

A IMPLEMENTAÇÃO DO BLOCKCHAIN HYPERLEDGER FABRIC EM AMBIENTE LINUX UTILIZANDO CONTAINERS DOCKER¹

Júlio Sérgio Quadros dos Santos²

Adilso Nunes de Souza³

Élder Francisco Fontana Bernardi⁴

RESUMO

O armazenamento de dados de forma segura é uma preocupação em diversos setores da economia. No modelo atual de gestão de segurança da informação, são necessárias medidas preventivas para evitar fraudes. Os sistemas de armazenamento de dados que utilizam o padrão blockchain surgem como alternativa para garantir nativamente proteção contra alteração de informações, proporcionando a centralização das informações, descentralização do armazenamento, auditabilidade e transparência no processo de armazenamento. Características que tem a capacidade de viabilizar projetos em benefício da sociedade em áreas como a saúde, educação, transportes, alimentos e meio ambiente. O blockchain do Bitcoin e a rede Ethereum são amplamente utilizados em sistemas de criptomoedas, na geração de moedas digitais e para o registro de transações financeiras, porém, sem padrões definidos para utilização em outros setores. A Linux Foundation, conhecida pelo estabelecimento de padrões de software para utilização na indústria, vêm, em parceria com empresas multinacionais, apresentar uma ampla gama de frameworks e ferramentas para implementação da tecnologia blockchain Hyperledger Fabric. Desta forma, o presente estudo tem como objetivo identificar as necessidades tecnológicas para a implementação do Blockchain Hyperledger Fabric, a estrutura de hardware utilizada para a implementação da tecnologia, o sistema operacional, o sistema de virtualização com os serviços necessários para seu funcionamento, e uma breve introdução sobre a integração entre o Hyperledger Fabric e sistemas externos por meio de uma *Application Programming Interface* (API).

Palavras-chave: Armazenamento de Informações, Segurança, Auditabilidade.

¹ Trabalho de Conclusão de Curso (TCC) apresentado ao Curso de Tecnologia em Sistemas para Internet do Instituto Federal Sul-rio-grandense, Campus Passo Fundo, como requisito parcial para a obtenção do título de Tecnólogo em Sistemas para Internet, na cidade de Passo Fundo, em 2017.

² Graduando em Tecnologia em Sistemas para Internet (julioquadros@gmail.com).

³ Orientador do trabalho, graduado no Curso Superior de Tecnologia Em Processamento de Dados, especialista em Desenvolvimento de sistemas de informação com ênfase em desenvolvimento para Web, mestre em Engenharia Civil e Ambiental.

⁴ Co-Orientador do trabalho, graduado em Ciência da Computação, mestre em Ciência da Computação, com aperfeiçoamento em Formação Pedagógica.

1 INTRODUÇÃO

Em sua essência, Blockchain é uma tecnologia que armazena transações permanentemente, de forma que as informações não podem ser apagadas posteriormente, somente atualizadas, mantendo para sempre os registros anteriores (MOUGAYAR, 2016).

Segundo Tapscott (2014), esta nova forma de armazenamento de dados de transações financeiras pode ser programada para armazenar virtualmente qualquer coisa com valor e importância para a humanidade, como certidões de nascimento e de óbito, certidões de casamento, escrituras e títulos de propriedade, diplomas de cursos educacionais, dados contábeis, registros médicos, requerimentos de seguros, registros de votos, rastreabilidade de alimentos, entre outras coisas que possam ser expressas em código de computador.

Segundo artigo *Hyperledger Goes to School*, publicado no site da Nasdaq, os blockchains atuais, utilizados pelas moedas digitais com maior participação no mercado, como o Blockchain do Bitcoin (BTC) e a rede Ethereum do Ether (ETH) não possuem um padrão. Por este motivo, a Linux Foundation que é referência em implementação de padrões, decidiu iniciar o projeto Hyperledger como padronização para a indústria (BRADBURY, 2017).

Diante do exposto, o presente estudo tem como objetivo identificar as necessidades tecnológicas para a implementação do Blockchain Hyperledger Fabric. Inicialmente será descrita a estrutura de hardware utilizada para a implementação da tecnologia, seguido pelo sistema operacional e o sistema de virtualização com os serviços necessários para o funcionamento do Blockchain Hyperledger Fabric. Por fim, será apresentada uma breve introdução sobre a integração entre o Blockchain Hyperledger Fabric e sistemas externos por meio de uma *Application Programming Interface* (API).

