamento teórico deste trabalho com a diferença em dois pontos fundamentais que é o fluxo da portabilidade numérica implementada no Brasil e a proposta a criação de diferentes canais onde cada par de operadoras atua em um canal separado. Isso torna a solução mais complexa, porém aumenta a confiabilidade das transações mantendo apenas as operadoras envolvidas na transação.

Sobre a mesma plataforma Hyperledger Fabric, o artigo “Hyperledger Fabric: A Distributed Operating System for Permissioned Blockchains” (Androulaki, et al. 2018) apresenta conceitos fundamentais para o entendimento e implementação da solução. Introduz o conceito de Chaincode, que é o nome dado aos contratos inteligentes (*Smart Contract*) na plataforma Hyperledger Fabric e descreve a utilização da entidade ordenadora (*orderer*) sua arquitetura e também as limitações. Apresenta também a visão de todas as fases e o fluxo de todas as transações....

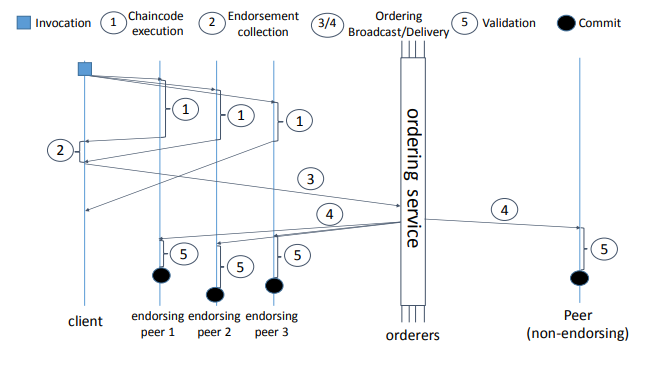


Figura . - Fluxo das transações da plataforma Hyperledger Fabric. (Androulaki, et al. 2018)

Sobre a plataforma Hyperledger Fabric, existe uma vasta documentação (Hyperledger Fabric - Key Concepts 2023) tanto sobre sua arquitetura quanto sobre os destalhes de sua instalação, configuração e testes, e que está sendo utilizada como referência neste trabalho.

# Propostas Tecnológicas

A solução do problema de portabilidade numérica discutido neste trabalho teve como proposta a implementação de uma prova de conceito (POC) com as seguintes fases discutidas logo abaixo.



## Infraestrutura Computacional

Foi utilizada a plataforma de Cloud Computing, Microsoft Azure (Microsoft Azure 2023) para criação das máquinas virtuais com sistemas operacionais Linux Ubuntu através da licença Azure for Students (Azure 2023).

Para instalação das ferramentas necessárias para a plataforma Hyperledger Fabric, foi criada a máquina virtual (VM) chamada *WebApp*.

|  |  |
| --- | --- |
|  |  |
| Figura . - Painel de controle da plataforma Azure. | |

O acesso à VM WebApp é por SSH utilizando o IP Público definido pela plataforma Azure.

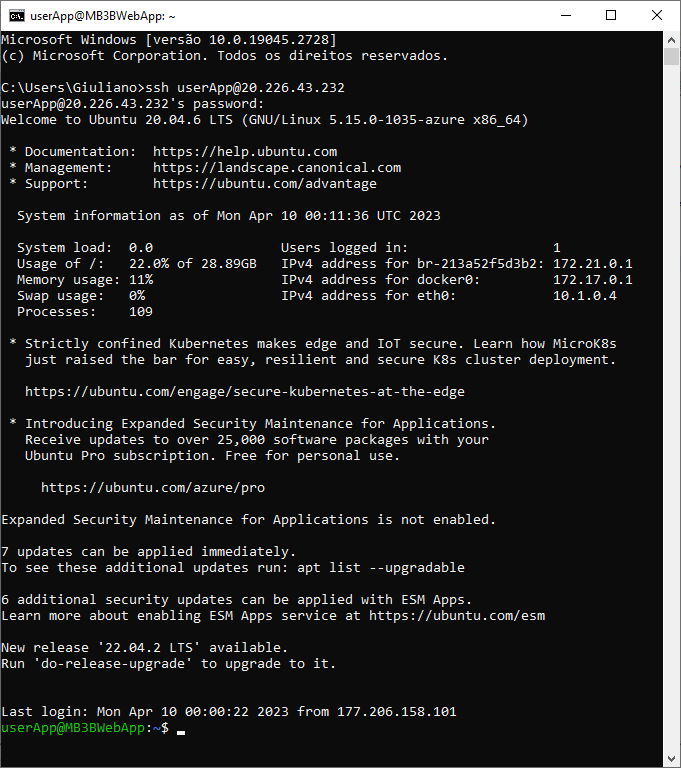


Figura . - Acesso SSH à VM WebApp

Outra ferramenta utilizada para transferência de arquivos através do protocolo SFTP é a aplicação FileZilla Client 3.63.2.1, conforme mostrado abaixo:

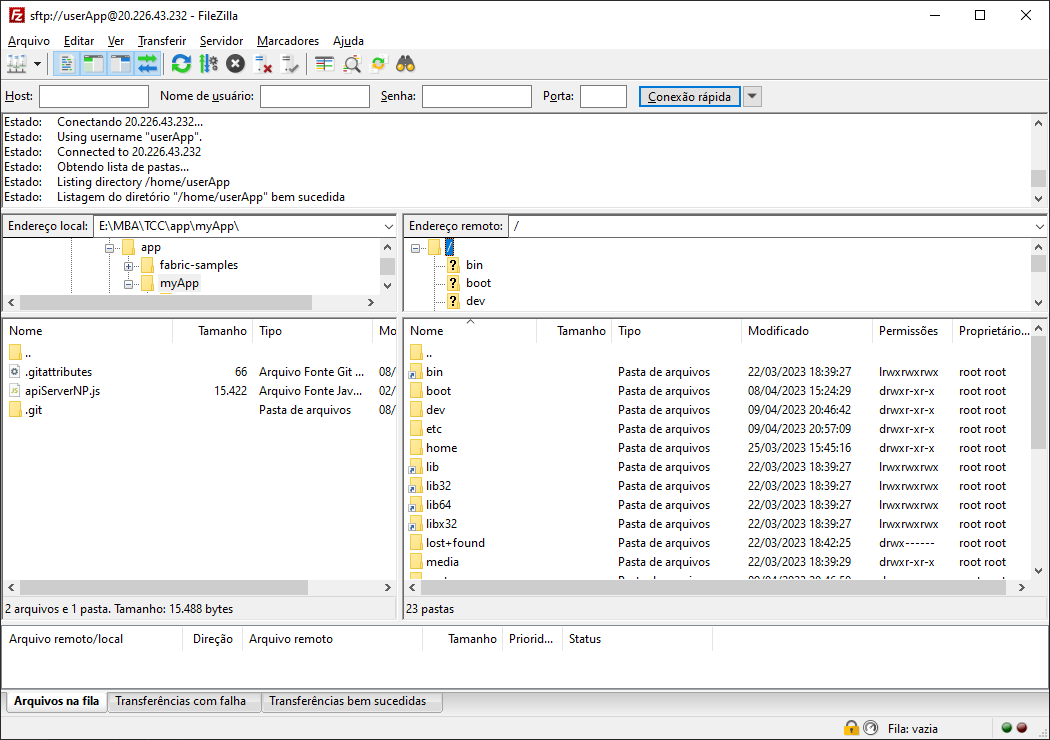


Figura . – Utilização da aplicação FileZilla Client.

Os pré-requisitos apresentados pela Hyperledger Fabric (Hyperledger Fabric - PreReqs 2023) são as aplicações listadas abaixo:

**Git** - Sistema de controle de versão gratuito e de código aberto.

Instalação no Linux Ubuntu:

$ sudo apt-get install git

**cURL** - Usado em linhas de comando ou scripts para transferir dados.

Instalação no Linux Ubuntu

$ sudo apt-get install curl

**Docker** - Plataforma open source que viabiliza a criação e administração de ambientes isolados, containers.

$ sudo apt-get -y install docker-compose

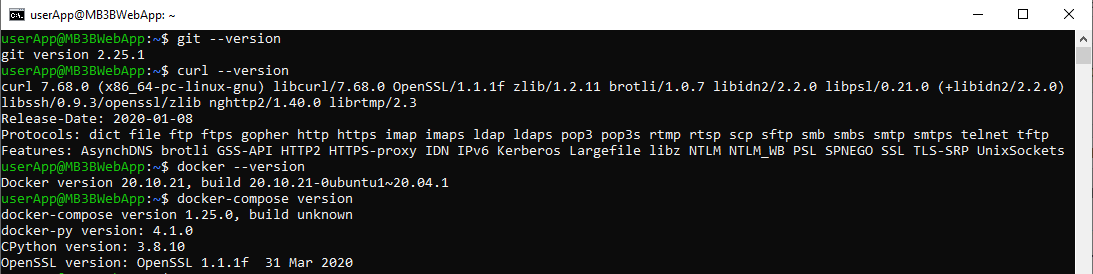


Figura . - Pré-requisitos da plataforma Hyperledger Fabric.

## Plataforma Hyperledger Fabric

## Elementos da rede Blockchain

A rede Blockchain implementada nesta prova de conceito simula a relação de duas operadoras de telefonia e a agência reguladora ANATEL, que são as três organizações que denominamos Org1 e Org2 para as operadoras de telefonia, e nó de ordenação O1 para a agência reguladora. Org1 e Org2 precisam de uma comunicação privada e executam transações de portabilidade numérica através do canal C1. O nó P1(Org1) mantém uma cópia do livro-razão (*ledger*) L1 associada a C1. O canal C1 é governado de acordo com as regras da política especificada na configuração de canal CC1 e está sob o controle das organizações Org1, Org2 e O1 Nó de Ordenação (*Orderer*). O serviço de ordem O1, nó de ordenação (*Orderer*), atua como um ponto de administração de rede e usa o canal do sistema. Esse serviço de ordem também suporta o canal C1, para fins de ordem de transações dos blocos para distribuição. Cada uma das três organizações possui uma Autoridade de Certificação.

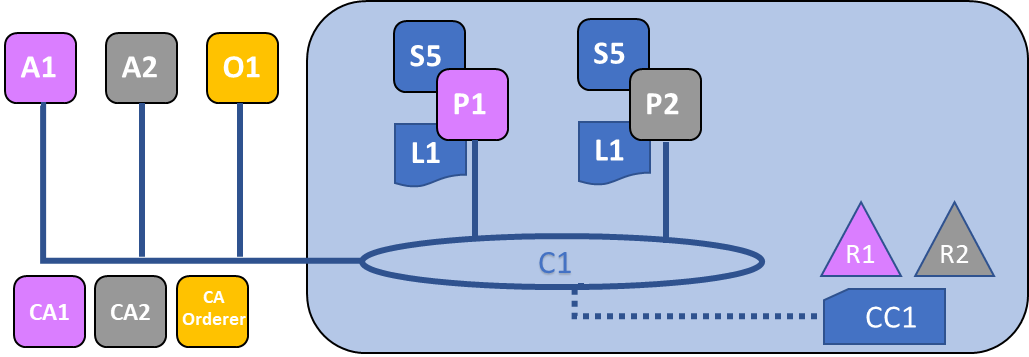


Figura . - Visualização da rede blockchain utilizada neste trabalho.

## Conceitos do Hyperledger Fabric

De acordo com descrição acima, ilustrada na figura 11, alguns conceitos da rede blockchain precisam ser apresentados. São eles:

* **Organizações**

São os membros da rede blockchain e são representadas por um ou mais nós (*peers*), que mantêm cópias locais do livro-razão (*ledger*). São identificadas por meio de certificados digitais, que são usados para autenticação e autorização de acesso à rede através do seu provedor de serviços de associação ([MSP](https://hyperledger-fabric.readthedocs.io/pt/latest/glossary.html#msp)) à rede.

Uma coleção de organizações forma um consórcio. Embora todas as organizações em uma rede sejam membros, nem todas as organizações farão parte de um consórcio.

* **MSP - Provedor de Serviço de Associação (*Membership Service Provider*)**

Como está sendo implementada uma rede privada permissionada, essas organizações precisam provar sua identidade para as outras organizações da rede para que seja autorizada a realizar as transações de portabilidade numérica.

Os MSPs são uma parte fundamental da arquitetura de segurança do Hyperledger Fabric e são usados para garantir que apenas participantes autorizados tenham acesso à rede e possam realizar transações. São responsáveis por gerenciar a identidade e a autenticação dos participantes da rede blockchain. Em outras palavras, os MSPs são responsáveis por criar, emitir e gerenciar os certificados digitais usados para autenticação e autorização dos participantes na rede.

A implementação padrão do MSP no Hyperledger Fabric usa certificados X.509 como identidades, adotando um modelo hierárquico tradicional de Infraestrutura de Chave Pública (PKI) e são localizados em duas partes dessa rede blockchain:

* + MSP local: são definidos para clientes e nós das organizações (*peers*) e definem as permissões para um nó.

 Todo nó deve ter um MSP local que irá definir quem possui direitos administrativos ou participativos.

* + MSP de canal: definem direitos administrativos e participativos no nível do canal. Isso significa que, se uma organização desejar ingressar no canal, um MSP que incorpore a cadeia de confiança dos membros da organização precisará ser incluído na configuração do canal.
* **Canal**:

Um canal é um recurso da rede blockchain privada que permite o isolamento e a confidencialidade dos dados. É uma maneira de criar um subconjunto de participantes em uma rede blockchain que pode se comunicar e realizar transações sem que toda a rede tenha acesso a essas transações ou informações. Isso é útil neste cenário de portabilidade numérica onde ocorrem transações específicas entre duas operadoras apenas.

* **Serviços de Ordem (*Ordering Service*)**

Muitas redes blockchain não são permissionadas (públicas), o que significa que qualquer nó pode participar do processo de consenso. Essas redes blockchain públicas se baseiam em algoritmos de consenso **probabilísticos** que garantem a consistência do livro razão com um alto grau de probabilidade, mas que ainda são vulneráveis a divergências, onde diferentes participantes na rede têm uma visão diferente da ordem das transações aceitas.

Por ser uma rede blockchain privada e permissionada, o Hyperledger Fabric se baseia em algoritmos de consenso determinístico onde qualquer bloco validado pelo nó(*peer*) é garantido como final e correto. Para isso existe o nó chamado ***nó de ordenação*** que garante essa ordem das transações.

Então, os serviços de ordem (***Ordering Service***) são os componentes executados por esse **nó de ordenação** para processar e validar as transações, e garantir o consenso e a ordem correta dos blocos que são adicionados à blockchain.

Caso existam diferentes nós de serviços (ordenação), será necessário implementar serviços de ordem para obter esse consenso. Essas aplicações e seus algoritmos não são objeto de estudo neste trabalho.

* **Identidade**

A identidade refere-se à identificação digital de uma entidade ou participante na rede blockchain, baseada em certificados digitais emitidos por um Membership Service Provider (MSP). Cada participante na rede possui uma identidade composta por um par de chaves criptográficas (chave pública e chave privada) e um certificado digital.

A chave privada é mantida em posse do titular da identidade e é usada para assinar digitalmente as transações e outras interações com a rede. A chave pública é compartilhada com os outros participantes e é usada para verificar a assinatura digital das transações.

O certificado digital, emitido pelo MSP, contém informações sobre a identidade, identificação da organização a que pertence. O certificado digital X.509 é usado no Hyperledger Fabric.

## Instalação do Hyperledger Fabric

Esta prova de conceito utilizou uma estrutura do Hyperledger Fabric para implementar a rede definida na figura 11. Para isso, um modelo de exemplo da plataforma Hyperledger Fabric foi adequado aos requisitos desta prova de conceito. Sua configuração é composta pelos seguintes elementos e suas características:

* A implementação foi feita através dos scripts nativos do Hyperledger Fabric;
* Os nós (peers) são todos implementados em uma máquina virtual;
* São duas organizações (peers), que representam as operadoras de telefonia Org 1 e Org 2, e uma organização de ordenação que representa a Anatel;
* Todos os certificados são emitidos por apenas uma CA (*Certificate Authority*);
* A aplicação Fabric-CA é a autoridade certificadora que emites os certificados X.509.

O *Fabric-CA* é uma autoridade de certificação distribuída que emite certificados X.509, usados para autenticação e autorização dos participantes da rede do Hyperledger Fabric, como já citado anteriormente. É baseado em uma arquitetura de múltiplas autoridades de certificação, o que significa que cada organização pode ter sua própria autoridade de certificação e emitir certificados para seus próprios membros, que é a característica necessária para esta rede blockchain.

## Instalação dos exemplos, executáveis e imagens Docker

Conforme mencionado anteriormente, existe uma vasta documentação de conceitos, instalação e configuração desta plataforma Hyperledger Fabric (Hyperledger Fabric - Key Concepts 2023) que está sendo utilizada como referência e que guiou a todos os passos desta instalação.

Para baixar os arquivos binários e as imagens Docker, é necessário executar o seguinte comando disponibilizado pela Hyperledger.

$ curl -sSL https://bit.ly/2ysbOFE | bash -s

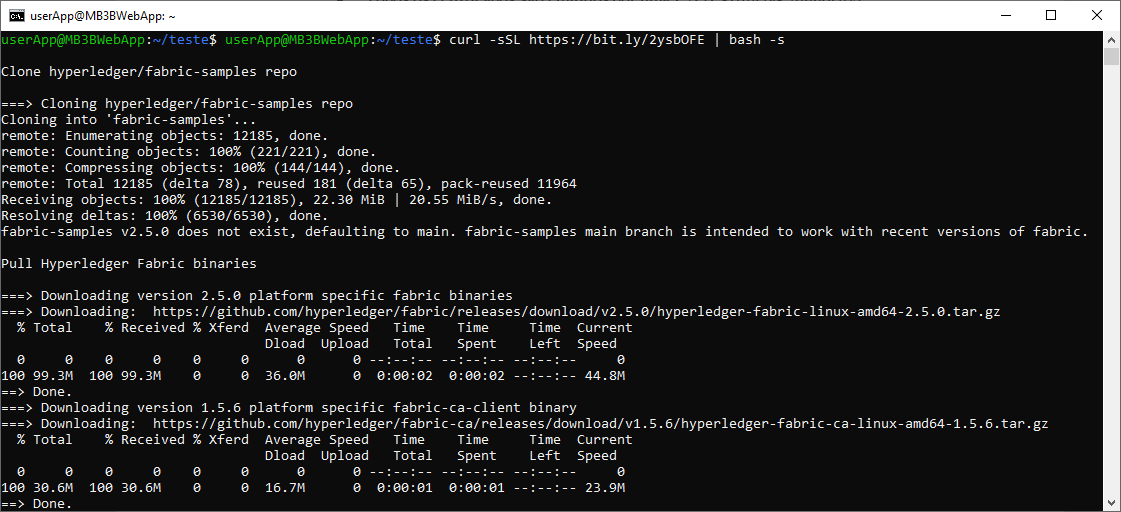


Figura . -Aplicação CURL usada para clonar o repositório fabric-samples.

Este comando executa o script *install-fabric.sh*, que baixa e extrai os arquivos necessários para configuração do Hyperledger Fabric. Este script é disponibilizado no endereço abaixo, acessado em 13/04/2023:

<https://raw.githubusercontent.com/hyperledger/fabric/main/scripts/bootstrap.sh>

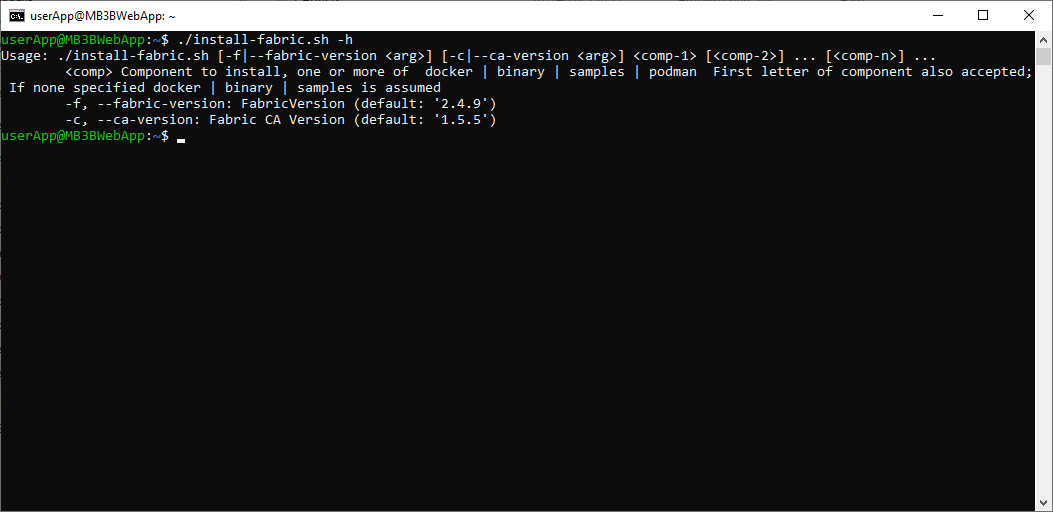


Figura . - Script de instalação do Hyperledger Fabric

Com a conclusão do *download* de todos os arquivos necessários para instalação e configuração do Hyperledger Fabric, os arquivos executáveis são disponibilizados na pasta bin do diretório de trabalho, para que sejam utilizados nos próximos passos:

* configtxgen,
* configtxlator,
* cryptogen,
* discover,
* idemixgen
* orderer,
* peer,
* fabric-ca-client,
* fabric-ca-server.

## Inicializando a rede e os componentes do Hyperledger Fabric

O script network.sh é utilizado como a ferramenta de inicialização de uma instância do Hyperledger Fabric e será analisado nos tópicos abaixo.