2 REFERENCIAL TEÓRICO

O processo de implementação do Blockchain Hyperledger Fabric depende da utilização de tecnologias relacionadas, como sistemas operacionais, sistemas de virtualização e softwares para o funcionamento do Blockchain Hyperledger Fabric (HYPERLEDGER).

As tecnologias utilizadas para a instalação, compilação e implementação dos serviços relacionados, regras de negócios e os sistemas de comunicação com sistemas externos, fazem parte das indicações da Linux Foundation e da IBM, que inicialmente foi a idealizadora do Hyperledger Fabric, cedendo o projeto posteriormente a comunidade de software livre.

2.1 BLOCKCHAIN

O blockchain é uma das tecnologias necessárias para o funcionamento da moeda digital Bitcoin. Porém, não é a única tecnologia necessária para o funcionamento do Bitcoin (NORTON, 2017).

Blockchain é o livro razão que serve como registro de todas as transações de Bitcoins já executadas. O Blockchain cresce constantemente em virtude dos blocos gerados a cada 10 minutos pelos mineradores. Os blocos gerados são inclusos no blockchain de forma linear, e cronológica, mantendo uma cópia completa dos dados em cada nó da cadeia do blockchain, que é atualizada automaticamente pelo sistema quando um novo minerador se afilia a rede Bitcoin (SWAN, 2015).

Segundo Norton (2017), tradicionalmente, quando uma transação entre duas partes é salva em um banco de dados, existe a necessidade de um intermediário que certifique a veracidade das informações, como o governo, um banco ou outra empresa, evitando que uma das partes possa levar vantagem se tiver a possibilidade de alteração dos dados registrados. Atualmente, novos sistemas de blockchains foram desenvolvidos para suprir necessidades de inúmeras áreas, tornando possível com que o blockchain possa ser utilizado para garantir a confiabilidade no armazenamento de dados, que não possam ser alterados depois de salvos.

A possibilidade de utilização do blockchain, para a automação de processos burocráticos, abre um novo horizonte para serviços que até então não poderiam ser explorados, devido a necessidade de intermediários. Por exemplo, serviços como o Mercado Pago do site de compras Mercado Livre, que centraliza operações de compra e venda entre pessoas, atua como intermediário para garantir que o comprador receba o produto conforme o anúncio no site e o vendedor receba o valor acordado após a entrega do produto, dessa forma, o comprador escolhe um produto e envia o dinheiro para ao Mercado Pago, o Mercado Pago informa o vendedor sobre a compra autorizando o envio do produto, e assim que o comprador confirmar

o recebimento do produto nas condições pré-estabelecidas, o Mercado Pago libera o pagamento para o vendedor. Através da definição de regras de negócios em *Smart Contracts Chaincode*, os parâmetros definidos podem ser acompanhados pelo serviço de consenso do Blockchain Hyperledger Fabric, eliminando a necessidade de intermediários como o Mercado Pago, permitindo com que o Blockchain possa identificar quando as regras são satisfeitas, e atuar conforme os contratos estabelecidos e aceitos pelas duas partes, como no exemplo citado, recebendo o valor do comprador e liberando para o vendedor no momento em que o comprador confirmar o recebimento do produto.

2.2 HYPERLEDGER FABRIC

O Hyperledger Fabric faz parte da família de softwares open source mantidos pela Linux Foundation, é um *framework* com o propósito de criação da infraestrutura necessária para a implementação de blockchains privados.

Inicialmente foram comparados com o blockchain Hyperledger Fabric, os blockchains que servem como base para as moedas digitais mais populares: o blockchain do Bitcoin e a rede Ethereum. Entre as principais características que destacam o Hyperledger Fabric das tecnologias comparadas, podem ser citados o gerenciamento pela Linux Foundation, não ser baseado em criptomoedas, permitir a criação de redes privadas com gestão de usuários através uso de certificados e sua modularidade, o que permite agregar diferentes serviços como diferentes algoritmos de consenso ou diferentes bases de dados (Tabela 1).