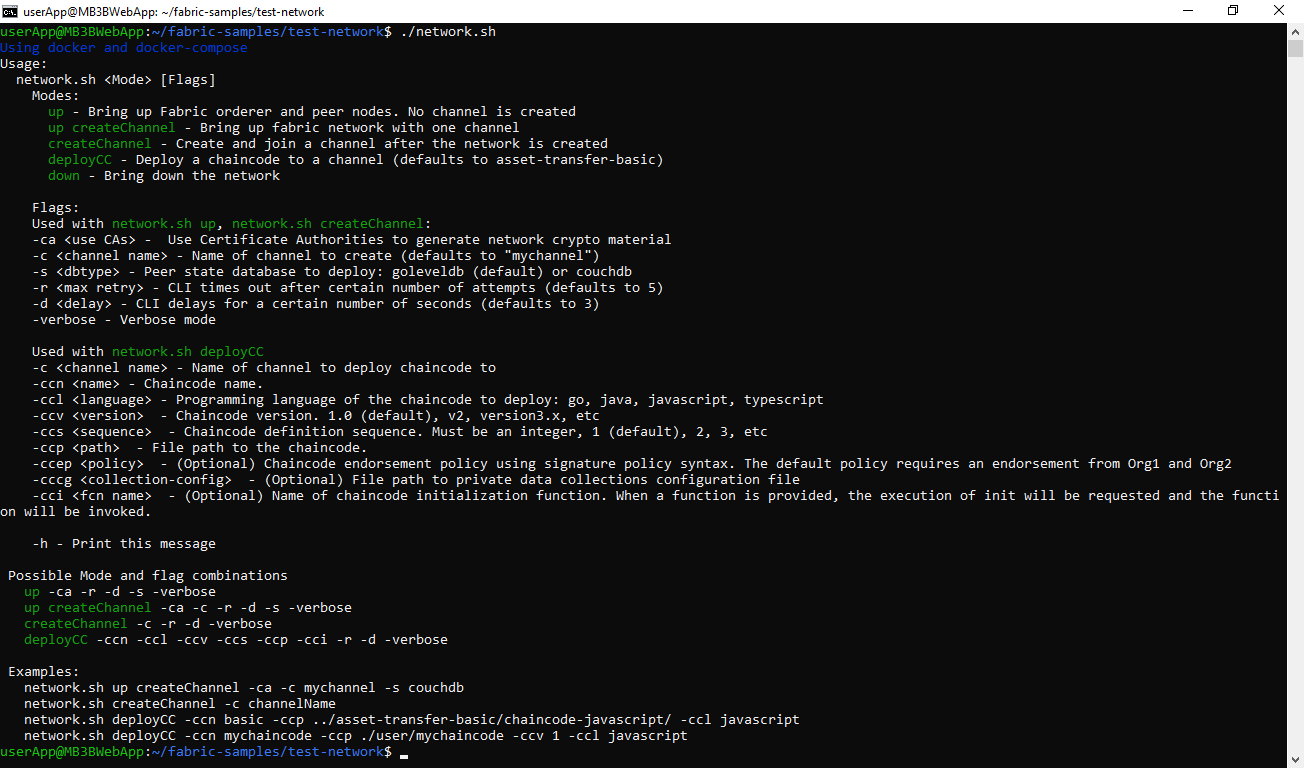


Figura . - Script network.sh e suas opções.

A opção para inicialização da rede utilizando o Fabric-CA como autoridade certificadora é a seguinte:

$ ./network.sh up -ca

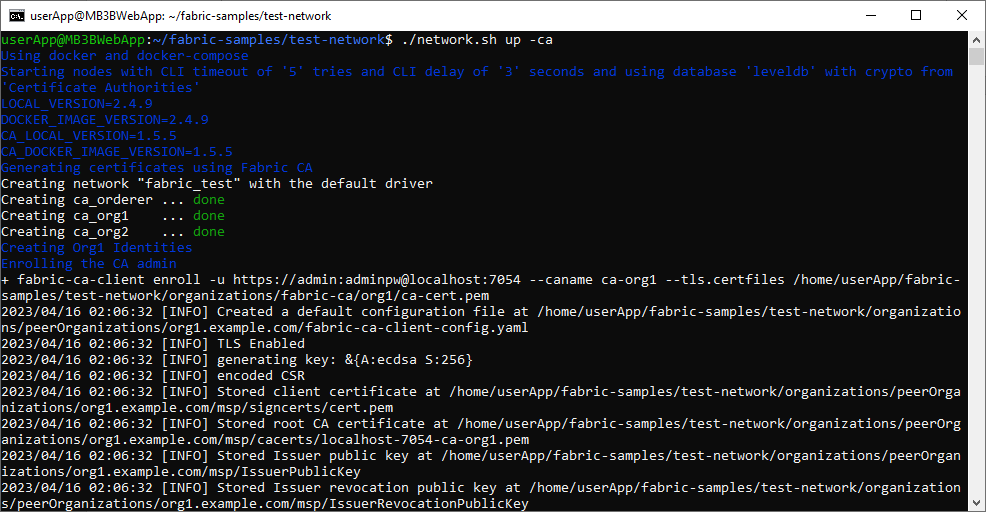


Figura . - Script network.sh com a opção de inicialização da rede

O script *network.sh* com as opções UP e -CA executa as seguintes ações:

**checkPrereqs**: Analisa se houve a clonagem do repositório e todos os binários e arquivos de configuração foram baixados.

**createOrgs**: Como o Hyperledger Fabric é uma rede blockchain permissionada, conforme já descrito anteriormente, cada nó (*peer*) e seus usuários devem usar os certificados e suas chaves para assinatura e validação de suas transações e ações. Com a inclusão da opção “-ca” neste comando, as organizações são criadas e o Fabric-CA é definido como autoridade certificadora e configurado pelo comando *fabric-ca-client*.

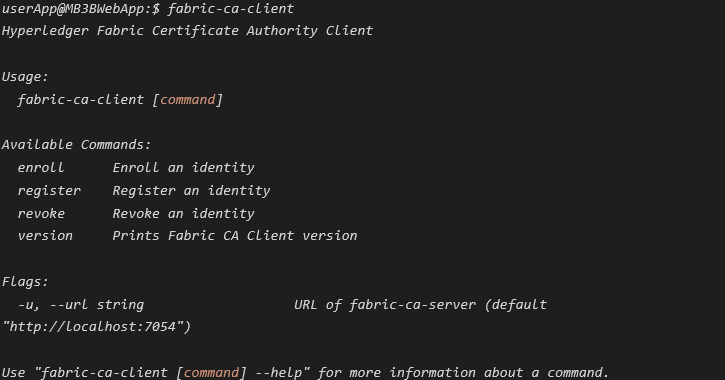


Figura . - comando fabric-ca-client e os prinicipais parâmetro e opções.

Neste script network.sh, o comando *fabric-ca-client* é utilizado nos 8 passos para criação das chaves pública e privada e os certificados:

1. Criar um certificado e registrar o usuário admin e sua senha adminpw da organização Org1, que representa uma operadora de telefonia neste trabalho:

***fabric-ca-client enroll*** *-u --caname --tls.certfiles*

Onde:

* *enroll* – registro e obtenção de certificados do *ca-org1*;
* *-u* [*https://admin:adminpw@localhost:7054*](https://admin:adminpw@localhost:7054)**–** endereço para autenticação das chaves com os usuário *admin* e senha *adminpw.*
* --caname ca-org1 – nome da autoridade certificadora
* --tls.certfiles "/organizations/fabric-ca/org1/ca-cert.pem" – caminho do arquivo que contém o certificado TLS (Transport Layer Security) do servidor de Autoridade de Certificação (CA) ca-org1.

Sobre o TLS (Transport Layer Security), é um protocolo de segurança utilizado para proteger a comunicação entre os componentes de uma rede, garantindo a confidencialidade, integridade e autenticidade dos dados transmitidos (Rescorla 2018). Cada nó (peer) da rede tem um certificado TLS que é usado para autenticar e estabelecer uma conexão segura com outros nós na rede. Os certificados raiz são usados ​​para verificar a autenticidade desses certificados TLS e garantir que a conexão segura seja estabelecida com o nó correto.

A partir deste comando é possível identificar as chaves pública e privada para a Org1.

**Public Key -** arquivo com a seguinte localização:

$~/test-network/organizations/peerOrganizations/ org1.example.com/msp/signcerts/cert.pem

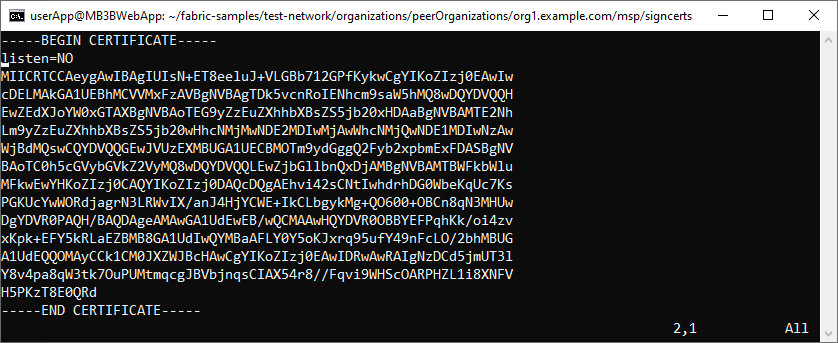


Figura . - Conteúdo ao arquivo cert.perm - chave pública da Org1.

**Private Key: -** arquivo com a seguinte localização:

$~/test-network/organizations/peerOrganizations/org1.example.com/msp/cacerts/localhost-7054-ca-org1.pem

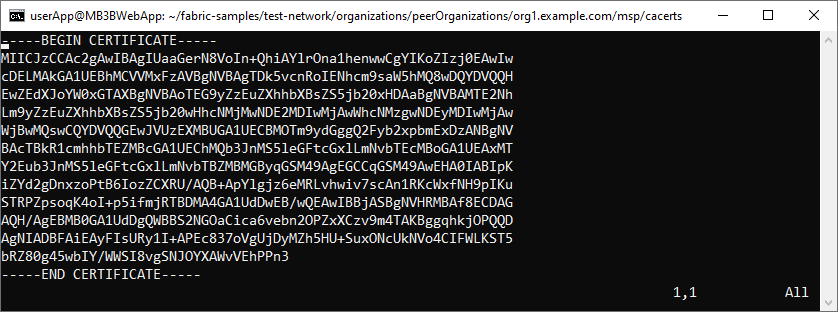


Figura . - Conteúdo ao arquivo localhost-7054-ca-org1.pem - chave privada da Org1

1. Registrar o nó (peer) como identidade.

***fabric-ca-client register*** *--caname --id.name --id.secret --id.type --tls.certfiles*

Onde:

* *register* - registra da identidade *ca-org1*;
* --caname ca-org1 – nome da autoridade certificadora
* --id.name peer0 – identificação do nó (peer) peer0, representando a Org1.
* --id.secret *peer0pw* – senha do certificado
* --id.type ***peer*** – tipo da identidade que está sendo registrada. Podendo ser:
  + client (padrão) - Para registrar uma identidade de usuário. As identidades de usuário são usadas por clientes para interagir com a rede do Hyperledger Fabric.
  + "peer": Para registrar uma identidade de nó ("peer").
  + "orderer": Para registrar uma identidade de nó "orderer" (ordenação) que é responsável por receber, ordenar e validar transações em um canal específico;
  + "admin": Para registrar uma identidade de administrador, que é uma identidade privilegiada com acesso a recursos e operações de gerenciamento de rede.
* --tls.certfiles "/organizations/fabric-ca/org1/ca-cert.pem" – caminho do arquivo que contém o certificado TLS (Transport Layer Security) do servidor de Autoridade de Certificação (CA) ca-org1.

1. Registrar o usuário.

***fabric-ca-client register*** *--caname --id.name --id.secret --id.type --tls.certfiles*

Onde:

* *register* - registra o usuário use1 na *ca-org1*;
* --id.name user1 – identificação do usuário
* --id.secret *peer0pw* – senha do certificado
* --id.type **client** – tipo da identidade que está sendo registrada.

1. Registrar o usuário ADMIN.

***fabric-ca-client register*** *--caname --id.name --id.secret --id.type --tls.certfiles*

Onde:

* *register* - registra o usuário use1 na *ca-org1*;
* *--id.name org1admin* – identificação do usuário Admin
* --id.secret *org1adminpw* – senha do usuário ADMIN
* --id.type **admin** – tipo da identidade que está sendo registrada.

1. Geração do MSP para o nó (*peer*).

***fabric-ca-client enroll*** *-u –caname -M --csr.hosts --tls.certfiles*

Onde:

* *enroll* – registro e obtenção de certificados do *MSP*
* *-u* [*https://peer0:peer0pw@localhost:7054*](https://peer0:peer0pw@localhost:7054)**–** endereço para autenticação das chaves com os usuário *peer0* e senha *peer0pw.*
* --*caname* ca-org1 – nome da autoridade certificadora
* *-M* – diretório do Membership Service Provider
  + *"/organizations/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/msp"*
* --csr.hosts - especificar os nomes de host (FQDNs) que devem ser incluídos no pedido de assinatura de certificado (CSR - Certificate Signing Request) gerado durante o processo de inscrição (*enrollment*) da identidade na CA ca-org1.
  + *peer0.org1.example.com*

Sobre FQDN, significa Fully Qualified Domain Name, que é um nome de domínio totalmente qualificado e único e que identifica um host (nó) em uma rede de computadores, de forma completa e sem ambiguidades (Mockapetris 1987).

1. Geração certificado TLS para o nó (peer).

***fabric-ca-client enroll*** *-u --caname -M --enrollment.profile --csr.hosts --csr.hosts --tls.certfiles*

Onde:

* *enroll* – registro e obtenção de certificados do *MSP*
* *-u* [*https://peer0:peer0pw@localhost:7054*](https://peer0:peer0pw@localhost:7054)**–** endereço para autenticação das chaves com os usuário *peer0* e senha *peer0pw.*
* --*caname* ca-org1 – nome da autoridade certificadora
* *-M* – diretório do Membership Service Provider
  + "/organizations/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/tls"
* *--enrollment.profile tls* - especifica o perfil de inscrição que será utilizado ao inscrever a identidade na CA ca-org1.
* --csr.hosts - especificar os nomes de host (FQDNs) que devem ser incluídos no pedido de assinatura de certificado (CSR - Certificate Signing Request) gerado durante o processo de inscrição (enrollment) da identidade na CA ca-org1.
  + *peer0.org1.example.com*
* --tls.certfiles "/organizations/fabric-ca/org1/ca-cert.pem" – caminho do arquivo que contém o certificado TLS (Transport Layer Security) do servidor de Autoridade de Certificação (CA) ca-org1.

1. Geração do usuário MSP.

***fabric-ca-client enroll*** *-u --caname -M --tls.certfiles*

Onde:

* *enroll* – registro e obtenção de certificado do usuário user1 no *MSP;*
* *-u* [*https://user1:user1pw@localhost:7054*](https://user1:user1pw@localhost:7054)**–** endereço para autenticação das chaves com os usuário *user1* e senha *user1pw.*
* --*caname* ca-org1 – nome da autoridade certificadora
* *-M* – diretório do Membership Service Provider
  + [/organizations/peerOrganizations/org1.example.com/users/User1@org1.example.com/msp](mailto:$%7bPWD%7d/organizations/peerOrganizations/org1.example.com/users/User1@org1.example.com/msp)

1. Geração do usuário ADMIN MSP.

***fabric-ca-client enroll*** *-u --caname -M --tls.certfiles*

Onde:

* *enroll* – registro e obtenção de certificado do usuário admin no *MSP*
* *-u* <https://org1admin:org1adminpw@localhost:7054> **–** endereço para autenticação das chaves com os usuário *org1admin* e senha *org1adminpw.*
* --*caname* ca-org1 – nome da autoridade certificadora
* *-M* – diretório do Membership Service Provider
  + [/organizations/peerOrganizations/org1.example.com/users/User1@org1.example.com/msp](mailto:$%7bPWD%7d/organizations/peerOrganizations/org1.example.com/users/User1@org1.example.com/msp)

Na criação do nó de ordenação (*orderer*), na etapa de registro do nó de ordenação (*orderer*) é necessária a seguinte alteração:

***fabric-ca-client register*** *--caname --id.name --id.secret --id.type --tls.certfiles*

Onde:

* *register* - registra da identidade *ca-orderer*;
* --caname *ca-orderer* – nome da autoridade certificadora
* --*id.name* **orderer** – identificação do nó de ordenação(*orderer*).
* --id.secret *ordererpw* – senha do certificado
* --id.type ***peer*** – tipo da identidade que está sendo registrada.
  + "orderer": Para registrar uma identidade de nó "orderer" (ordenação) que é responsável por receber, ordenar e validar transações em um canal específico;
* --*tls.certfiles* "/organizations/fabric-ca/ordererOrg/ca-cert.pem" – caminho do arquivo que contém o certificado TLS (Transport Layer Security) do servidor de Autoridade de Certificação (CA) *ca-orderer*.

Além da criação das organizações Org1, Org2 e Orderer, é executado o script "./organizations/ccp-generate.sh", usado para gerar arquivos de perfil de conexão de cliente (Client Connection Profile - CCP) e contém os detalhes como endereços de nós (peers), portas, certificados e chaves que são necessários para se comunicar com a rede.

Um template gerado por esse script é o ccp-template.yaml.



Figura . - Perfil de conexão de cliente (Client Connection Profile - CCP)

Com a estrutura das organizações criadas, o script *network.sh* irá carregar pelo Docker, os nós (*peers* e *orderer*) através do arquivo de configuração *docker-compose-test-net.yaml*. Neste arquivo estão os detalhes de configuração dos containers *orderer.example.com*, *peer0.org1.example.com* e *peer0.org2.example.com*,e ainda um container chamado CLI utilizado para desenvolvimento, depuração e testes, uma vez que contém a interface de linha de comando (*Command Line Interface*) do Hyperledger Fabric.

Além dos containers, o Docker cria as redes conforme a configuração definida neste arquivo *docker-compose-test-net.yaml.* Como a faixa de IP não foi definida neste arquivo, o Docker utiliza sua faixa de IP padrão que é:

Fabric-test subnet: 172.19.0.0/16 [ATUALIZAR TABELA DE IPs]

|  |  |  |
| --- | --- | --- |
| Container | IPV4 subnet | MacAddress |
| ca\_org2 | 172.19.0.2/16 | 02:42:ac:13:00:02 |
| ca\_org1 | 172.19.0.3/16 | 02:42:ac:13:00:03 |
| ca\_orderer | 172.19.0.4/16 | 02:42:ac:13:00:04 |
| peer0.org2.example.com | 172.19.0.5/16 | 02:42:ac:13:00:05 |
| peer0.org1.example.com | 172.19.0.6/16 | 02:42:ac:13:00:06 |
| orderer.example.com | 172.19.0.7/16 | 02:42:ac:13:00:07 |
| cli | 172.19.0.8/16 | 02:42:ac:13:00:08 |

Tabela . - Faixa de endereços IP definido automaticamente pelo Docker.

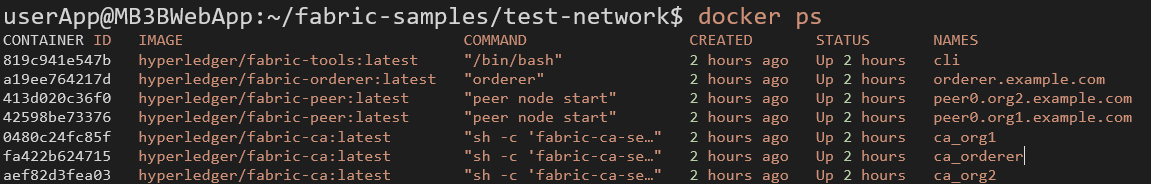


Figura . - Resposta do comando mostrando os containers criados para os nós.

## Criando o canal

Conforme apresentado anteriormente, um canal é uma "sub-rede" privada de comunicação entre dois ou mais membros específicos com o objetivo de realizar transações privadas, onde cada parte deve ser autenticada e autorizada para executar essas transações. Cada novo nó que se junta a um canal tem sua própria identidade fornecida pelo provedor de serviços de associação (MSP), que autentica cada nó aos seus pares e serviços do canal.

Os canais são criados através da geração de uma transação de criação de canal em um bloco inicial (bloco de gênese). A transação de criação de canal especifica a configuração inicial do canal através do comando **configtxgen**. Esse comando lê o arquivo ***configtx.yaml*** que define a configuração do canal.

Essas ações também estão sequenciadas no script ***network.sh*** e, como nesta prova de conceito será criado o canal *MyChannel*, será aciona com os seguintes:

$ ***./network.sh*** createChannel -c mychannel -s couchdb

Onde:

-c mychannel – Nome do canal que está sendo criado.

-s couchdb – base de dados para armazenamento do estado global (*Word State*): *goleveldb* ou *couchdb.*

O estado global (*Word State*) é o estado atual do bloco, ou os valores mais recentes das chaves nessa cadeia de blocos. Esse estado é armazenado em um banco de dados que contém o valor do conjunto “chave-valor” que foram adicionados, modificados ou excluídos pelo conjunto de transações validadas e confirmadas na blockchain.

## Criando o bloco de gênese

O bloco de gênese é o primeiro bloco em uma blockchain, criado quando um novo canal é estabelecido na rede. Este bloco contém informações de configuração, como a lista de membros (organizações) que fazem parte desse canal, os nós âncora (*Anchor Peer*) para cada membro, políticas de acesso e outros parâmetros necessários para validação das transações e autenticações. A partir deste bloco de gênese, outros blocos serão adicionados à cadeia, cada um contendo as suas respectivas transações.

A criação do bloco gêneses é feito através do comando ***configtxgen*** com a seguinte sintaxe**:**

***configtxgen*** *-profile**-outputBlock -channelID*

Onde:

* -*profile* *TwoOrgsApplicationGenesis* - perfil de configuração definido no arquivo ***configtx.yaml*** que inclui informações das organizações que fazem parte da rede, os canais existentes e as políticas de acesso.
* -*outputBlock* ./channel-artifacts/MyChannel.block - caminho e o nome do arquivo que será gerado como o bloco de gênese da rede.
* -*channelID* Mychannel – identifica o canal para o qual a transação será gerada.

Nós âncora (*Anchor Peer*) são utilizados pelo protocolo ***Gossip*** para garantir que os pares de diferentes organizações se conheçam.

Em um exemplo de estrutura de com 3 organizações A, B e C, com seus respectivos nós peer0 e peer1, em um mesmo canal e tendo o nó **peer0**.orgC como o único um ponto de ancoragem (Nó Âncora – *Anchor Peer*), quando **peer0**.orgA ( nó da organização A ) contactar **peer0**.orgC, este irá informar sobre **peer1**.orgA. Quando em um outro momento **peer1**.orgB contactar **peer0**.orgC este irá informá-lo sobre **peer1**.orgA e as organizações A e B poderão negociar transações sem assistência do **peer0**.orgC.

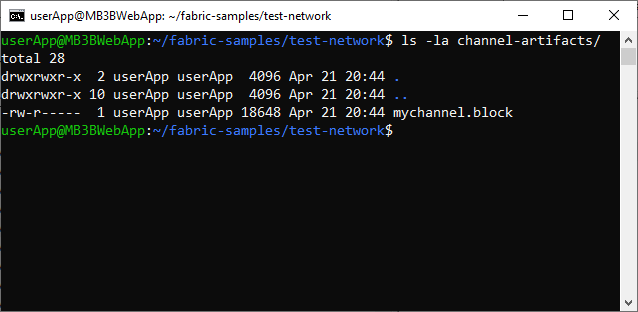


Figura . - Bloco gênese criado como mychannel.block

A partir do bloco gênese será criado o canal através do comando abaixo:

***osnadmin channel join*** *--channelID --config-block -o --ca-file --client-cert --client-key*

Onde:

* *channel join* – inclui *Ordering Service Node* (OSN), os nós de ordenação (*orderer*) no canal.
* *--config-block* ./channel-artifacts/mychannel.block - caminho e o nome do arquivo que será gerado como o bloco de gênese da rede, onde contém um bloco de configuração para este canal.
* *-o* localhost:7053 – endereço do nó de ordenação (*orderer*).
* *--ca-file* ./organizations/ordererOrganizations/example.com/tlsca/tlsca.example.com-cert.pem
* *--client-cert* ./organizations/ordererOrganizations/example.com/orderers/orderer.example

.com/tls/server.crt

* *--client-key* ./organizations/ordererOrganizations/example.com/orderers/orderer.example

.com/tls/server.key

O comando ***osnadmin channel*** é a simplificação de *Orderer Service Node (OSN) administration*, e permite que operações relacionadas com os canais em um nó de ordenação (*orderer*) sejam executadas, como adicionar um nó a um canal, listar os canais aos quais um nó de ordenação(orderer) está conectado e remover um canal.

Este comando possui a opção de ***osnadmin channel list***, para listar o canal criado e confirmar o seu estado.

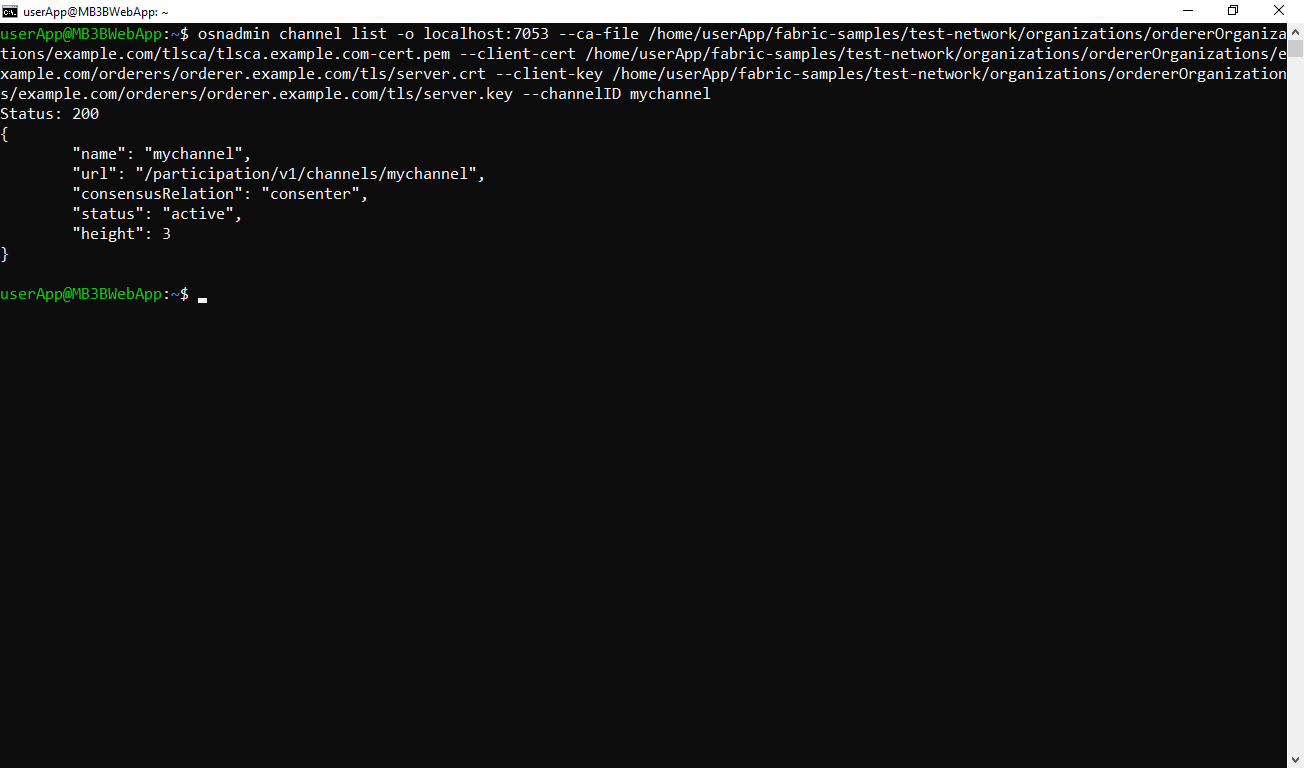


Figura . - Estado do canal **mychannel**

## Inserir os nós no canal

Após a criação do bloco gênese e a criação do canal, é necessário inserir os nós neste canal, e isso é feito através do comando ***peer***.

Este comando ***peer***, possui cinco diferentes subcomandos que permitem executar tarefas nos nós e serão abordados conforme sua utilização.

Antes da execução do comando ***peer channel*** é necessário que as variáveis de ambiente sejam atualizadas conforme o nó que será incluído no canal.

* *CORE\_PEER\_LOCALMSPID* – identificação do MSP.
* *CORE\_PEER\_TLS\_ROOTCERT\_FILE* - arquivos que contêm os certificados raiz usados para estabelecer conexões seguras (TLS) entre os componentes da rede.
* *CORE\_PEER\_MSPCONFIGPATH* - diretório que contém o arquivo de configuração do MSP (Membership Service Provider) para o nó (peer) em execução.
* *CORE\_PEER\_ADDRESS* – Endereço e porta do nó.

***peer channel join*** *-b ./channel-artifacts/mychannel.block*

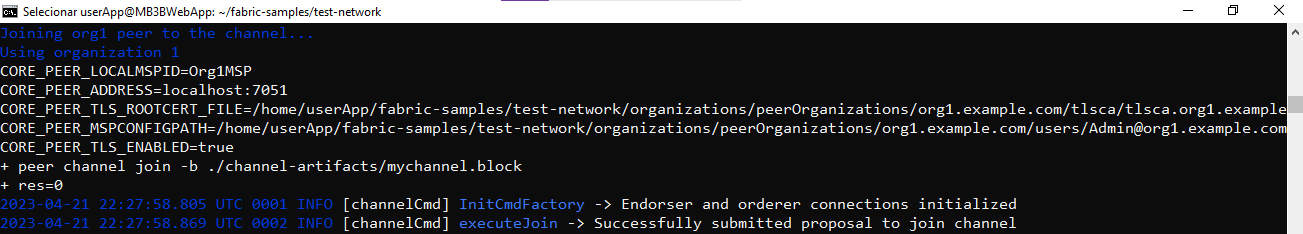


Figura . - Comando peer channel Join executado para Org1.

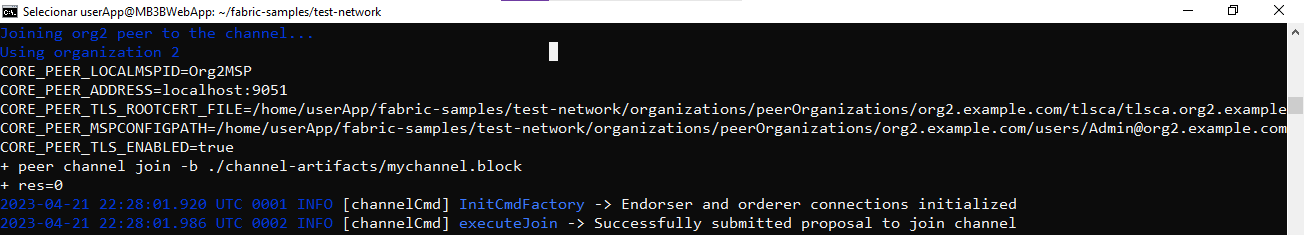


Figura . - Comando peer channel join executado para Org2.

## Selecionando os nós âncora (anchor peer)

Após as organizações terem conectados seus nós (*peers*) ao canal, é preciso selecionar um de seus nós(*peers*) para se tornar um nó âncora(*anchor peer*). Conforme já apresentado anteriormente, eles são necessários para otimizar consultas entre os nós.

O primeiro passo é coletar o bloco de configuração mais recente através do comando ***peer channel fetch***. Este comando recupera a informação de um bloco e salva em um arquivo.

***peer channel fetch config*** *config\_block.pb -o --ordererTLSHostnameOverride -c --tls --cafile*

Onde:

* *channel fetch* – recupera informação de um bloco específico e armazena em um arquivo chamado ***config\_block.pb***.
* *ordererTLSHostnameOverride* orderer.example.com – hostname usado para validar a conexão TLS com o nó de ordenação (*orderer*).

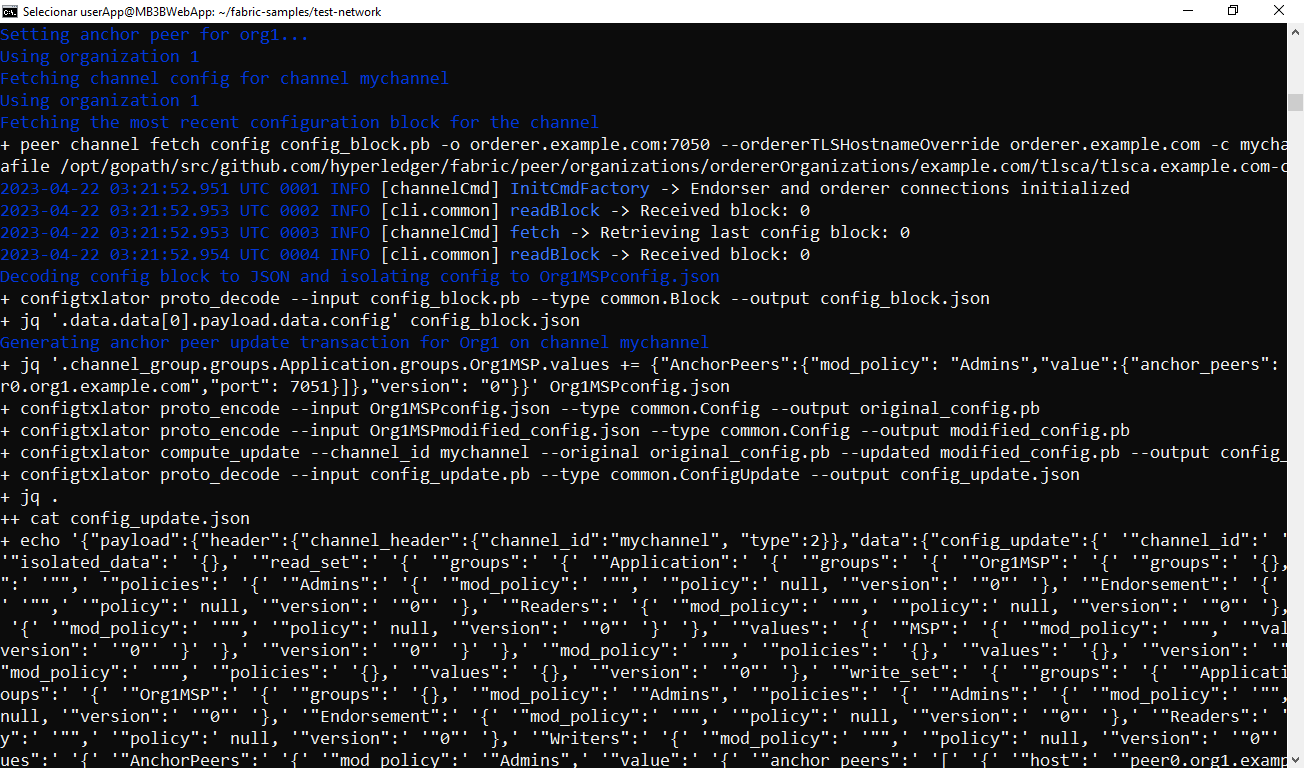


Figura .- Comando peer channel fetch executado em Org1.

Como o bloco de configuração mais recente é o bloco gêneses, a resposta desse comando aparece como “received block: 0”.

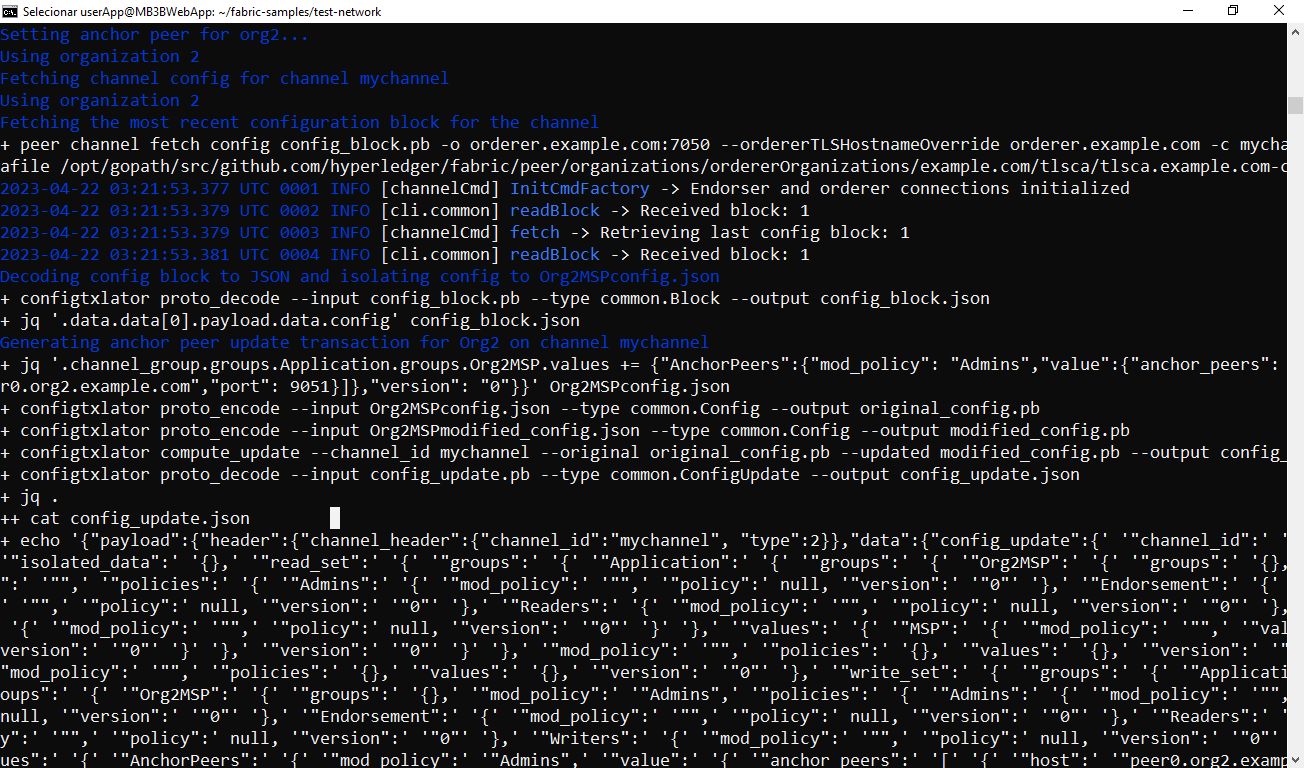


Figura . - Comando peer channel fetch executado em Org2.

Protobuf (Protocol Buffers) é um mecanismo de serialização de dados desenvolvido pelo Google, que é utilizado pelo Hyperledger Fabric para representar dados em formato binário. É uma forma eficiente e compacta de armazenar e transmitir dados estruturados, permitindo que esses dados sejam facilmente lidos e processados por diferentes plataformas e linguagens de programação (Google - Protobuf 2023).

O comando ***configtxlator*** permite a tradução do formato protobuf em um objeto JSON que possa ser lido e editado. O arquivo gerado pelo comando acima foi nomeado ***config\_block.pb*** e está no formato protobuf. O primeiro passo é decodifica-lo em um objeto.

***configtxlator proto\_decode*** *--input --type --output*

Onde:

* *proto\_decode* – converte uma mensagem protobuf para JSON.
* *--input* config\_block.pb *–* o arquivo contendo o formato protobuf.
* *--type* common.Block - especifica o tipo da mensagem protobuf a ser decodificada. Isso é necessário porque o protobuf pode ser usado para representar diferentes tipos de mensagens. O tipo deve ser especificado no formato <package>.<message>. Por exemplo: common.Block ou common.Envelope.
* *--output* config\_block.json – arquivo destino onde será escrito o JSON.

O novo arquivo JSON ***config\_block.json*** será tratado através do comando JQ que é um processador de JSON. Nesta linha de comando, é extraído apenas o bloco de configuração do canal.

***jq*** *.data.data[0].payload.data.config config\_block.json*

A edição desse arquivo é feita através deste comando JQ de processamento do JSON e os nós são incluídos como nós âncora deste canal.

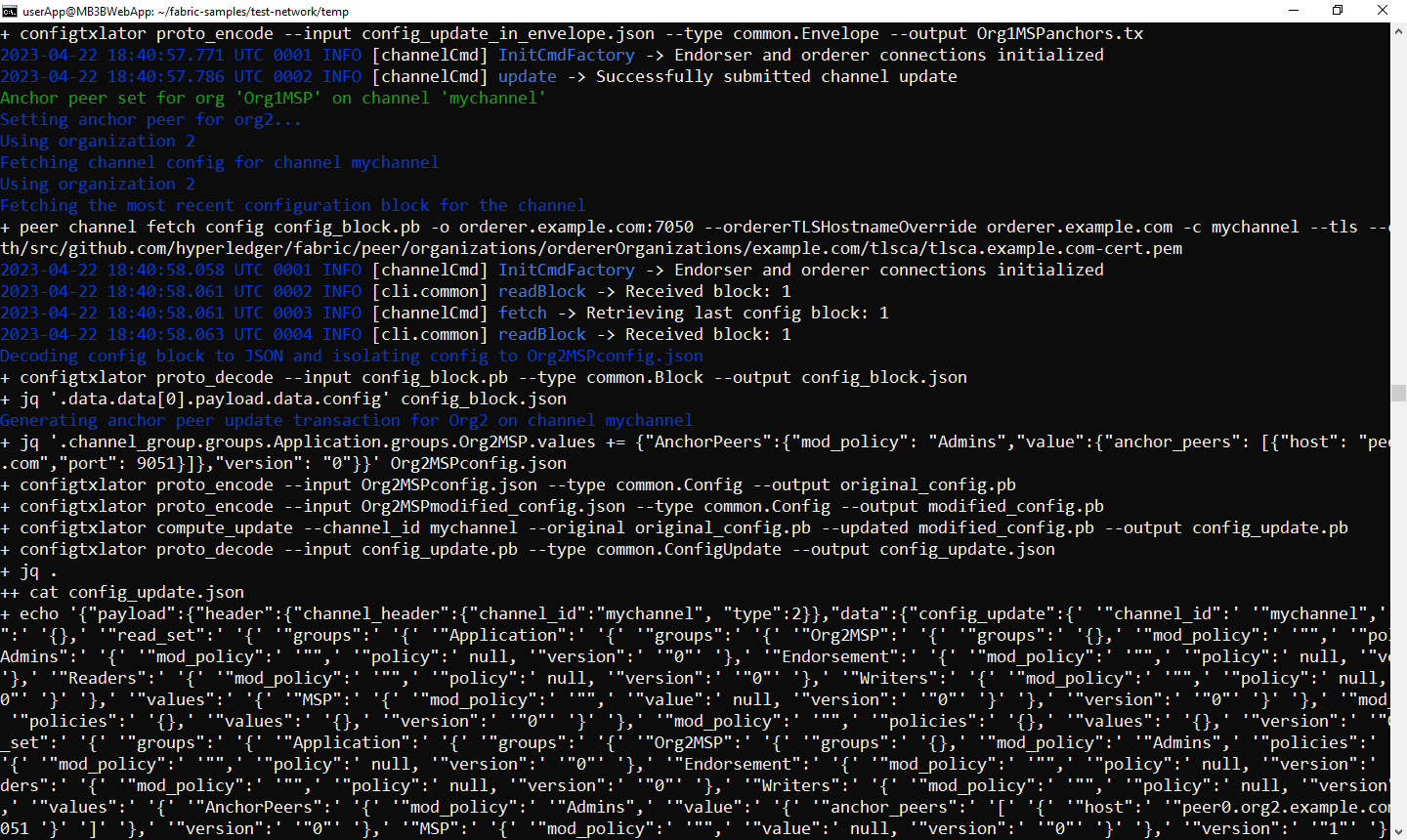


Figura . - Confirmação do nó Org1MSP incluído como nó âncora.

O mesmo processo ocorre para o nó Org2MSP.

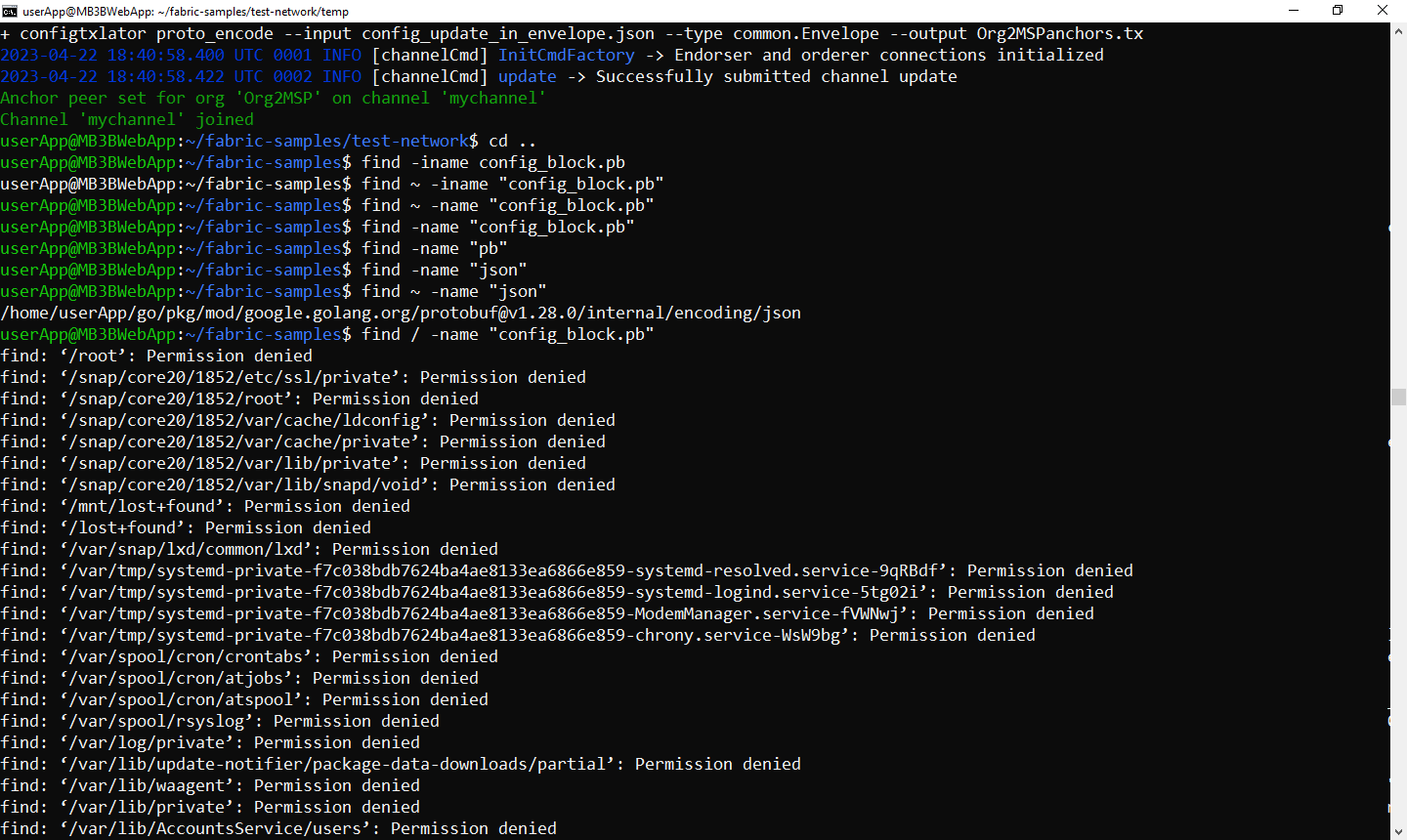


Figura . - Confirmação do nó Org2MSP incluído como nó âncora.