Tabela 1 – Comparação entre os blockchains do Bitcoin, Ethereum e o Hyperledger Fabric.

Blockchain	Bitcoin	Ethereum	Hyperledger Fabric
Descrição	Blockchain para transações	Blockchain para uso geral	Blockchain para uso geral
Gerenciamento	Desenvolvedores Bitcoin	Desenvolvedores Ethereum	Linux Foundation
Baseado em Criptomoeda	Sim (BTC)	Sim (ETH)	Não
Permissões	Pública	Pública	Privada
Pseudoanônimo	Sim	Não	Não
Auditável	Sim	Sim	Sim
Imutável	Sim	Sim	Sim
Modularidade	Não	Não	Sim

Smart Contracts	Não	Sim	Sim
Protocolo de Consenso	PoW (Mineração)	PoW (Mineração)	Apache Kafka

Fonte: Informações extraídas da documentação oficial HYPERLEDGER.

O Hyperledger Fabric possui uma estrutura que exige a verificação dos dados antes do armazenamento do bloco no Blockchain, fazendo com que somente os dados que satisfaçam as características estabelecidas pelo Chaincode possam ser persistidos, característica relevante em virtude de que os dados uma vez armazenados no Blockchain, não podem ser alterados (HYPERLEDGER).

2.3 GO LANG

A linguagem de programação Go surgiu como o resultado dos esforços do Google quanto a necessidade de uma linguagem de programação que pudesse trabalhar com as tecnologias de processamento paralelo, como em processadores de múltiplos núcleos lançados pela Intel em 2006 (GOLANG).

Para o projeto, a linguagem Go é indicada por ser a primeira linguagem nativa para a escrita de Chaincodes, necessários para a definição de regras de negócios para o Hyperledger Fabric.

2.4 NODE.JS

Node.JS é uma plataforma de desenvolvimento rápido e escalável para aplicações baseadas em rede que rodam no lado do servidor. O Node.JS utiliza o motor de processamento de JavaScript V8, utilizado pelo navegador Google Chrome (NODEJS).

2.5 SMART CONTRACT CHAINCODE

Chaincode é um programa escrito na linguagem de programação Go, Node.JS ou Java, que define as regras de processamento dos dados introduzidos no Blockchain Hyperledger, sendo necessário para o correto funcionamento da estrutura do Blockchain. O Chaincode roda como um serviço seguro isolado, não permitindo a comunicação entre o Chaincode e os processos dos nós da rede (HYPERLEDGER).

Um Chaincode geralmente gerencia a lógica de negócios introduzida no Blockchain, como um *Smart Contract*, que define as condições de negócios aceitas pelas partes envolvidas em uma negociação através do Blockchain Hyperledger.

2.6 LINUX

Linux basicamente é um sistema operacional, um sistema operacional é um software que gerencia todos os recursos de hardware associados com computadores ou notebooks. O Linux gerencia a comunicação entre os softwares instalados e o hardware, sem um sistema operacional os computadores não funcionam (LINUX).

2.7 REST

REST significa *Representational State Transfer*, ou, traduzido para o português, Transferência do Estado Representacional, um conjunto de abstrações dos padrões adotados pela arquitetura *World Wide Web* (WWW), sendo assim, um padrão amplamente utilizado para comunicação entre computadores na arquitetura cliente-servidor (FIELDING, 2000).

O padrão REST faz com que clientes e servidores possam interagir, inclusive de formas complexas, sem a necessidade de um lado precisar conhecer a arquitetura utilizada pelo outro lado, possibilitando a homogeneização na comunicação. A principal restrição no uso do REST, é de que os dois lados precisam chegar a um acordo de qual o meio deve ser utilizado para a comunicação, nesse caso, o protocolo HTTP com os comandos GET, POST, PUT e DELETE.

O padrão REST é utilizado como padrão de comunicação entre os sistemas externos e a API, responsável pelo armazenamento dos dados no Blockchain Hyperledger Fabric.

2.8 JSON

JavaScript Object Notation (JSON) é um formato de troca de informações baseada em texto, leve e independente de linguagem de programação, derivado do

padrão da linguagem de programação ECMAScript. JSON define um pequeno conjunto de regras para a representação estruturada transportável de dados (JSON).

O formato JSON permite com que o projeto possua um ambiente de troca de informações através da API REST, possibilitando a comunicação entre os sistemas externos e o Chaincode do Blockchain Hyperledger.

2.9 DOCKER E DOCKER COMPOSE

Docker é uma empresa que desenvolve sistemas de virtualização que possibilitam, através da utilização de containers, o acesso de aplicações em ambientes híbridos, possibilitando que os softwares funcionem corretamente, independente do sistema operacional utilizado (DOCKER).

O projeto de implementação do Blockchain Hyperledger Fabric utiliza o Docker como solução de virtualização, através de containers Docker, para os serviços necessários para o correto funcionamento do Hyperledger.

3 RESULTADOS

3.1 METODOLOGIA

Inicialmente, o objetivo do estudo seria a utilização da estrutura da IBM Blockchain como base de testes, com o intuito de avaliar os prós e contras da utilização do Blockchain Hyperledger para armazenamento de dados cadastrais de pessoas físicas. Porém, devido a descontinuidade da disponibilização dos serviços IBM Blockchain de forma gratuita, o projeto teve a necessidade de ser reavaliado.

Como forma de dar continuidade aos estudos propostos, tornou-se necessário o estudo da implementação da tecnologia Hyperledger Fabric como base para os estudos, e, para evitar que o projeto sofresse intervenções, optou-se pela utilização de software livre, em infraestrutura própria, onde possa ser observada a continuidade das versões de software, evitando assim, necessidades de alterações nos códigos por incompatibilidade com futuras versões.

Para avaliação do hardware necessário, inicialmente foram efetuados testes em três estruturas distintas, sendo elas:

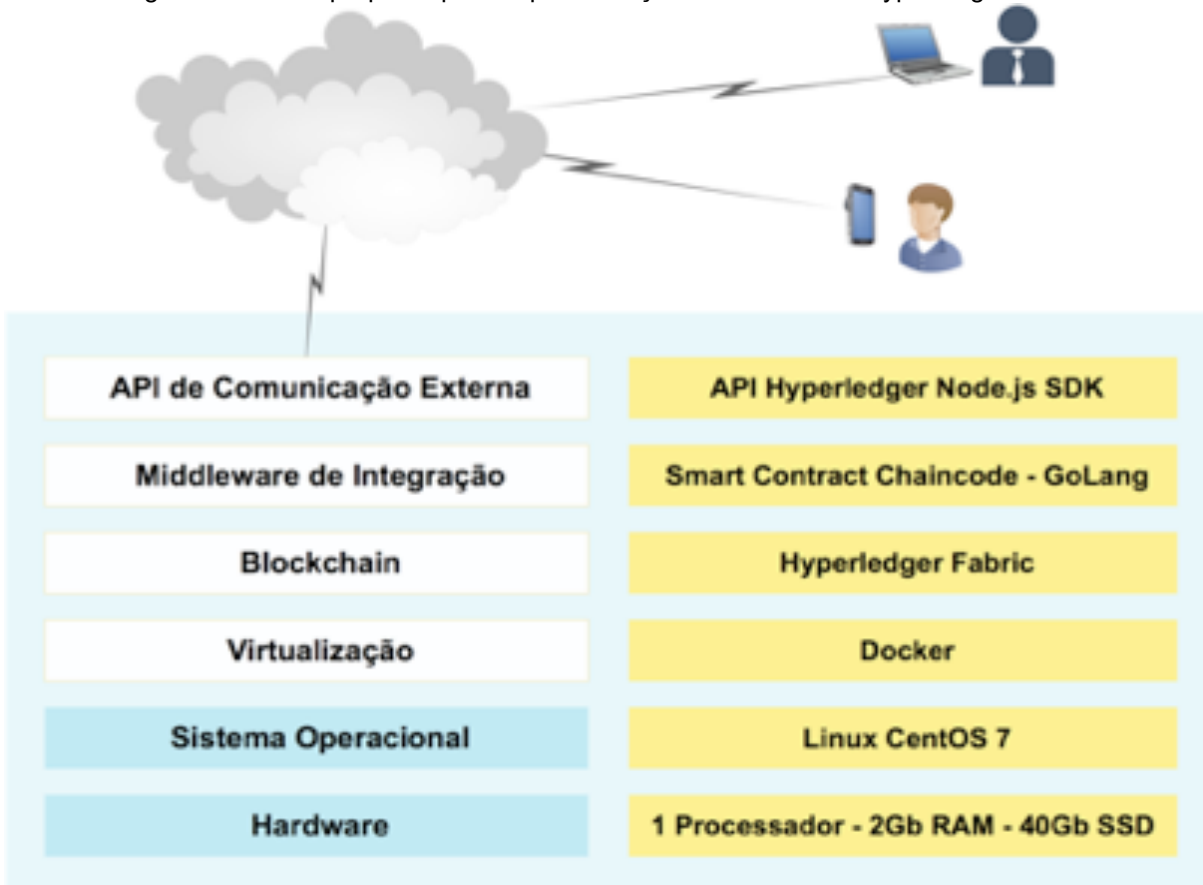
- Macbook Air com processador I5 e 4Gb de memória, rodando o Sistema Operacional Mac OSX Sierra 10.2.3;
- Raspberry Pi 3 Model B com processador 1.2GHz 64bit quad-core ARMv8 e 1Gb de memória RAM, rodando o Sistema Operacional Raspbian Stretch Lite, Kernel 4.9;
- Máquina virtual em nuvem, utilizando os serviços da Digital Ocean, onde foi contratado inicialmente um servidor com 1 processador, 512 Mb de memória RAM, rodando o Sistema Operacional Centos 7.