## Smart Contracts

Como este trabalho foi construído baseado em uma prova de conceito, os *smart contracts* são o principal ponto de atuação. Todos os outros tópicos citados representam configuração e ajustes de estruturas existentes. Nos *smart contracts* as regras de negócio da portabilidade numérica foram implementadas e todos os conceitos se materializaram em códigos na linguagem Javascript.

Neste tópico serão descritos os smart contracts e a forma como são tratados pela plataforma Hyperledger Fabric. Além disso também serão apresentados os códigos e o detalhamento da forma como estão implementados, e no capítulo seguinte os testes executados, resultados obtidos e as possibilidades de novos trabalhos.

## Smart Contracts no Hyperledger Fabric

Os usuários interagem com a estrutura do livro razão da blockchain através dos *smart contracts*. No Hyperledger Fabric, os *smart contracts* são chamados de ***chaincode***. Organizações que pretendem validar as transações ou consultar o livro razão precisam instalar um chaincode em seus nós (*peers*). Depois que um chaincode foi instalado nos nós (*peers*) que se juntaram a um canal, os membros do canal podem implantar o chaincode no canal e usar os smart contracts no chaincode para criar ou atualizar ativos no livro razão do canal.

Um chaincode é implantado em um canal usando um processo conhecido como o ciclo de vida do chaincode Fabric. O ciclo de vida do chaincode Fabric permite que várias organizações concordem com a forma como um chaincode será manipulado antes que ele possa ser usado para criar transações.

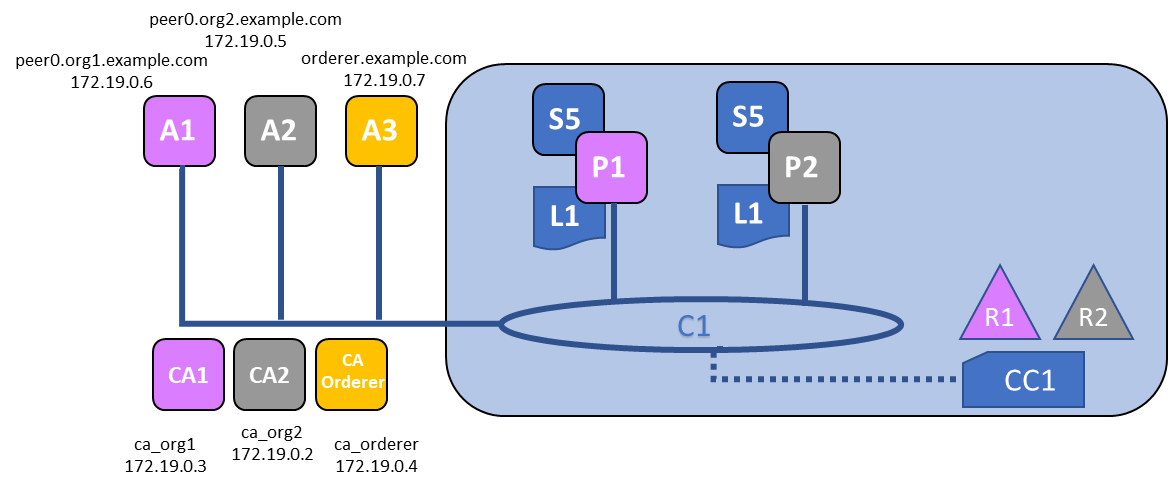


Figura . - Revisão da rede blockchain utilizada neste trabalho.

A partir da revisão da figura 11, é possível seguir os próximos passos de implementação do chaincode referenciando aos elementos desta rede blockchain.

Conforme descrito nos tópicos anteriores, o script network.sh possui as rotinas de criação de cada um dos artefatos necessários para instalação e configuração do Hyperledger Fabric e nesta etapa de implementação dos chaincodes será tratado da mesma maneira.

*Chaincode lifecycle* é um processo na plataforma Hyperledger Fabric para que diferentes organizações gerenciem os chaincodes conforme os acordos estabelecidos. Dentro deste *chaincode lifecycle* a implantação dos chaincodes deve ter as etapas de empacotamento, instalação do chaincode através do pacote, aprovação da definição do chaincode pelas organizações, confirmação da definição do chaincode no canal (*commit*) e a invocação para execução das transações.

Conforme explicado anteriormente, o script *network.sh* é utilizado como a ferramenta de inicialização de uma instância do Hyperledger Fabric e possui diversas funções conforme a etapa de implementação desta plataforma. Nesta etapa de implantação dos chaincode, será utilizada a função deployCC, e que possui os parâmetros de entrada para todos os comandos descritos nos próximos passos.

***..../network.sh deployCC*** *-ccn -ccp -ccl*

Onde:

* -*ccn* basic – nome da aplicação.
* *-ccp* ../asset-transfer-basic/chaincode-javascript/ *–* indica a localização dos arquivos contendo os códigos na linguagem de programação definida no parâmetro
* -*ccl* javascript – linguagem em que foi implementado o chaincode.

## Packing ID - Empacotamento

O empacotamento do chaincode no Hyperledger Fabric é feito para permitir a instalação e a aprovação do chaincode pelos membros de uma rede, garantindo que o código esteja em conformidade com as políticas de segurança e governança estabelecidas pela organização. Uma vez que um chaincode é empacotado, ele recebe um identificador "Package ID" que é usado para associar o chaincode instalado em um nó (*peer*) garantindo que as transações estão sendo executadas pelo código do chaincode negociado.

O comando que será utilizado para tratar o empacotamento do chaincode é o ***peer lifecycle chaincode*** e será tratado neste trabalho dentro das variações encontradas no script ./***network.sh***.

***peer lifecycle chaincode*** *package basic.tar.gz –path --lang --label*

Onde:

* *peer lifecycle chaincode package* – permite o acessar *chaincode lifecycle* para empacotar um chaincode, instalá-lo em seus peers, aprovar uma definição de chaincode para sua organização e, em seguida, carregar a definição em um canal.
* *package* basic.tar.gz *–* arquivo de saída onde o chaincode será empacotado no formato tar.gz.
* *--path* ../asset-transfer-basic/chaincode-javascript/ *–* indica a localização dos arquivos contendo os códigos na linguagem de programação definida no parâmetro --lang.
* *--lang* node– definição da linguagem de programação que será utilizada na implementação do chaincode. Node é o valor que representa que será desenvolvido o chaincode em Javascript.
* *--label* basic\_1 – rótulo como descrição do pacote.

Para definição do Package ID do pacote resultante do comando acima é utilizado o seguinte comando:

***peer lifecycle chaincode calculatepackageid*** *basic.tar.gz*

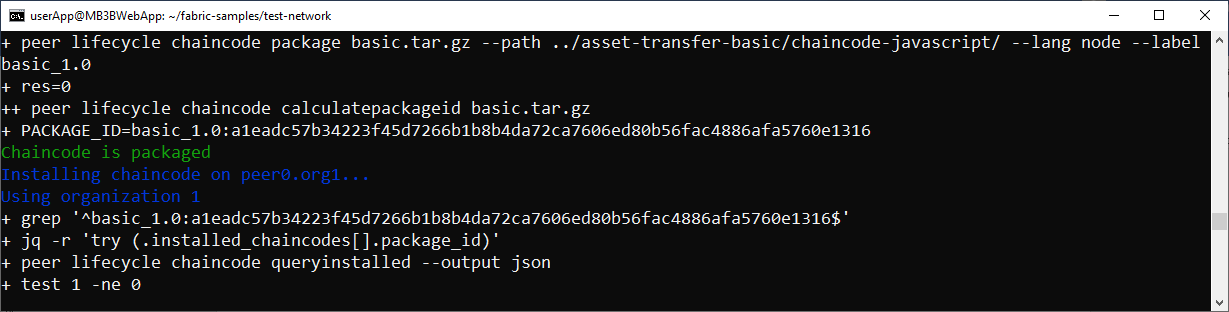


Figura . - Empacotamento e definição do Package ID

## Instalação do chaincode em cada nó

Após o empacotamento do chaincode e geração do Package ID, é necessário instalar o chaincode em todos os nós (*peers*). Como está definida a política de aprovação para exigir aprovações de ambas as Org1 e Org2, precisamos instalar o chaincode nos nós operados por ambas as organizações:

peer0.org1.example.com

peer0.org2.example.com

Neste trabalho, como está sendo utilizado uma máquina virtual para todas organizações, a instalação do chaincode será a partir de alteração de variáveis que identificam em qual organização o comando de instalação está sendo executado.

Para organização Org1, temos os seguintes comandos que afetam variável de ambiente para o nó *peer0.org1.example.com*:

* *CORE\_PEER\_LOCALMSPID* – identificação do MSP.
* *CORE\_PEER\_TLS\_ROOTCERT\_FILE* - arquivos que contêm os certificados raiz usados para estabelecer conexões seguras (TLS) entre os componentes da rede.
* *CORE\_PEER\_MSPCONFIGPATH* - diretório que contém o arquivo de configuração do MSP (Membership Service Provider) para o nó (peer) em execução.
* *CORE\_PEER\_ADDRESS* – Endereço e porta do nó.

Com essas variáveis configuradas a instalação do chaincode é executada através do comando *peer lifecycle chaincode install*:

***peer lifecycle chaincode*** *install basic.tar.gz*

Onde:

* *Basic.tar.gz* – arquivo gerado no empacotamento do chaincode, conforme descrito anteriormente.

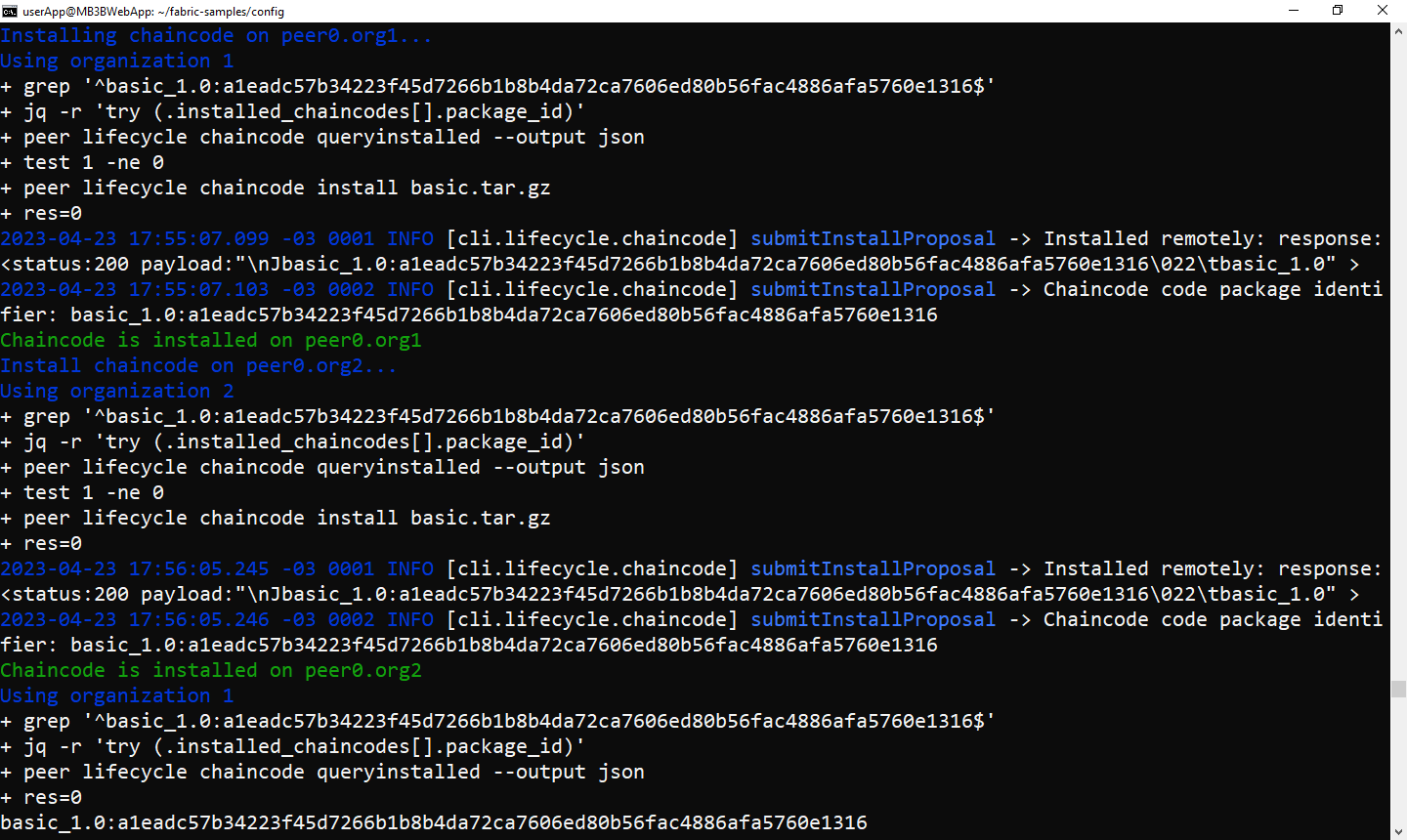


Figura . - Instalação do chaincode em peer0.org1.example.com

## Aprovação das definições do chaincode

Após instalação do pacote chaincode, é necessário aprovar uma definição de chaincode para a organização. A definição inclui parâmetros de governança de chaincode, como o nome, versão e a política de endosso de chaincode.

O comando utilizado para aprovação das definições do chaincode é o *peer lifecycle chaincode approveformyorg*.

***peer lifecycle chaincode approveformyorg*** *-o –tls --cafile --channelID --name --version --package-id --sequence*

Onde:

* *-o* localhost:7053 – endereço do nó de ordenação (*orderer*).
* *ordererTLSHostnameOverride* orderer.example.com – hostname usado para validar a conexão TLS com o nó de ordenação (*orderer*).

É possível fornecer dois argumentos *--signature-policy* ou *--channel-config-policy* para esse comando ***peer lifecycle chaincode approveformyorg*** para especificar uma política de endosso(*endorsement*) de chaincode. A política de endosso especifica quantos nós do canal precisam validar uma transação de um determinado chaincode. A definição do chaincode desta prova de conceito usará a política de endosso padrão, que exige que uma transação seja endossada pela maioria dos membros do canal presentes quando a transação é enviada. Isso implica que caso novas organizações sejam adicionadas ou removidas do canal, a política de endosso é atualizada automaticamente para exigir mais ou menos endossos. As transações precisarão ser endossadas pelos nós Org1 e Org2 (nós de validação - *endorsers*). Um exemplo de utilização deste parâmetro é a utilização *--signature-policy* "AND('Org1MSP.member', 'Org2MSP.member')".

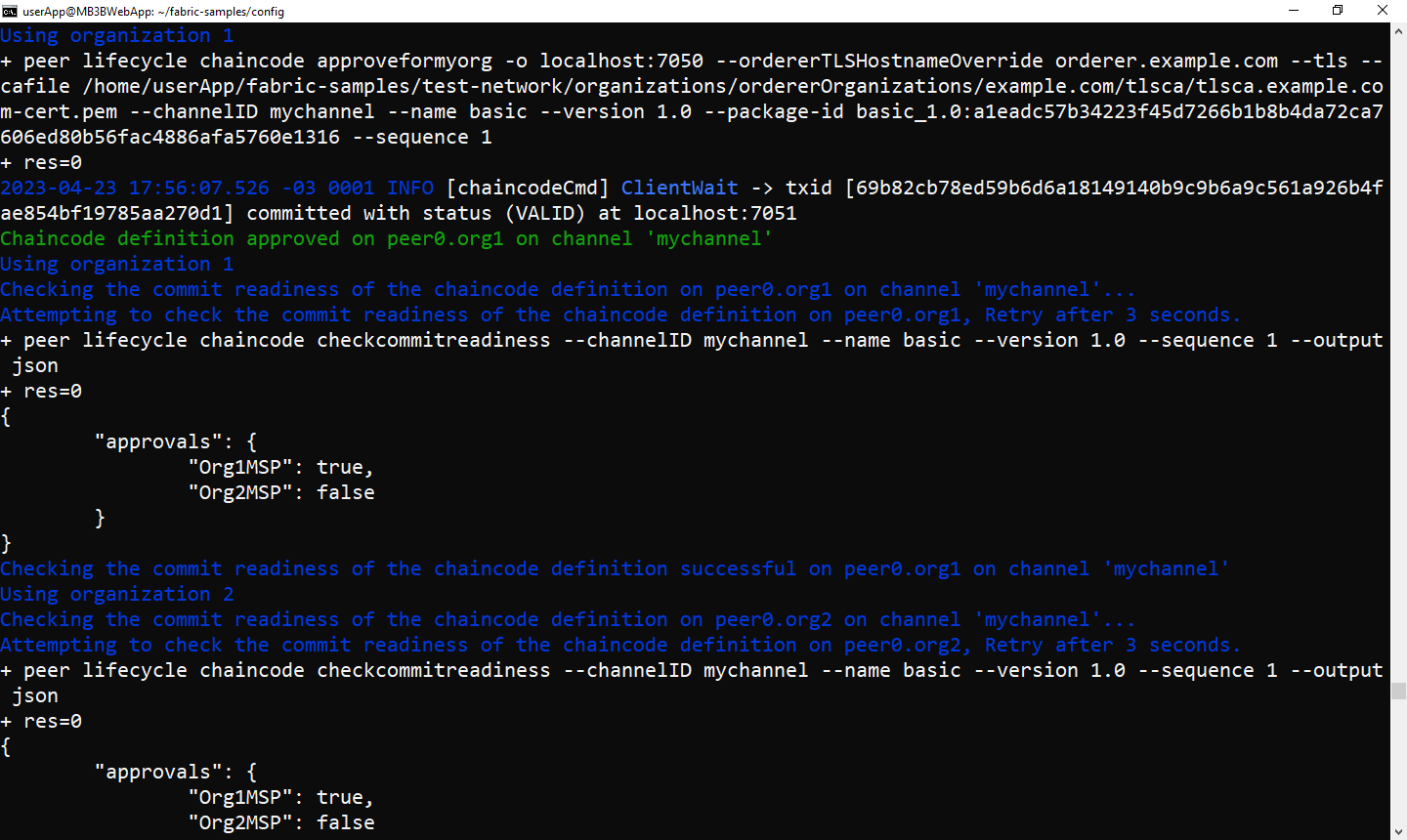


Figura . – Comando peer lifecycle chaincode approveformyorg em **org1.**

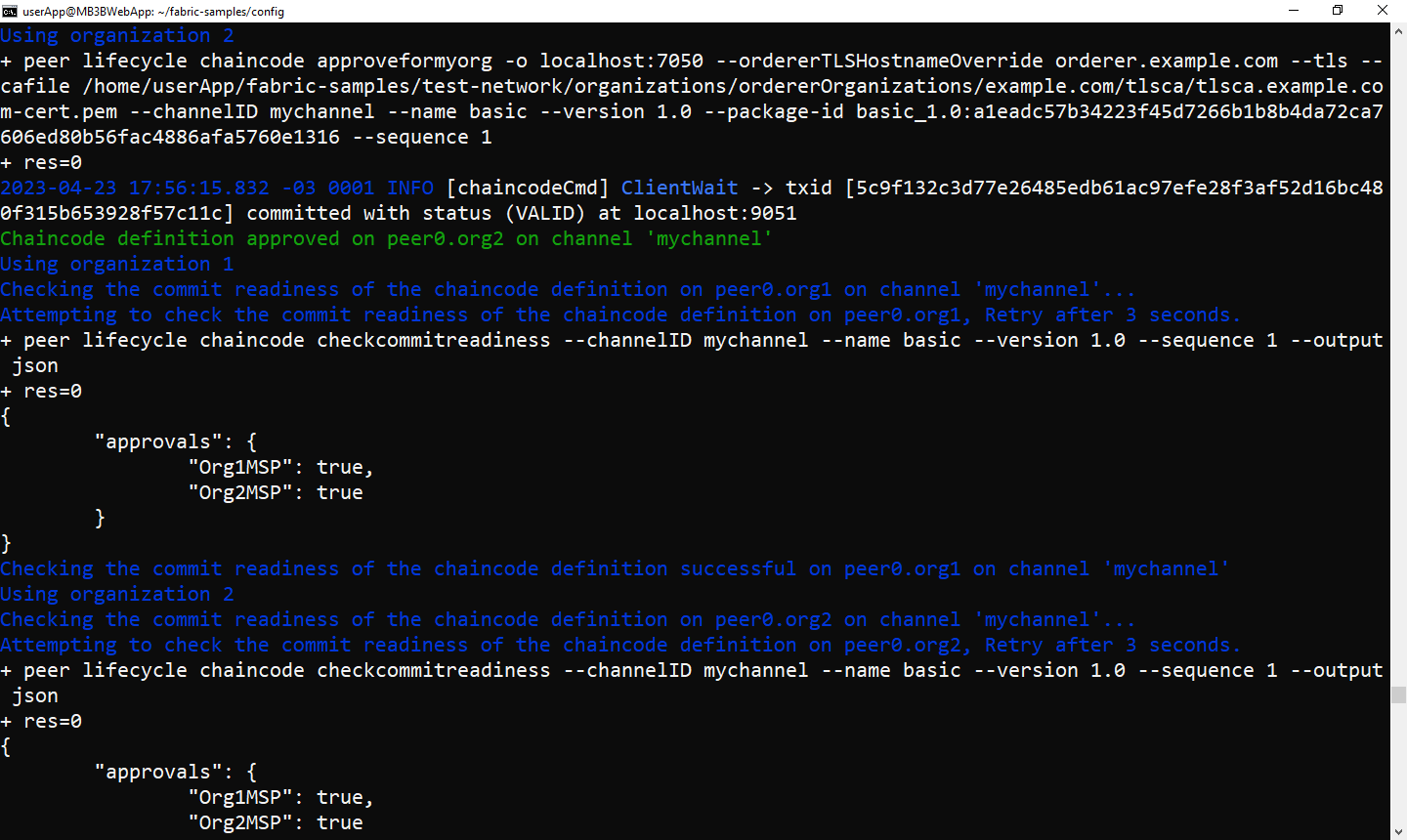


Figura . - Comando peer lifecycle chaincode approveformyorg em **org2**.