Devido a possibilidade de expansão de recursos de forma facilitada, foi escolhida a utilização da máquina virtual na nuvem da Digital Ocean com a utilização do Sistema Operacional Linux CentOS 7. A utilização do Sistema Operacional Mac OSX apresentou incompatibilidades de softwares para a compilação dos códigos, tornando a implementação mais complexa que o esperado. Já o uso do Raspberry Pi como hardware também foi descartado após os primeiros testes, devido a não existência de imagens Docker no momento para a arquitetura de processadores ARM.

Após os testes realizados para implementação do projeto, foram configurados os sistemas básicos de infraestrutura para um blockchain privado, utilizando o sistema operacional Linux CentOS 7, as imagens de containers Docker da Hyperledger Fabric, as imagens oficiais são disponibilizadas no repositório da Hyperledger no Docker (<https://hub.docker.com/u/hyperledger/>), e os exemplos disponibilizados pela Hyperledger para verificação do ambiente.

Posteriormente foi desenvolvido um *Smart Contract* Chaincode, com os ajustes necessários para a estrutura de dados referente aos dados de cadastro de uma pessoa. Na sequência foi desenvolvida uma API REST de integração entre o blockchain Hyperledger e sistemas externos, efetuando a comunicação utilizando JSON (Figura 1).

Figura 1: Modelo proposto para implementação do Blockchain Hyperledger Fabric.



Fonte: Desenvolvido pelo autor

3.2 HARDWARE

Para o desenvolvimento do estudo, foram consideradas as necessidades de hardware para a implementação do Hyperledger Fabric no sistema operacional Linux CentOS 7.