## Confirmação da definição do chaincode

Após a confirmação de que todas as organizações aprovaram as definições do chaincode, o nó de ordenação (*orderer*) pode confirmar (*commit*) essas definições no canal

***peer lifecycle chaincode commit*** *-o --ordererTLSHostnameOverride --tls --cafile --channelID --name --peerAddresses --tlsRootCertFiles*

Este comando ***peer lifecycle chaincode commit*** usa a flag ***--peerAddresses*** para os endereços peer0.org1.example.com da Org1 e o peer0.org2.example.com da Org2. Para obter a confirmação da política de implantação de um chaincode. Como a aprovação é distribuída dentro de cada organização, você pode direcionar qualquer peer que pertença a um membro do canal.

Essas solicitações de aprovação são enviadas ao serviço de ordenação (nó de ordenação) para serem adicionadas a um bloco e distribuídas ao canal. Os nós (peers) do canal confirmar se um número suficiente de organizações aprovou a definição do chaincode. O comando ***peer lifecycle chaincode commit*** aguardará as validações do nó de ordenação (*orderer*) antes de retornar uma resposta.



Figura . - Confirmação das definições do chaincode.

## Invocação do chaincode

A aplicação cliente é o ponto de interação entre o mundo externo, a rede blockchain e as regras de negócio implementadas nos chaincodes e esta interação é feita através dos SDKs.

O Hyperledger Fabric disponibiliza os *Software Development Kits* (SDKs) para Java, Node.js (JavaScript) e Go. Nesta prova de conceito todos os códigos fonte foram escritos na linguagem Javascript (Node.js).

Existem dois lados para desenvolvimento de software, as aplicações clientes que invocam os chaincodes, e os próprios chaincode que implementam as regras de negócio da portabilidade numérica.

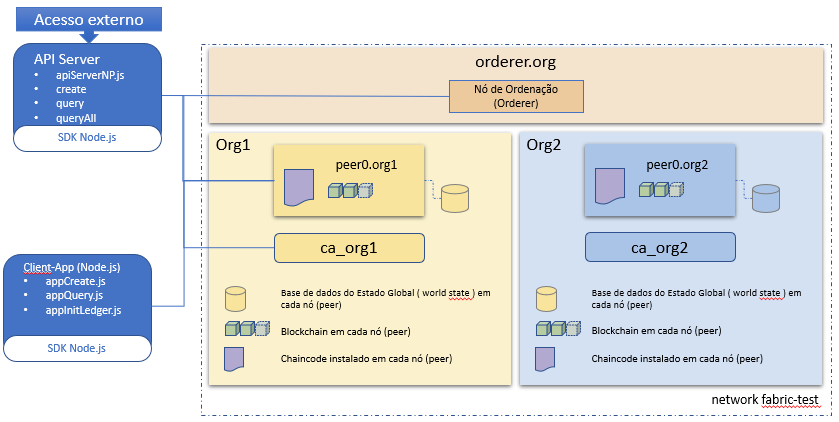


Figura . - Integrações entre o cliente, acesso externo e a rede fabric-test.

A figura 34 mostra a rede utilizada nesta prova de conceito onde todos os componentes foram implementados como containers e rodam em um mesmo host. Nesta figura são apresentadas algumas formas de invocação dos chaincodes, uma aplicação *Client-A*pp composta pelos códigos em Javascript appCreate.js, appQuery.js e appInitLedger.js salvos em uma pasta local */home/userApp/fabric-samples/asset-transfer-basic/application-javascript*, e uma API (Application Programming Interface) Server através do framework *ExpressJS*, que implementa requisições de invocação aos chaincodes.

Nesta prova de conceito, a aplicação de Portabilidade Numérica é uma base de dados com a seguinte estrutura:



Figura . - Estrutura dos dados.

Onde:

* MSISDN é a chave primária desta estrutura e representa o número do telefone do usuário;
* NOME e CPF são as informações de identificação do usuário;
* MCCMNC é um par de informações MCC (Mobile Country Code - Código do país) e MNC (Mobile Network Code - Código da operadora)
* OPERADORA é um rótulo de identificação da operadora.

Alguns campos são redundantes como CPF e NOME, e MCCMNC e OPERADORA, porém permanecerão na estrutura para melhor visualização das invocações e resultados.

## Invocação por Aplicativo Cliente – Cliente-App

Como a Hyperledger Fabric é uma plataforma de blockchain privada e permissionada, todos os participantes devem ser autorizados antes de poderem interagir. Desta forma, até a aplicação cliente (*client-app*) precisa dessa permissão. Isso ocorre através dos certificados digitais emitidos pela Autoridade Certificadora aceitos pela rede. Em nossa estrutura da prova de conceito são as entidades ca\_org1 e ca\_org2.

Essa a função *main()* cliente irá interagir com o nó (*peer*) e o nó de ordenação (*orderer*) para obter a aprovação e gerar os blocos, de acordo com as transações.

Dois módulos são necessários para o desenvolvimento dessa aplicação, que são *fabric-ca-client* and *fabric-network*. O *fabric-ca-client* é uma Autoridade Certificadora e o fabric-network disponibiliza recursos para submissão das consultas e transações na rede. Não é objetivo deste trabalho detalhar como esses módulos são utilizados no desenvolvimento desta aplicação, mas explorar as suas lógicas de implementação.

A primeira parte do código é responsável pelo *enrollment* dos usuários ***appUser*** e ***admin***. *Enrollment* se refere ao processo de registrar um usuário. Durante o processo de registro, um usuário é autenticado e recebe um certificado digital e uma chave privada, que são usados para identificar e assinar transações. Esse processo de registro é realizado pela Autoridade Certificadora (CA) e é usado para estabelecer a identidade e a autorização do usuário na rede. Uma vez que um usuário é inscrito com sucesso, ele pode acessar a rede e interagir com os chaincodes.

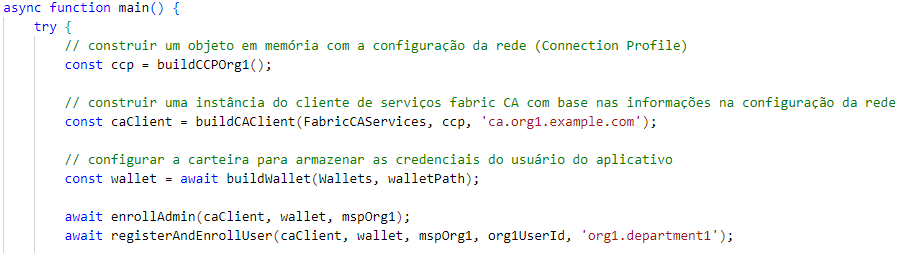


Figura . - Código appCreate.js e a criação da carteira (wallet)

Como resultado desta ação, os arquivos com os certificados dos usuários são armazenados na pasta wallet....

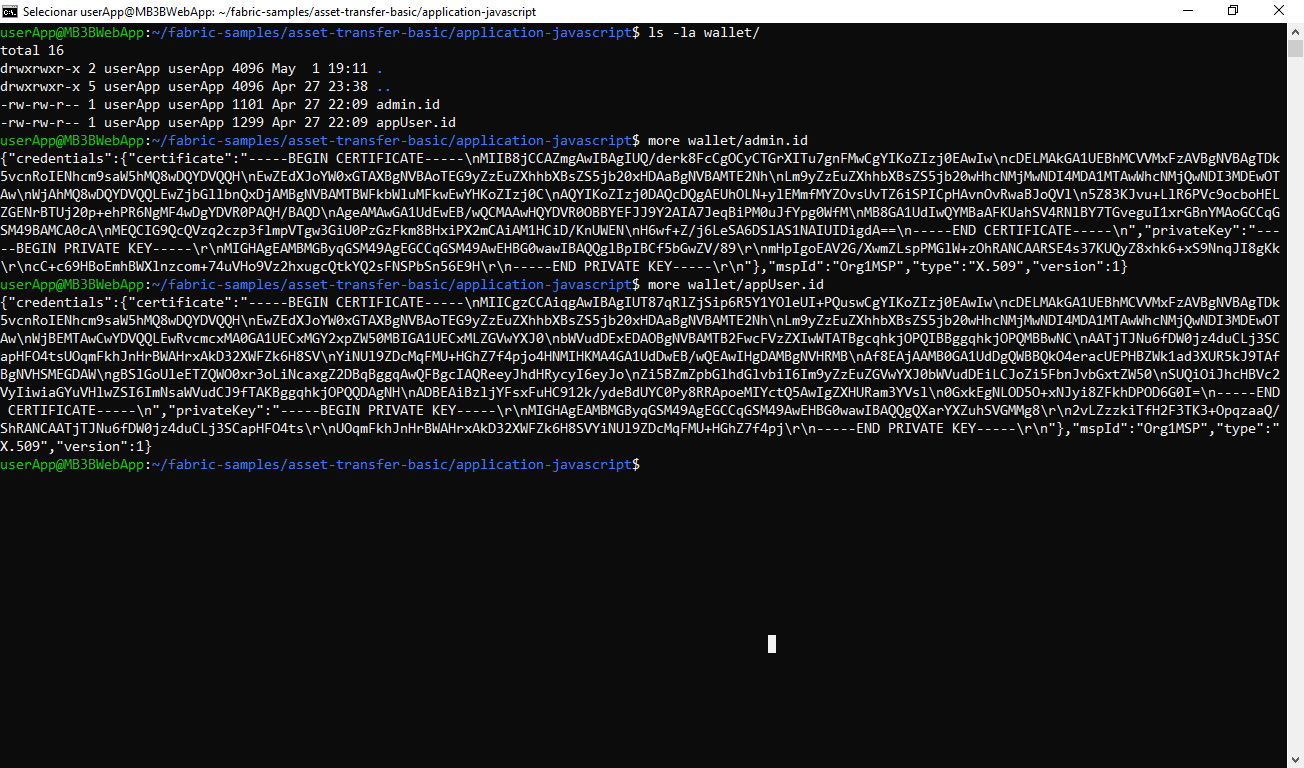


Figura . - Certificados dos usuários admin e appUser na carteira (wallet).

O próximo passo da função é criação de um gateway para se conectar com o nó (*peer*) e ao canal “myChannel”, e instanciar um *contract*, que será a API de interação com a chaincode....

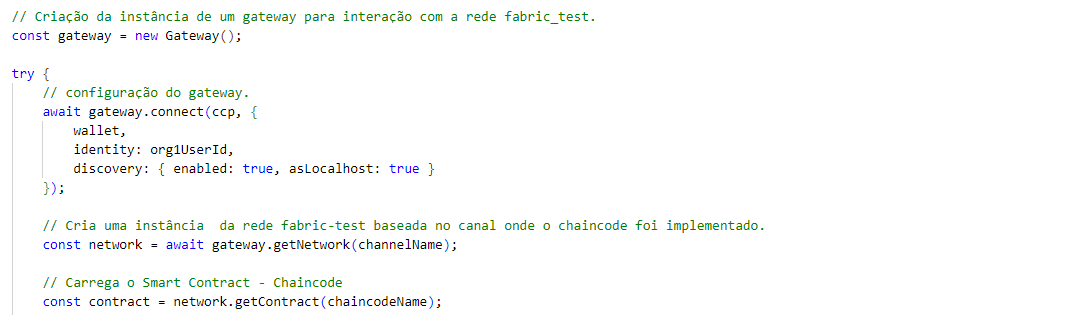


Figura . - Criação do gateway para conexão ao nó e ao canal.

Finalmente os códigos que são o objetivo desta função, que é a criação de um novo usuário *CreateAsset*, através da API *submitTransaction* com os respectivos argumentos. Neste exemplo é criado um usuário fictício para teste da função.

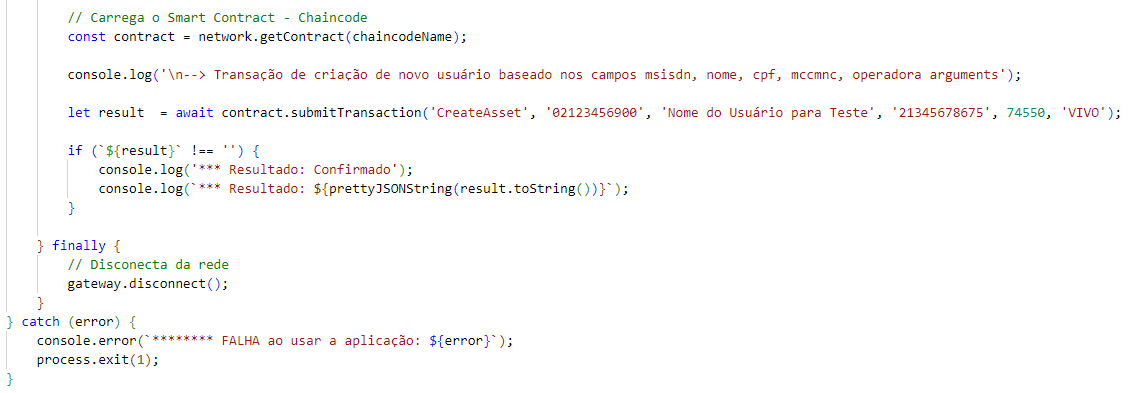


Figura . - Criação de um novo usuário.

Na criação deste usuário neste exemplo temos os seguintes dados:

* MSISDN = 02123456900;
* NOME = 'Nome Fictício do Usuário para Teste';
* CPF = 21345678675;
* MCCMNC = 74550;
* OPERADORA = VIVO.

A resposta para o comando de criação do novo usuário seguido pela invocação da consulta ao mesmo usuário segue abaixo:

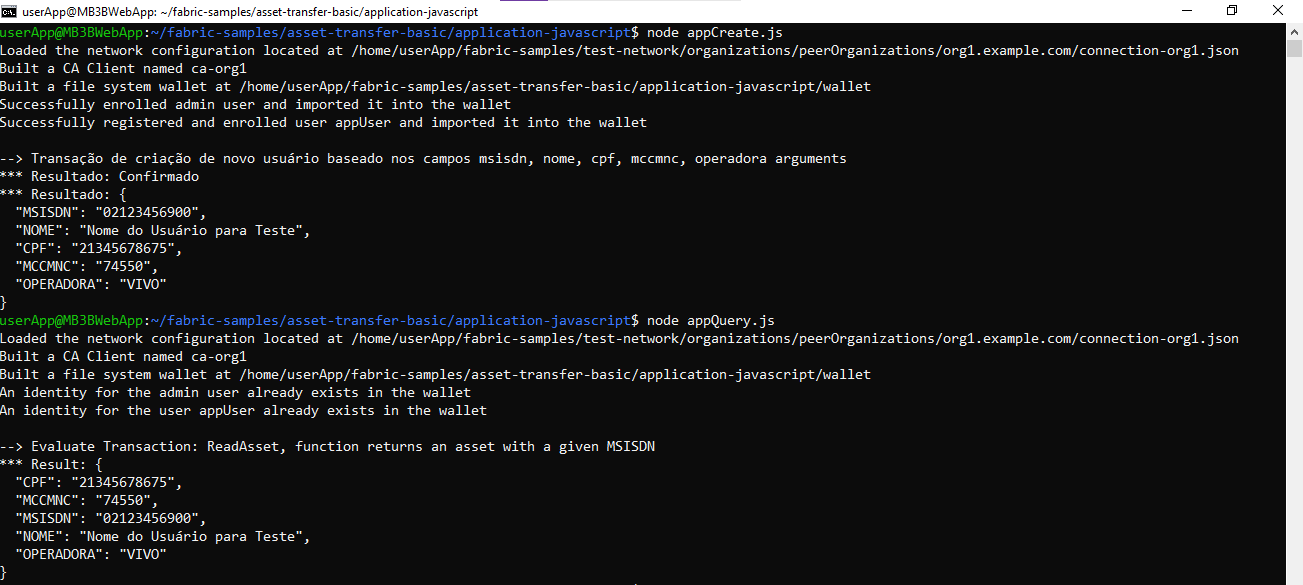


Figura . - Resposta à execução de appCreate.js e appQuery.js.

## Invocação por API Server

O exemplo que foi analisado mostra toda a estrutura de módulos para a invocação do chaincode, que é uma opção de interação com a plataforma Hyperledger Fabric utilizada nesta prova de conceito. Outra forma de interação com essa plataforma é através de implementação de APIs através de um servidor Web.

Neste trabalho está sendo utilizado o framework ExpressJS que dispõe de uma forma muito simples para implementação das APIs.

Para este framework foram implementadas essas quatro APIs para acompanhamento dos resultados esperados.

[Descrever o Express.js]



Figura . - Estrutura do apiServerNP.js e sua APIs.

O arquivo apiServerNP.js foi escrito para o framework ExpressJS com essas APIs que irão invocar os chaincodes. As APIs implementadas neste arquivo são descritas abaixo:

## API initLedger

Esta API invoca o chaincode InitLedger para uma carga inicial dos primeiros blocos do livro razão.

O chaincode é obtido através de uma instancia da classe *contract* através do código:

*const contract = network.getContract(chaincodeName);*

*Onde chaincodeName* é o nome dado ao chaincode no momento em que a rede é carregada, nesta aplicação da provade conceito possui o nome “*basic*”.

Para submeter uma transação ao chaincode são utilizadas duas funções *submitTransaction* e *evaluateTransaction*. A diferença entre essas duas funções é a base das transações nesta rede blockchain, submitTransaction submete os resultados da transação que são retornados ao invocar o chaincode, ao nó de ordenação (*orderer*) e aguarda a confirmação da transação pelos nós (*peers*). Isso significa que os resultados da transação serão ordenados e entregues aos nós (*peers*) pares para validação e serão gravados na blockchain.

Em resumo, a função *submitTransaction* é usado para transações que alteram o estado global e *evaluateTransaction* é usado para transações que apenas consultam o estado global.

A partir da instanciação do chaincode, a transação “InitLedger” é submetida ao chaincode pelos códigos abaixo:

*await contract.submitTransaction ('InitLedger');*

*............console.log ('\*\*\* Resultado: inicialização completa');*

*............res.status(200).json({Response: "Resultado: inicialização completa"})*

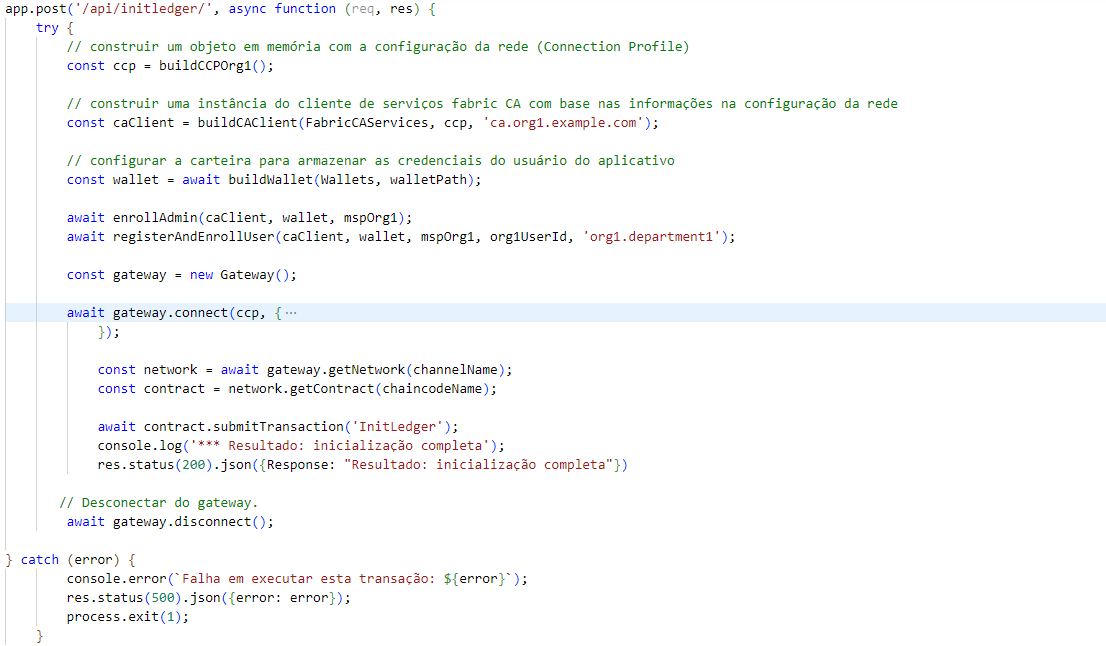


Figura . – Código da função de Invocação da API InitLeadger.

A resposta ao comando é uma confirmação de que a transação ocorreu com sucesso, ou o erro é apresentado na estrutura de exceção pelos blocos try-catch mostrados na figura 41.

## API Create

Esta API invoca o chaincode *CreateAsset* para adicionar novo registro de usuário no blockchain. Como se trata de um novo registro, todas as informações deste usuário são esperadas, que representa uma lista dos 5 parâmetros apresentados anteriormente, MSISDN, NOME, CPF, MCCMNC e OPERADORA.



Figura . - Código da função de Invocação da API Create

A resposta ao comando é uma confirmação de que a transação ocorreu com sucesso e entrega dos dados do novo usuário, ou apresentação do erro.

## API Transfer

Esta API invoca a principal função do chaincode nesta prova de conceito, que é a execução da portabilidade numérica com a alteração dos campos MCCMNC e OPERADORA.

O chaincode *TransferAsset* é invocado sendo esperados os parâmetros MSISDN, novo MCCMNC e nova OPERADORA, caracterizando a atualização desses dados de um usuário na blockchain e, consequentemente a troca de operadora.

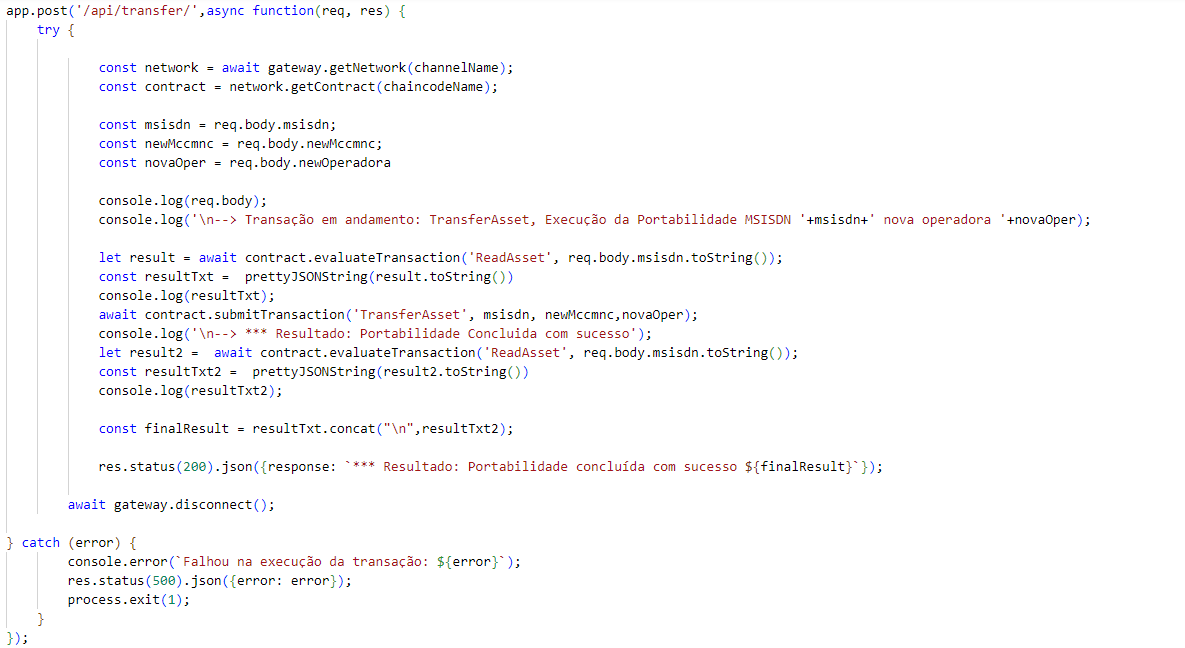


Figura . - Código da função de Invocação da API Transfer.

A resposta ao comando é uma confirmação de que a transação ocorreu com sucesso e comparação dos dados do usuário antes e depois da transferência, ou apresentação do erro.

## API Query

Esta API invoca o chaincode *ReadAsset* para consulta de um usuário no blockchain, através do parâmetro MSISDN recebido na invocação desse chaincode.

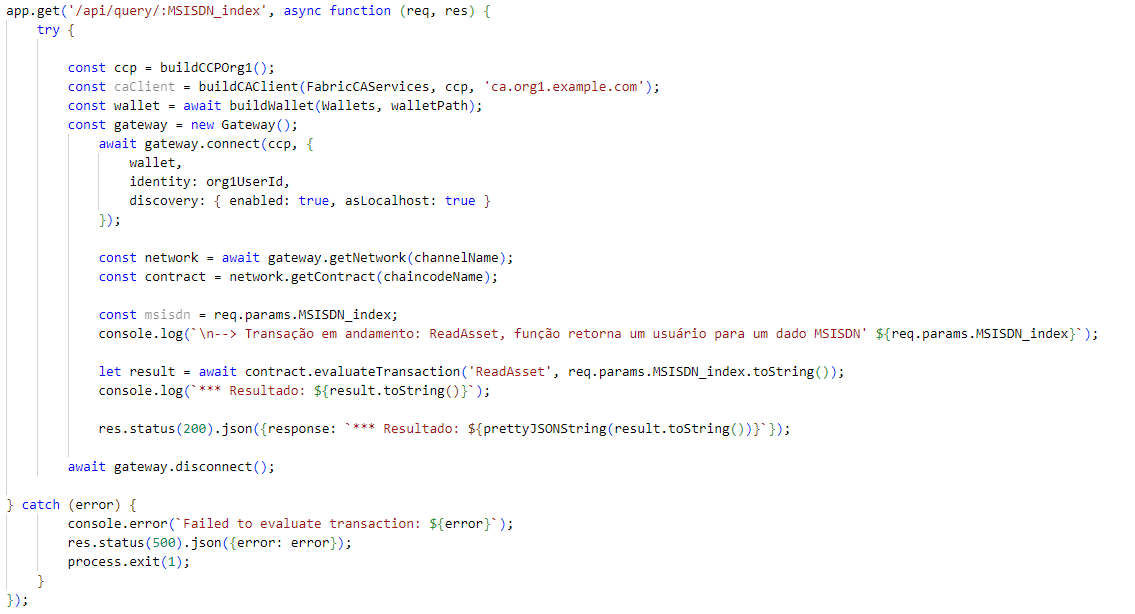


Figura . - Código da função de Invocação da API Query

## API QueryAll

Esta API invoca o chaincode *GetAllAssets* para uma consulta de todos os usuários existentes.

Figura . - Código da função de Invocação da API QueryAll





## Chaincodes

Conforme descrito anteriormente, Chaincode é um programa que pode ser desenvolvido nas linguagem de programação GoLand (Google 2023), Node.JS (OpenJS JavaScript 2023) ou Java (Oracle 2023), e que define as regras de negócio das transações que irão ocorrer na rede blockchain.

Neste trabalho todos os códigos foram escritos em Javascript (Node.JS) e estão sendo seguidas as orientações e modelos da documentação oficial. A implementação começa com uma instância da classe *contract* que implementada através do pacote *fabric-contract-api* (Fabric Contract API 2023).

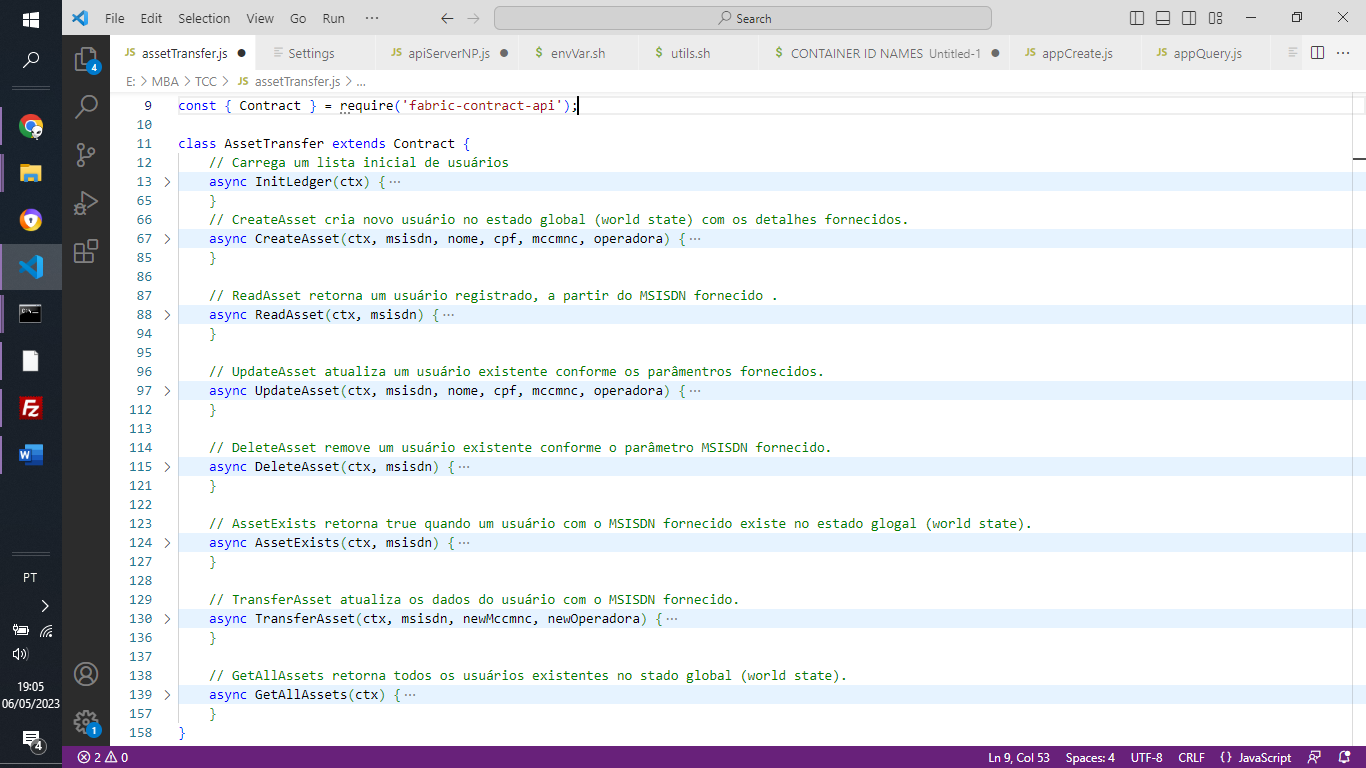


Figura . - Arquivo AssetTransfer.js com todas os Chaincodes.

## Chaincode InitLedger

A API *InitLedger* da API Server, explicada anteriormente, invoca este chaincode que, por sua vez, executa linhas de comando que irão carregar uma lista com os dados iniciais de alguns usuários fictícios na base de dados.

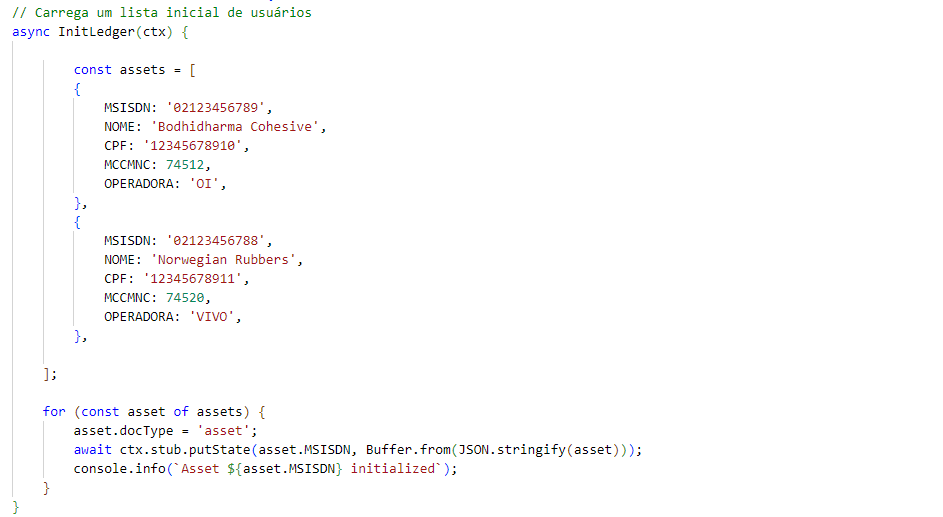


Figura . - Chaincode InitLedger.

## Chaincode CreateAsset

A API *Create* da API Server, invoca este chaincode para criação de um novo usuário conforme os parâmetros informados.

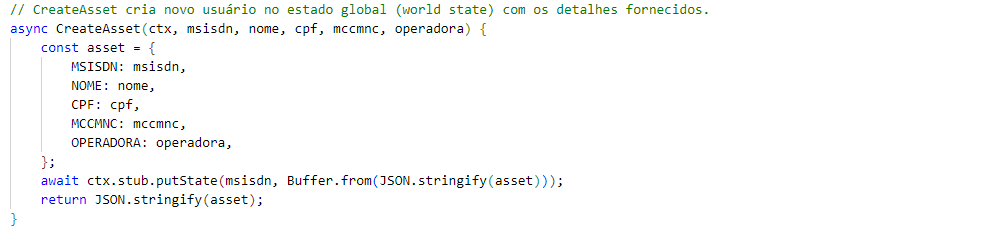


Figura . - Chaincode CreateAsset.

## Chaincode ReadAsset

A API Query da API Server, invoca este chaincode para consulta de um novo usuário conforme o MSISDN informado.

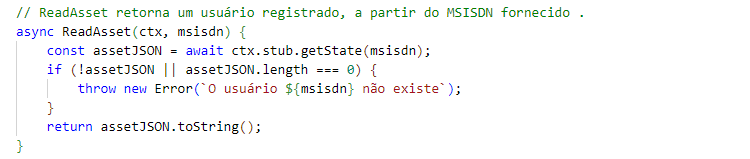


Figura . - Chaincode ReadAsset.

## Chaincode UpdateAsset

Este chaincode atualiza os dados de um usuário existente. Nesta função é usada outra função para verificar se o usuário existe *AssetExists*.

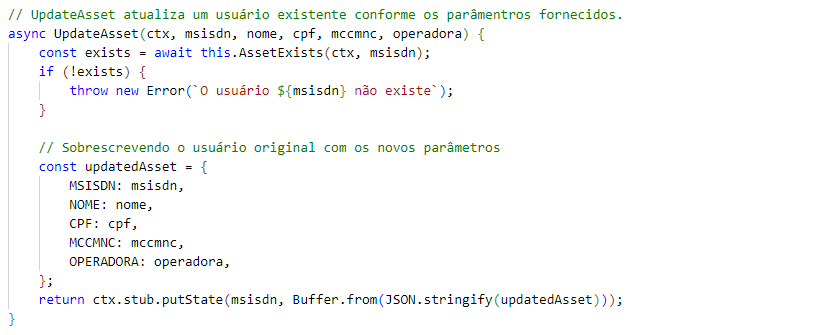


Figura . – Chaincode UpdateAsset.

## Chaincode DeleteAsset

Este chaincode remove um usuário de acordo com o parâmetro MSISDN fornecido e também executa a verificação da existência do usuário.

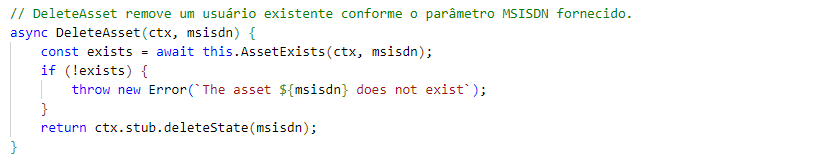


Figura . - Chaincode DeleteAsset.

## Chaincode AssetExists

Este chaincode é uma função de apoio às outras funções e retorna TRUE quando o usuário com o MSISDN fornecido existe na base de dados.

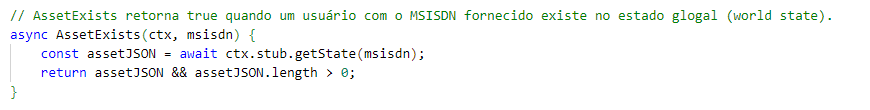


Figura . - Chaincode AssetExists.

## Chaincode TransferAsset

A API *Transfer* da API Server, foi explicada anteriormente como a função que implementa a portabilidade numérica, que é o principal objetivo desta prova de conceito ao qual este trabalho está baseado. O chaincode que implementa a regra de negócio de portabilidade numérica é este chaincode *TransferAsset*.

A função ReadAsset é chamada e irá retornar todos os parâmetros do usuário identificado pelo seu MSISDN. A Portabilidade numérica consiste na substituição dos campos MCCMNC e OPERADORA pelos novos campos conforme a transação.

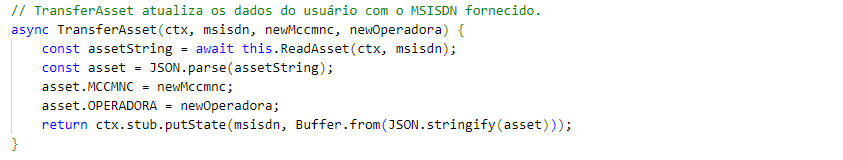


Figura . - Chaincode TransferAsset.

## Chaincode GetAllAssets

A API QueryAll da API Server invoca este chaincode para retornar todos os usuários registrados na base de dados.

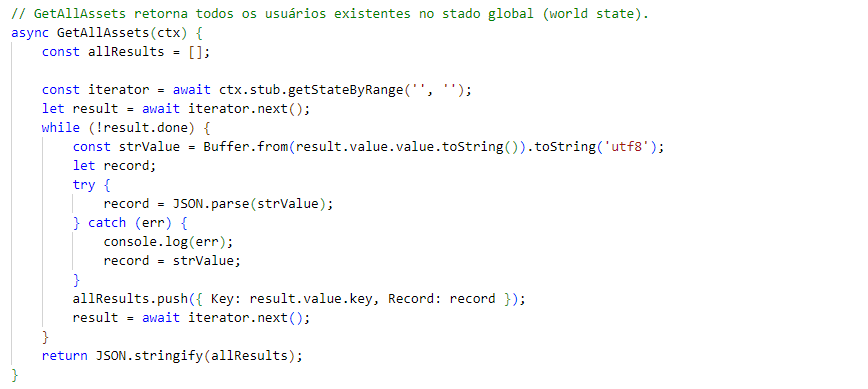


Figura . - Chaincode GetAllAssets.

# Resultados Esperados e Obtidos

O problema lançado na introdução deste trabalho questiona qual solução tecnológica pode ser a opção capaz de substituir o modelo de portabilidade numérica existente e que garanta a confiabilidade. Como proposta para essa solução foi apresentada a tecnologia blockchain e suas principais características, garantindo que os registros serão imutáveis e tratados em uma rede descentralizada com entidades que possuem altíssmo grau de desconfiança entre si.

Nos capítulos anteriores foram apresentadas as justificativas teóricas e os detalhes técnicos na implementação da plataforma Hyperledger Fabric para que fosse possível emular as transações de portabilidade numérica entre as operadoras de telefonia em um cenário hipotético para que os resultados obtidos sustentem a defesa da utilização da tecnologia blockchain como uma opção de implementação da portabilidade numérica.

Neste capítulo serão apresentados esses resultados obtidos e que sustentam a conclusão deste trabalho.

As transações de portabilidade numérica foram executadas com invocações dos chaincodes através de requisições HTTP realizadas pela aplicação CURL (CURL 2023) e para atender à estas requisições externas, foi mantida uma aplicação respondendo através da porta 8080 de um servidor Web implementado com o framework ExpressJS (Node.js 2023). As análises foram baseadas em captura dos logs coletados dos *containers*.

O fluxo de uma transação conforme definido em (Androulaki, et al. 2018) e mostrado na Figura 2.3 - Fluxo das transações da plataforma Hyperledger Fabric. O cliente envia uma proposta de uma transação (*transaction proposal*) para os nós de validação (*endorsers*). Estes nós de validação (*endorsers*) confirmam que o cliente é válido para enviar uma proposta de transação. Além dessa validação, o nó executa o *chaincode* conforme as informações especificadas na requisição e com o estado global (*world state*) e retornam a resposta à requisição ao cliente. Esta resposta é uma simulação (*Read Write Set (RWSet)*), que mostra o que foi lido do estado global (*world state*) e o que será escrito após a invocação do *chaincode*, validando esta resposta(*endorsement*).

O cliente após receber essas respostas, faz a validação para garantir que a transação ocorre conforme esperado. Se as respostas recebidas forem aderentes à política de aprovação, que nesta prova de conceito se refere a validação de todos os 2 nós, o cliente constrói uma transação definitiva e envia ao nó de ordenação(*orderer*). Essa nova transação inclui a proposta de transação (*transaction proposal*), resposta desta proposta (*proposal response*) e as aprovações (*endorsement*).

O nó de ordenação (*orderer*) recebe as transações dos clientes e faz sua validação. Em seguida, o nó de ordenação (*orderer*) cria um bloco contendo as transações validadas e transmite esse bloco recém-criado para todos os nós(*peers*). Ao receber esse bloco, o nó (*peer*) valida e executa essa transação, atualizando estado global (*world state*) conforme o conteúdo no conjunto de leitura e gravação (RWSet). Finalmente, este bloco é confirmado em cada nó (*peer*) como um bloco válido.

Além desta descrição do fluxo de execução de uma transação, na figura abaixo é mostrado o fluxo onde será analisada as etapas de criação dos blocos e os seus conteúdos.

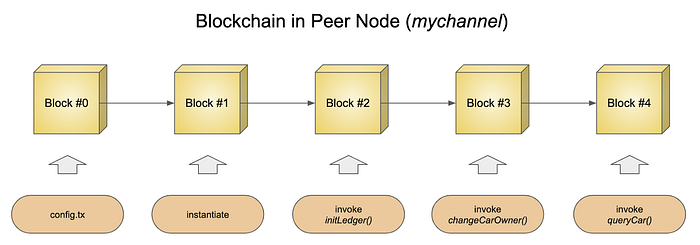


Figura . - Blocos criados no canal mychannel.

Quando todos os containers são carregados e o primeiro nó (peer) é conectado ao canal mychannel, o Block #0 é gerado.

O *block #0* é o bloco gênese que é criado através da execução do comando *configtxgen*, conforme apresentado na página 29, no capítulo 3.2.5.2 - *Criando o bloco de gênese*. São criados a pasta ./*channel-artifacts* e o arquivo *mychannel.block* com os detalhes definidos no arquivo *configtx.yaml.*

*O block #1* é gerado no momento em que o chaincode é instanciado no canal *mychannel,* conforme apresentado na página 37, no capítulo 3.3.3 - Instalação do chaincode em cada nó.

O *block #2* é criado na execução da função InitLedger(), conforme apresentado na página 44, no capítulo 3.3.6.2.1 - API initLedger. A invocação desta função cria 6 novos usuários.

O *block #3* é a simulação da transação de portabilidade onde um usuário tem o parâmetro OPERADORA e MCCMNC alterados, conforme apresentado na página 46, capítulo 3.3.6.2.3 - API Transfer.

O *block #4* é uma invocação à função ReadAsset, através da API Query, conforme apresentado na página 47, capítulo 3.3.6.2.4 - API Query.

# Como ler os blocos no Hyperledger Fabric

A leitura de um bloco da estrutura blockchain é feita através dos passos mostrados abaixo.

1 - Obter as informações do blockchain através do comando:

*docker exec [container] peer channel getinfo -c [channel\_name]*

Onde:

* *Container - em qual container será feita a análise;*
* *Channel\_name – em qual canal será feita essa análise.*

No cenário dessa prova de conceito, temos a seguinte resposta:

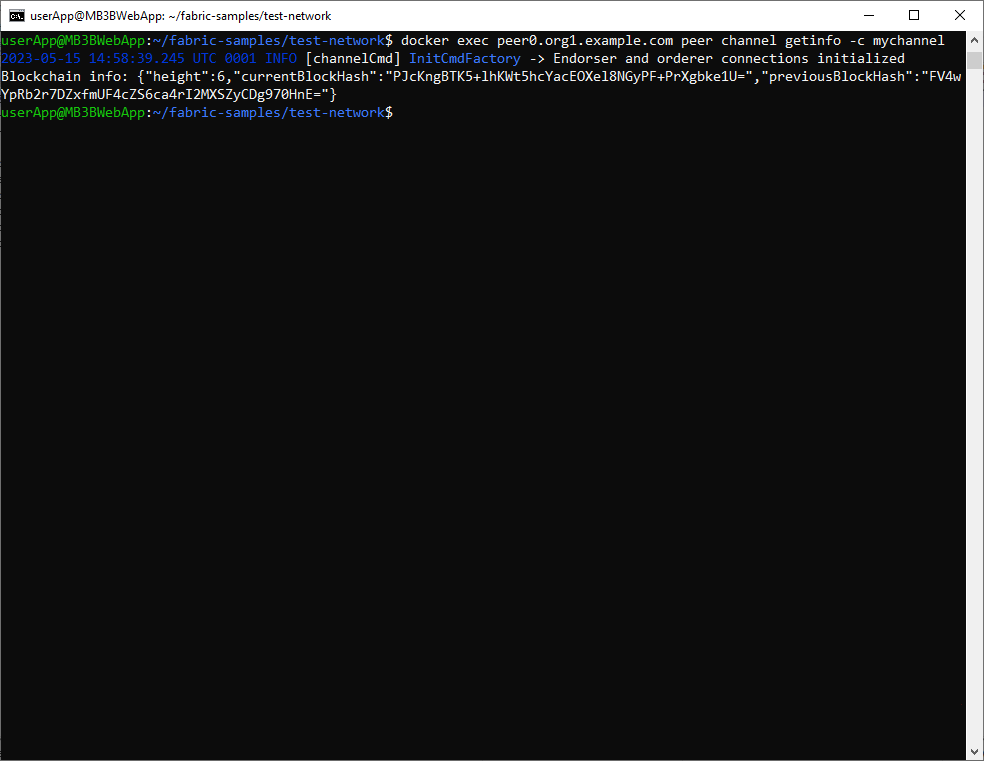


Figura - peer channel getinfo - informação dos blocos do peer 0, Org1.

Nesta resposta acima da figura 4.1.1 identificamos o tamanho (height = 6) e o block hash atual e do bloco anterior.

Segundo a definição de (PITTALIA 2019), uma função *hash* é uma função que recebe uma quantidade arbitrária de entrada e produz uma saída de tamanho fixo através da aplicação de algoritmos de criptografia. Sua justificativa é apresentada em (NAKAMOTO 2008), propondo que os valores de hash existentes nos blocos do blockchain são únicos e a garantem que não existe fraude uma vez que as alterações de qualquer bloco na cadeia alterariam imediatamente o respectivo valor de hash.

O estudo dessas funções *hash* e a forma como essa criptografia é implementada não foi tratado neste trabalho.

2 - Obter um bloco específico da estrutura blockchain.

Os blocos são numerados de bloco zero, que é bloco gênese, até o bloco “tamanho - 1” (height -1) e são obtidos através desse comando abaixo:

*docker exec [container] peer channel fetch [block\_number] -c [channel\_name]*

*Onde:*

* *[block\_number] - número do bloco*

Neste passo, o arquivo mychannel.block foi copiado para a pasta ./temp e convertido para o formato JSON.

*configtxgen -inspectBlock [block\_file\_name] > [block\_file\_name.JSON]*

*Onde:*

* *block\_file\_name - arquivo de origem;*
* *block\_file\_name.JSON – arquivo destino no formato JSON.*

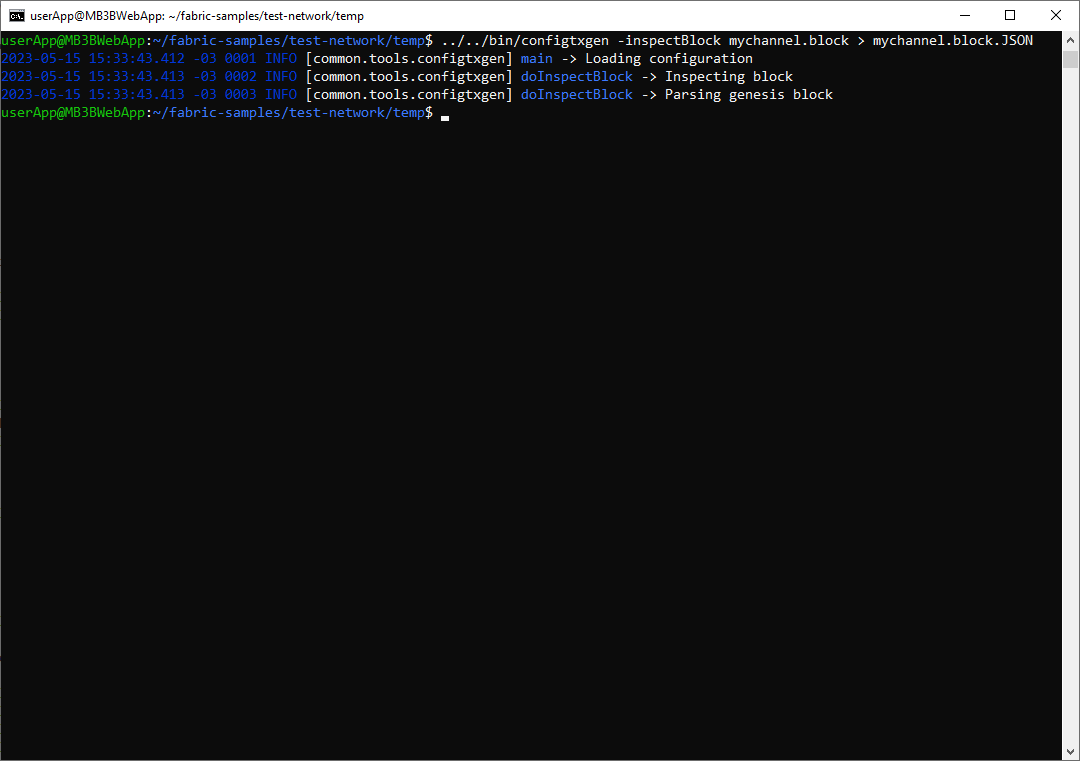


Figura - Comando de conversão do bloco em JSON.

O arquivo final, neste caso mychannel.block.JSON, possui uma estrutura conforme apresentado abaixo:

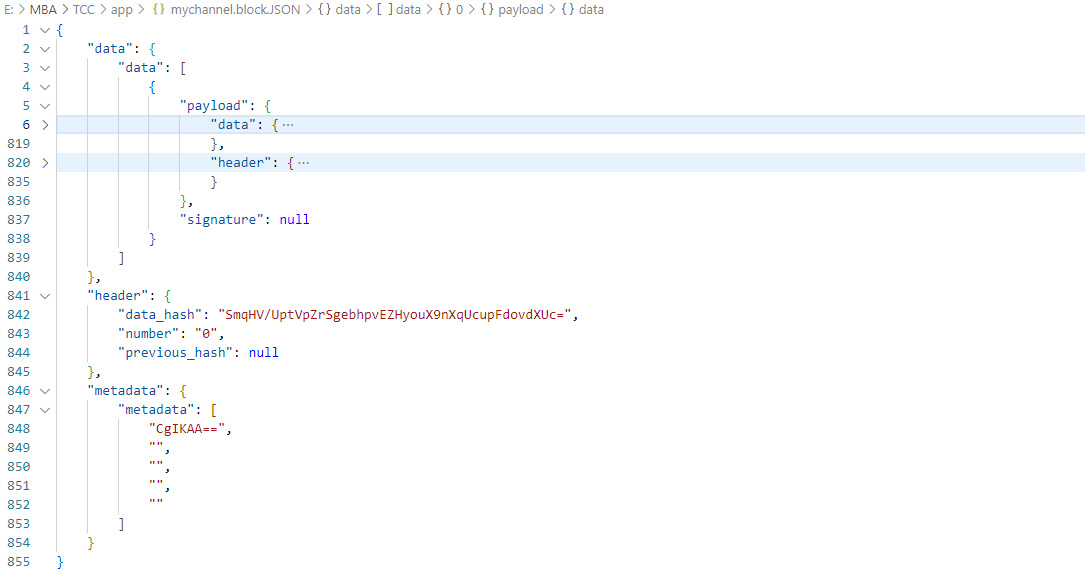


Figura - Arquivo do bloco convertido em JSON.

Conforme apresentado em (Hyperledger - Key concepts - Ledger 2023), os blocos que compõem um blockchain possuem uma estrutura composta por Header, Data e Metadata, conforme mostrado na figura abaixo.

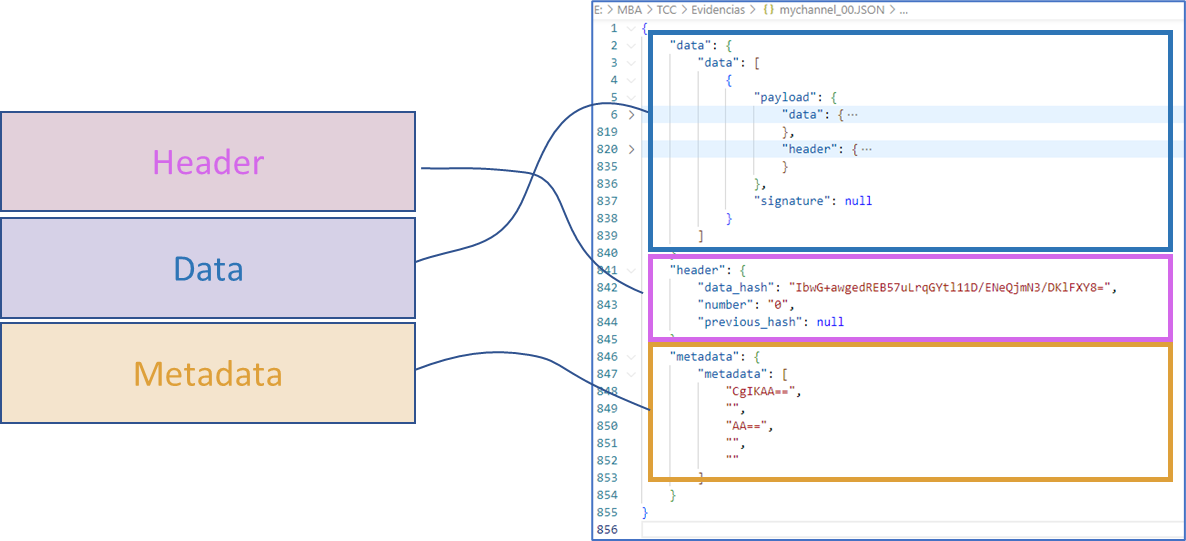


Figura - Estrutura básica de um bloco.

O *Header* apresentado nesta figura é do bloco gênese (block #0) e neste caso o "*previous\_hash: null”* mostra que não existe bloco anterior.

O bloco *Metadata* contém os certificados e assinaturas que serão usados validar o bloco nos outros nós da rede.

O bloco *Data* é onde as transações são armazenadas e é possível agrupar essas transações em três partes:

* Transaction Proposal: Um cliente envia esta proposta aos nós de validação (*endorsers*);
* Endorsements: Os nós de validação (*endorsers*) repondem com os seus Ids e assinaturas.;
* Proposal Response: É o resultado após os nós de validação (*endorsers*) executarem o chaincode invocado pela transação original.

A relação entre os blocos, as suas respectivas transações e essas três partes são mostradas na figura abaixo.

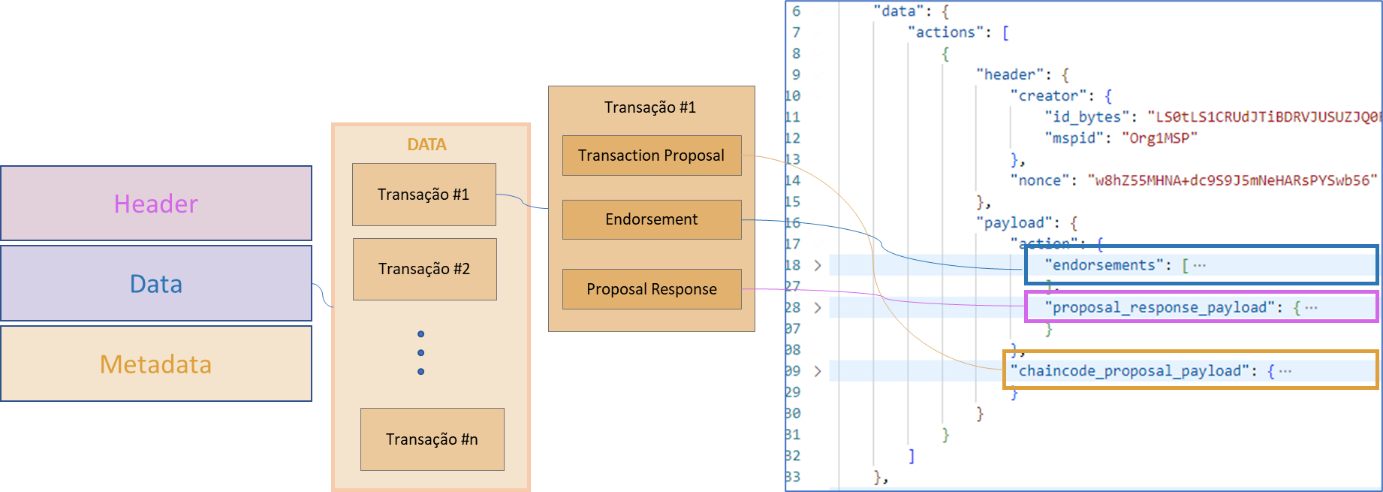


Figura - Relação entre os blocos e as transações.

*Transaction Proposal*

Conforme (Poniszewska-Marańda, Rojek e Pawlak 2022), a parte da estrutura DATA que representa os argumentos de entrada da transação é o campo ***args*** dentro da estrutura ***chaincode\_proposal\_payload***. Este campo é codificado como *BASE64* e após sua decodificação.



Figura - Estrutura do Transaction Proposal.

Neste caso os valores que temos são os seguintes:

|  |  |
| --- | --- |
| "args": [  "VHJhbnNmZXJBc3NldA==",  "MDIxMjM0NTY3ODQ=",  "NzQ1NDA=",  "U0VSQ09OVEVM"  ], | "args": [  "TransferAsset",  " 02123456784",  " 74540",  "SERCONTEL"  ], |

Tabela - Decodificação dos argumentos chaincode\_proposal\_payload

*Proposal Response* e *Endorsements*

Nos campos do *endorsements* mostrados abaixo*,* estão definidos os nós de validação (*endorsers*). Nestes casos estão os certificados de Org2MSP e Org1MSP.

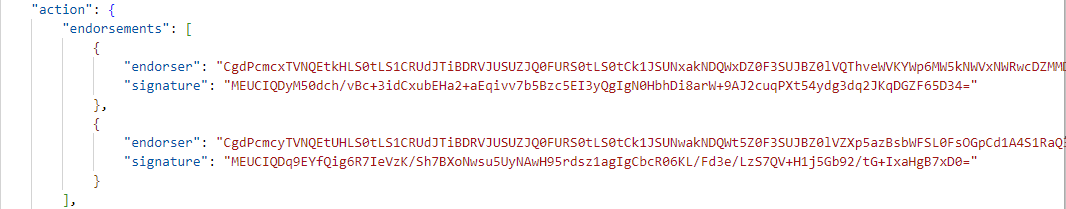


Figura 6 - Estrutura do Endorsements

Nos campos mostrados abaixo do *proposal\_response\_payload* o resultado é mostrado abaixo:



Figura - Estrutura do Proposal Response

Nos campos mostrados abaixo do *proposal\_response\_payload* o resultado é mostrado abaixo:

|  |  |
| --- | --- |
| "writes": [      {          "is\_delete": false,          "key": "02122345987",          "value": "eyJNU0lTRE4iOiIwMjEyMjM0NTk4NyIsIk5PTUUiOiJEZXNpc3RlbGlvIE1hbmRpdGFsIiwiQ1BGIjoiMDI0OTcyMTMxMTEiLCJNQ0NNTkMiOiI3NDU2MCIsIk9QRVJBRE9SQSI6IkJSSVNBTkVUIn0="      }  ] | "writes": [      {          "is\_delete": false,          "key": "02122345987",          "value": "{"MSISDN":"02122345987","NOME":"Desistelio Mandital", "CPF":"02497213111","MCCMNC":"74560","OPERADORA":"BRISANET"}”  }  ] |

Tabela - Decodificação do proposal\_response\_payload

# Análise das respostas dos chaincodes

Nesta análise foi utilizado o comando:

*docker logs [container] > [arquivo destino.log]*

Este comando exporta todos os logs do container.

# InitLedger()

A invocação desta função criou os primeiros 6 usuários fictícios no blockchain.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| MSISDN | NOME | CPF | MCCMNC | OPERADORA |
| '02123456789', | 'Bodhidharma Cohesive', | '12345678910', | 74512 | 'OI', |
| '02123456788', | 'Norwegian Rubbers', | '12345678911', | 74520 | 'VIVO', |
| '02123456787', | 'Mosaic Peppered', | '12345678912', | 74530 | 'CLARO', |
| '02123456786', | 'Assiduous Revere', | '12345678913', | 74540 | 'SERCONTEL', |
| '02123456785', | 'Delegated Waitresses', | '12345678914', | 74550 | 'SURF', |
| '02123456784', | 'Hotpoint Highlander', | '12345678915', | 74560 | 'BRISANET', |

Tabela - Tabela com dados de inicialização do blockchain

Esta transação encontra-se no bloco #6.

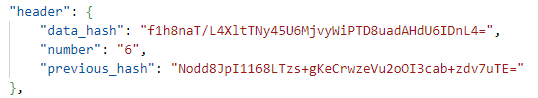


Figura - Header do bloco #6



Figura - Endorsements do Bloco #6 - InitLedger()



*Figura 4.10 - Visualização do proposal\_response\_payload do Bloco #6 - InitLedger()*

Decodificação dos valores encontrados no bloco #6, na estrutura *proposal\_response\_payload*:

*{"MSISDN":"02123456784","NOME":"Josebes Gallobe", "CPF":"12345678915", "MCCMNC":74560, "OPERADORA":"BRISANET","docType":"asset"}*

*{"MSISDN":"02123456785","NOME":"Astrelopitecus Robustus", "CPF":"12345678914", "MCCMNC":74550, "OPERADORA":"SURF","docType":"asset"}*

*{"MSISDN":"02123456786","NOME":"Hermogena Gongonzolina", "CPF":"12345678913", "MCCMNC":74540, "OPERADORA":"SERCONTEL","docType":"asset"}*

*{"MSISDN":"02123456787","NOME":"Gongonzolina Hermogena", "CPF":"12345678912", "MCCMNC":74530, "OPERADORA":"CLARO","docType":"asset"}*

*{"MSISDN":"02123456788","NOME":"Atadide de Austragesilo", "CPF":"12345678911", "MCCMNC":74520, "OPERADORA":"VIVO","docType":"asset"}*

*{"MSISDN":"02123456789","NOME":"Austragesilo de Atadide", "CPF":"12345678910", "MCCMNC":74512, "OPERADORA":"OI","docType":"asset"}*

Para análise das relações entre os nós (*peers*), será feita através da execução do comando *docker logs peer0.org1.example.com > logOrg1.log* para captura dos logs inicialmente no nó Org1, e em seguida nos nós Org2 e o nó de ordenação Orderer.

No log do nó Org1, três informações são relevantes, a confirmação de que está sendo analisado o bloco #6, o txID também existente no bloco e o commitHash.

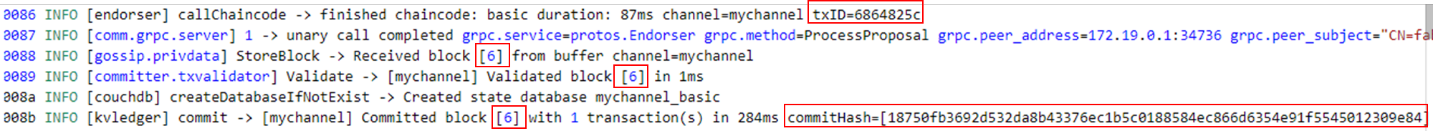


Figura - Log do nó Org1 para análise do bloco #6.

A execução do comando *docker logs peer0.org2.example.com > logOrg2.log* capturou os logs do nó Org2, e mostra que a mesma transação afetou este nó.

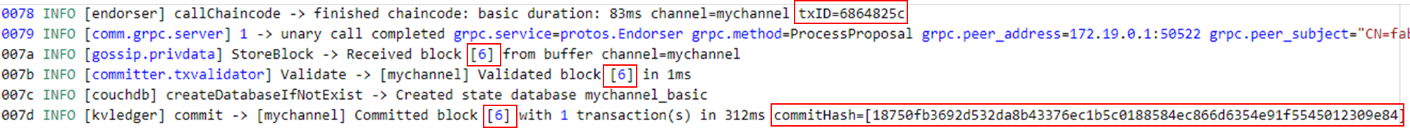


Figura 13 - Log do nó Org2 para análise do bloco #6.

No nó de ordenação os eventos referentes ao bloco #6.



Figura - Log do nó de ordenação.

A análise final desta transação ocorre na confirmação dos valores no estado global (World State) através do acesso ao banco de dados CouchDB utilizando a interface Fauxton.

A figura abaixo mostra a consulta à base de dados *mychannel\_basic*, onde o resultado esperado dos novos usuários criados através da função initLedger é mostrado.

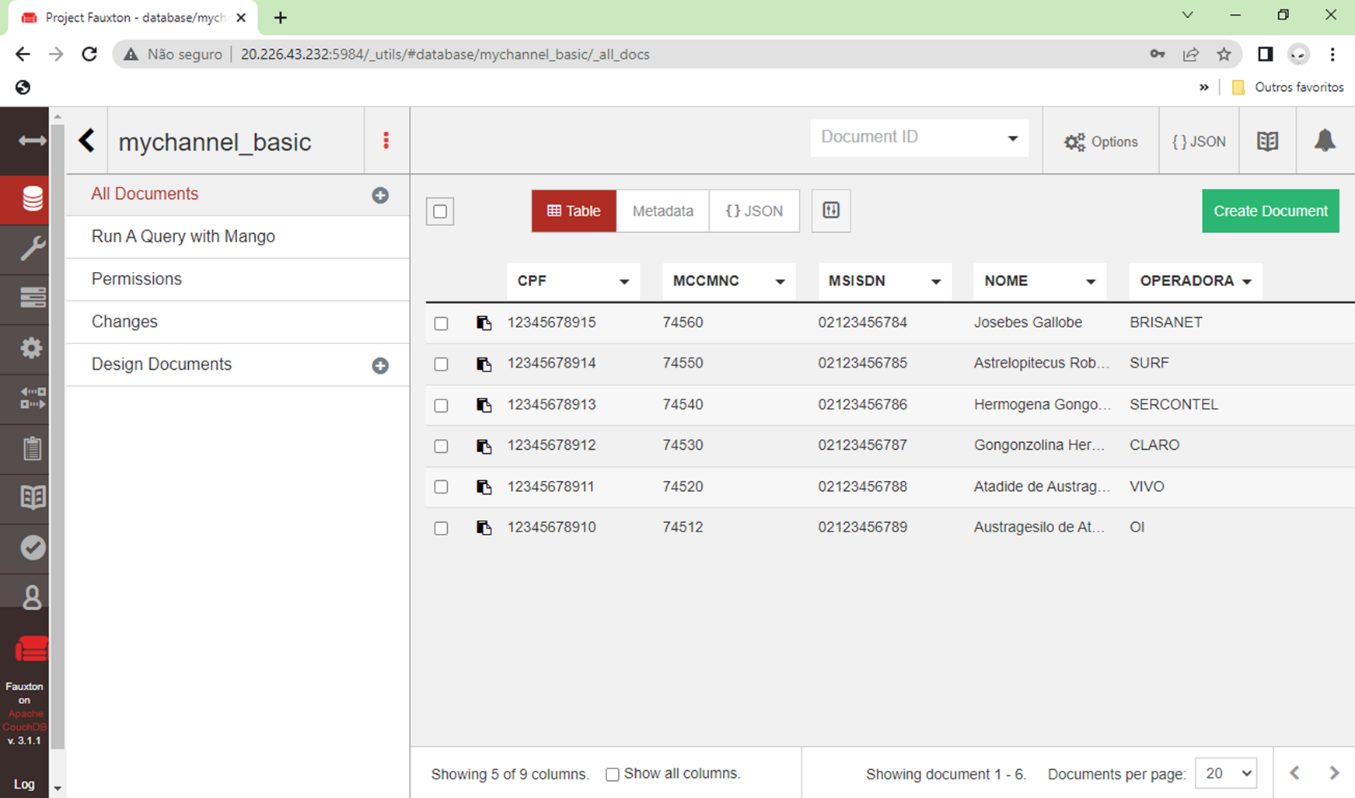
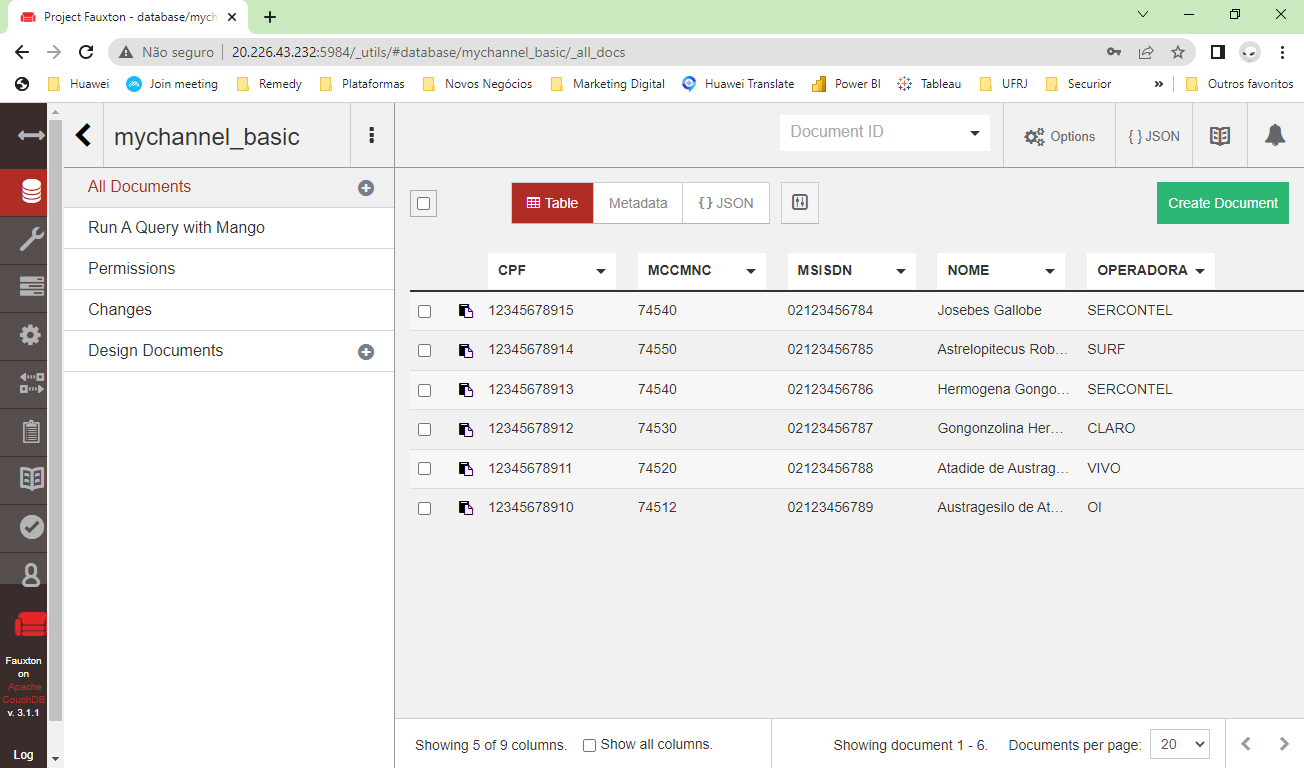
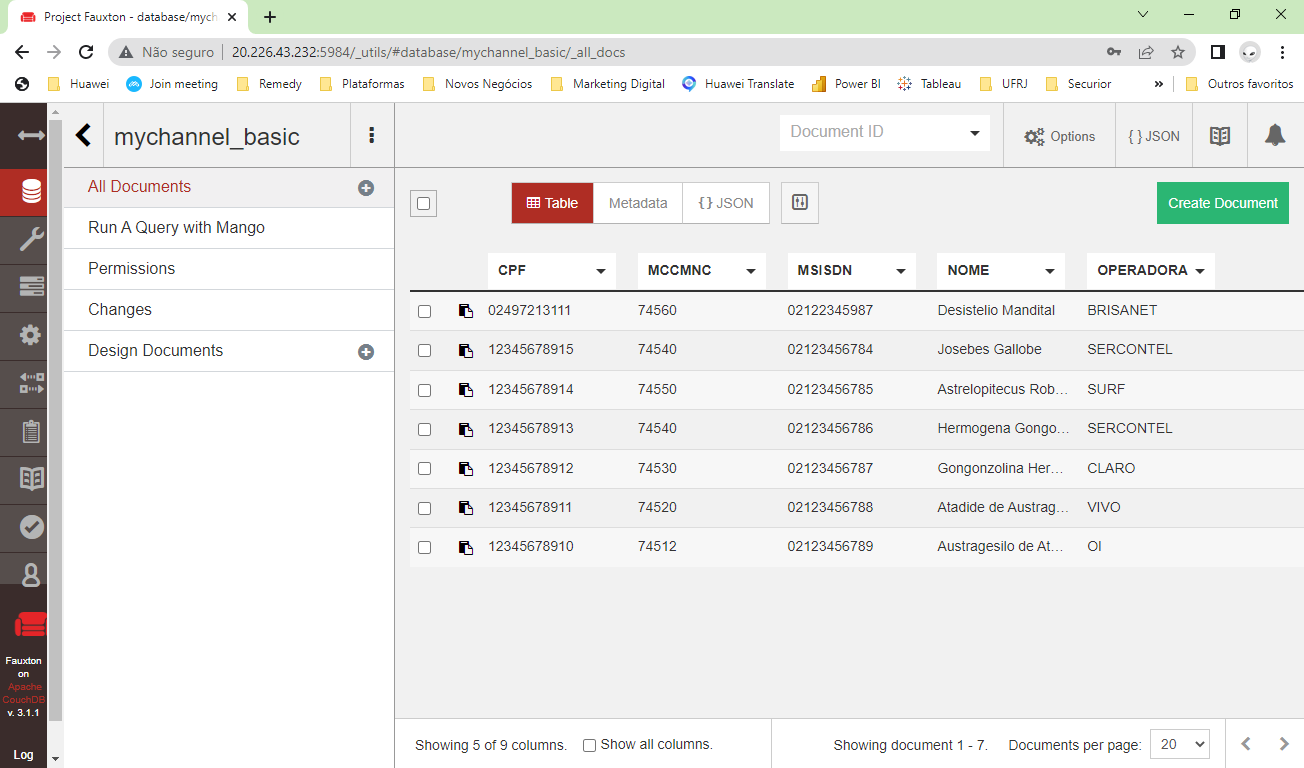


Figura - Resultado da função InitLedger no estado global (World State).

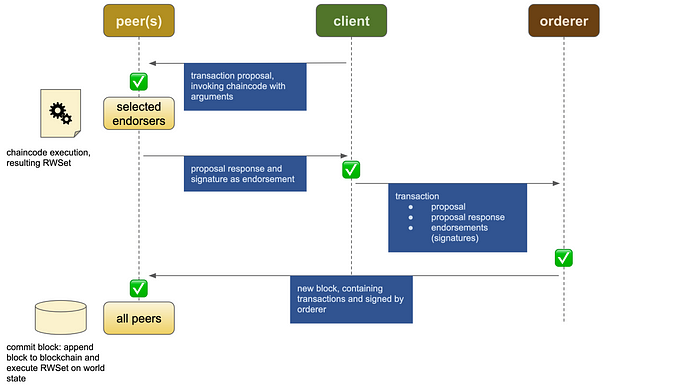
# Transfer()

Conforme já apresentado na página 46, capítulo 3.3.6.2.3 - API Transfer, esta função representa a portabilidade numérica que é objeto deste trabalho. É esperado que um cliente envie uma requisição de portabilidade de um usuário e a plataforma hyperledger fabric garanta que todos os nós irão receber esta transação com a garantia da integridade dos dados, segurança e transparência.





# Como ler os blocos no Hyperledger Fabric



[Mostrar os logos do Docker, mostrando as transações e as aceitações de cada nó.]

... comparar blocos com couchdb

<https://kctheservant.medium.com/transactions-in-hyperledger-fabric-50e068dda8a9>

<https://hyperledger-fabric.readthedocs.io/en/release-2.2/ledger/ledger.html#world-state>

Figura . - Código da função de Invocação da API QueryAll

**4.1 – Resultado 1**

1. **Conclusão e Trabalhos Futuros**

**5.1 – Conclusão**

**5.2 – Trabalhos Futuros**

# Referências Bibliográficas

# Bibliografia

ANATEL. “ANATEL - Agência Nacional de Telecomunicações - Resolução nº 750, de 15 de março de 2022.” s.d. https://informacoes.anatel.gov.br/legislacao/resolucoes/2022/1640-resolucao-750 (acesso em 03 de abril de 2023).

Anatel. *Resolução nº 460, de 19 de março de 2007.* 7 de junho de 2022. https://informacoes.anatel.gov.br/legislacao/resolucoes/22-2007/8-resolucao-460.

Androulaki, Elli, et al. “Hyperledger Fabric: A Distributed Operating System for Permissioned Blockchains.” *In Proceedings of the Thirteenth EuroSys Conference (EuroSys '18)*, 2018.

Azure, Microsoft. *Azure for Students.* 2023. https://azure.microsoft.com/pt-br/free/students/ (acesso em 02 de abril de 2023).

Chadwick, D., A. Otenko, e E. Ball. “Role-based access control with X.509 attribute certificates.” *IEEE Internet Computing*, 26 de março de 2003: 62-69.

COELHO, Igor M., e Vitor N. COELHO. “NeoCompiler Eco: experimentação de consenso em blockchain e contratos inteligentes.” *WORKSHOP DO TESTBED FIBRE (WFIBRE).* Porto Alegre: Sociedade Brasileira de Computação, 2021. 57-67.

CURIONI, Rosali Borges, e Antonio Carlos Gil. *Portabilidade Numérica: Impacto nas Operadoras, Clientes e Mercado.* s.d. http://www.rij.eng.uerj.br/professional/2009/pe091-01.pdf (acesso em 03 de abril de 2023).

*Docker e Docker Compose.* 02 de abril de 2023. https://hyperledger-fabric.readthedocs.io/pt/release-2.2/prereqs.html?highlight=docker#docker-e-docker-compose.

Docker Inc. *What is container.* s.d. https://www.docker.com/resources/what-container/ (acesso em 02 de abril de 2023).

*Fabric-CA Client’s CLI.* 2017. https://hyperledger-fabric-ca.readthedocs.io/en/release-1.4/clientcli.html#fabric-ca-client-s-cli (acesso em 16 de abril de 2023).

Foster, M., T. McGarry, e J. Yu. *Number Portability in the Global Switched Telephone Network (GSTN).* fevereiro de 2003. http://www.faqs.org/rfcs/rfc3482.html (acesso em 03 de abril de 2023).

GONÇALVES, Glauber Dias, Emanuel COUTINHO, e Allan Edgard Silva FREITAS. *Um Panorama da Pesquisa em Blockchain no Brasil.* SBC Horizontes. 2021. https://horizontes.sbc.org.br/index.php/2022/05/um-panorama-da-pesquisa-em-blockchain-no-brasil/ (acesso em 02 de abril de 2023).

Google - Protobuf. *Protocol Buffers.* 2023. https://protobuf.dev/ (acesso em 22 de abril de 2023).

*Hyperledger Architecture.* julho de 2019. https://www.hyperledger.org/wp-content/uploads/2017/08/Hyperledger\_Arch\_WG\_Paper\_1\_Consensus.pdf (acesso em 03 de abril de 2023).

Hyperledger Fabric - Commands. *Commands Reference.* 2023. https://hyperledger-fabric.readthedocs.io/en/release-2.5/command\_ref.html (acesso em 22 de abril de 2023).

Hyperledger Fabric - Key Concepts. *Hyperledger Fabric - Key Concepts.* 2023. https://hyperledger-fabric.readthedocs.io/en/release-2.4/key\_concepts.html (acesso em 03 de abril de 2023).

Hyperledger Fabric - PreReqs. *Fabric - PreReqs.* 2023. https://hyperledger-fabric.readthedocs.io/en/release-2.4/prereqs.html (acesso em 03 de abril de 2023).

Hyperledger. *Hyperledger Fabric Whitepaper.* 2023. https://www.hyperledger.org/wp-content/uploads/2020/03/hyperledger\_fabric\_whitepaper.pdf (acesso em 02 de abril de 2023).

Iansiti, Marco, e Karim R. Lakhani. “The truth about blockchain.” *Harvard business review*, 2017, 95.1 ed.: 118-127.

*IEEE vision and mission.* 2023. https://www.ieee.org/about/vision-mission.html (acesso em 03 de 2023ABRIL).

Krishnaswamy, Dilip, et al. “The Design of a Mobile Number Portability System on a Permissioned Private Blockchain Platform.” *2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, 2019: 90-94.

Linux Foundation. *Docker e Docker Compose.* 02 de abril de 2023. https://hyperledger-fabric.readthedocs.io/pt/release-2.2/prereqs.html?highlight=docker#docker-e-docker-compose.

—. *Hyperledger Fabric - Introdução.* 02 de abril de 2023. https://hyperledger-fabric.readthedocs.io/pt/release-2.2/whatis.html.

M. Belotti, N. Božić, G. Pujolle and S. Secci. “A Vademecum on Blockchain Technologies: When, Which, and How.” *IEEE Communications Surveys & Tutorials*, 2019: 3796-3838.

Microsoft Azure. *Azure.* 2023. https://azure.microsoft.com/pt-br/ (acesso em 02 de abril de 2023).

Mockapetris, P. . *RFC 1035 - Domain names - implementation and specification.* RFC Editor, 1987.

Moreira, Marcio Aurelio Ribeiro, Rogério Mendes, Flamaryon Guerin Gomes Borges, e Carvalho, Reginaldo Rosa de. “Uma Abordagem Do Processo De Portabilidade Numérica De Código De Acesso No Brasil Sob A Ótica Das Redes De Comutação.” *6o CONTECSI - CONGRESSO INTERNACIONAL DE GESTÃO DA TECNOLOGIA E SISTEMAS DE INFORMAÇÃO*, 05 de junho de 2009, 6a ed.

NAKAMOTO, Satoshi. “Bitcoin: A peer-to-peer electronic cash system.” *Decentralized business review*, 2008.

Nofer, Michael, Gomber, Peter, Hinz, Oliver, e Schiereck, Dirk. “Blockchain.” *Business & Information Systems Engineering*, 01 de 06 de 2017: 183-187.

OpenJS Foundation. *About Node.js.* s.d. https://nodejs.org/en/about (acesso em 03 de abril de 2023).

PIRES​, Mateus, Daniel SOUZA​, Rostand COSTA​, e Guido. LEMOS​. “Uma Abordagem Baseada em ​Brokers​ para Registro de Transações em Múltiplos Livros Razão Distribuídos.” *WORKSHOP EM BLOCKCHAIN: TEORIA, TECNOLOGIAS E APLICAÇÕES (WBLOCKCHAIN)*, 06 de maio de 2018.

PITTALIA, Prashant P. “A comparative study of hash algorithms in cryptography.” *International Journal of Computer Science and Mobile Computing*, junho de 2019: 147-152.

Rescorla, Eric . *RFC 8446: "The Transport Layer Security (TLS) Protocol Version 1.3".* Internet Engineering Task Force (IETF), 2018.

Ribeiro, Lucas, e Odorico Mendizabal. “Introdução à Blockchain e Contratos Inteligentes.” Universidade Federal de Santa Catarina, 2021.

Xu, Xun. “From cloud computing to cloud manufacturing.” *Robotics and Computer-Integrated Manufacturing*, fevereiro de 2012: 75-86.

**Apêndice 1**

Arquivo mychannel\_08.JSON

{

    "data": {

        "data": [

            {

                "payload": {

                    "data": {

                        "actions": [

                            {

                                "header": {

                                    "creator": {

                                        "id\_bytes": "",

                                        "mspid": "Org1MSP"

                                    },

                                    "nonce": "S1sObDs8y6KnB//lbH5jgUPSjNO0vXm4"

                                },

                                "payload": {

                                    "action": {

                                        "endorsements": [

                                            {

                                                "endorser": "",

                                                "signature": "MEUCIQDyM50dch/vBc+3idCxubEHa2+aEqivv7b5Bzc5EI3yQgIgN0HbhDi8arW+9AJ2cuqPXt54ydg3dq2JKqDGZF65D34="

                                            },

                                            {

                                                "endorser": "",

                                                "signature": "MEUCIQDq9EYfQig6R7IeVzK/Sh7BXoNwsu5UyNAwH95rdsz1agIgCbcR06KL/Fd3e/LzS7QV+H1j5Gb92/tG+IxaHgB7xD0="

                                            }

                                        ],

                                        "proposal\_response\_payload": {

                                            "extension": {

                                                "chaincode\_id": {

                                                    "name": "basic",

                                                    "path": "",

                                                    "version": "1.0"

                                                },

                                                "events": null,

                                                "response": {

                                                    "message": "",

                                                    "payload": "eyJNU0lTRE4iOiIwMjEyMjM0NTk4NyIsIk5PTUUiOiJEZXNpc3RlbGlvIE1hbmRpdGFsIiwiQ1BGIjoiMDI0OTcyMTMxMTEiLCJNQ0NNTkMiOiI3NDU2MCIsIk9QRVJBRE9SQSI6IkJSSVNBTkVUIn0=",

                                                    "status": 200

                                                },

                                                "results": {

                                                    "data\_model": "KV",

                                                    "ns\_rwset": [

                                                        {

                                                            "collection\_hashed\_rwset": [],

                                                            "namespace": "\_lifecycle",

                                                            "rwset": {

                                                                "metadata\_writes": [],

                                                                "range\_queries\_info": [],

                                                                "reads": [

                                                                    {

                                                                        "key": "namespaces/fields/basic/Sequence",

                                                                        "version": {

                                                                            "block\_num": "5",

                                                                            "tx\_num": "0"

                                                                        }

                                                                    }

                                                                ],

                                                                "writes": []

                                                            }

                                                        },

                                                        {

                                                            "collection\_hashed\_rwset": [],

                                                            "namespace": "basic",

                                                            "rwset": {

                                                                "metadata\_writes": [],

                                                                "range\_queries\_info": [],

                                                                "reads": [],

                                                                "writes": [

                                                                    {

                                                                        "is\_delete": false,

                                                                        "key": "02122345987",

                                                                        "value": "eyJNU0lTRE4iOiIwMjEyMjM0NTk4NyIsIk5PTUUiOiJEZXNpc3RlbGlvIE1hbmRpdGFsIiwiQ1BGIjoiMDI0OTcyMTMxMTEiLCJNQ0NNTkMiOiI3NDU2MCIsIk9QRVJBRE9SQSI6IkJSSVNBTkVUIn0="

                                                                    }

                                                                ]

                                                            }

                                                        }

                                                    ]

                                                }

                                            },

                                            "proposal\_hash": "itLcv//n5sZe9pw/Nun3yXmTwNqqXC7wUlhsTez1JTw="

                                        }

                                    },

                                    "chaincode\_proposal\_payload": {

                                        "TransientMap": {},

                                        "input": {

                                            "chaincode\_spec": {

                                                "chaincode\_id": {

                                                    "name": "basic",

                                                    "path": "",

                                                    "version": ""

                                                },

                                                "input": {

                                                    "args": [

                                                        "Q3JlYXRlQXNzZXQ=",

                                                        "MDIxMjIzNDU5ODc=",

                                                        "RGVzaXN0ZWxpbyBNYW5kaXRhbA==",

                                                        "MDI0OTcyMTMxMTE=",

                                                        "NzQ1NjA=",

                                                        "QlJJU0FORVQ="

                                                    ],

                                                    "decorations": {},

                                                    "is\_init": false

                                                },

                                                "timeout": 0,

                                                "type": "GOLANG"

                                            }

                                        }

                                    }

                                }

                            }

                        ]

                    },

                    "header": {

                        "channel\_header": {

                            "channel\_id": "mychannel",

                            "epoch": "0",

                            "extension": {

                                "chaincode\_id": {

                                    "name": "basic",

                                    "path": "",

                                    "version": ""

                                }

                            },

                            "timestamp": "2023-05-16T13:27:44.602Z",

                            "tls\_cert\_hash": "7snRvDEPDMZNurGdCLCJq6B8WPoLoRreY/AjE22fASw=",

                            "tx\_id": "23b490e48015044648c8b47d12240efba345407a057adbd29d529668a4ddb28e",

                            "type": 3,

                            "version": 1

                        },

                        "signature\_header": {

                            "creator": {

                                "id\_bytes": "",

                                "mspid": "Org1MSP"

                            },

                            "nonce": "S1sObDs8y6KnB//lbH5jgUPSjNO0vXm4"

                        }

                    }

                },

                "signature": "MEQCIEpV3MVB+8y1QWMG+8NLNCB6pKzGv4y3m/AJ6Tw4lh4sAiA9MolNctOIyrAm8nkQsc9QsYktzOf1bScmfBqP3dMt0A=="

            }

        ]

    },

    "header": {

        "data\_hash": "mUKgeBkW1QDmrqsWF6qqkG3erLp3NqITxVTInsU6TIM=",

        "number": "8",

        "previous\_hash": "X29GN47l0LNXKCAt4T5capNAkSzFTyCUk+TyxUUj29U="

    },

    "metadata": {

        "metadata": [

            "",

            "CgIIAg==",

            "AA==",

            "",

            "CiD1Otm/0O2AHKOb/hAZVANmrlSv3WX0wpIKiQz1h8KgLg=="

        ]

    }

}