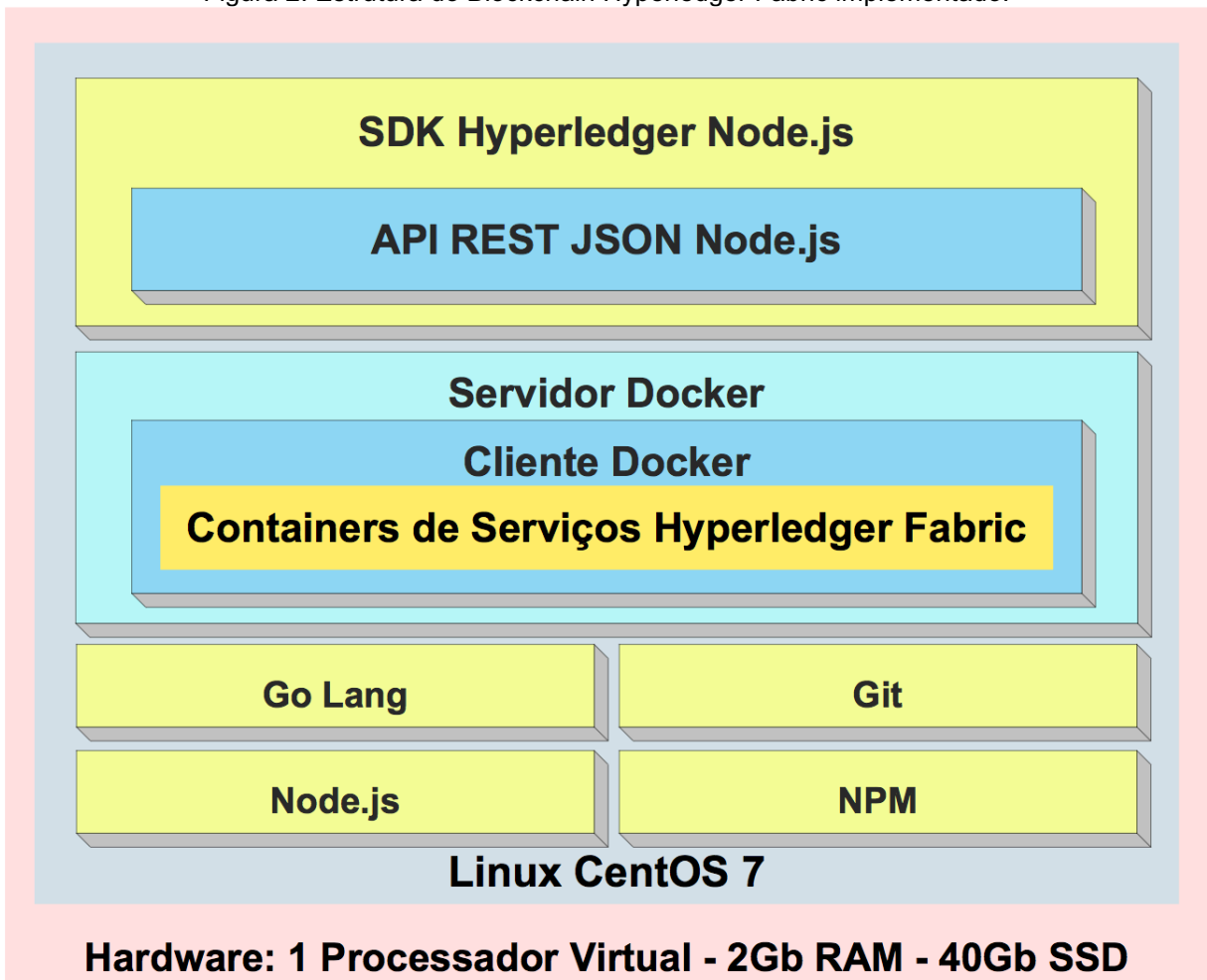
Essencialmente, para o funcionamento da solução, após testes com capacidades inferiores de hardware, iniciando com 512Mb de memória RAM, foram ampliadas as configurações até o momento onde as necessidades de hardware foram satisfeitas utilizando um servidor com 1 processador virtual, 2 Gb de memória RAM, e 40 Gb de memória SSD.

3.3 ESTRUTURA LÓGICA

Para a base da estrutura foi utilizado o Sistema Operacional Linux CentOS 7 com os pré-requisitos necessários para o funcionamento dos containers Docker do Hyperledger Fabric como a linguagem de programação Go Lang, Node.js, Git, NPM,

o servidor e cliente Docker e o SDK Hyperledger Node.js (Figura 2), de forma a inicializar os serviços necessários para o funcionamento da estrutura necessária para o blockchain Hyperledger Fabric, as versões dos softwares utilizados devem ser compatíveis com o Hyperledger Fabric, para o projeto foram utilizados os softwares Go Lang versão 1.6.3, Node.js versão 6.11.1, Git versão 1.8.3.1, NPM versão 3.10.10, o servidor Docker versão 17.07.0-ce, o cliente Docker versão 17.07.0-ce (Figura 3).

Figura 2: Estrutura do Blockchain Hyperledger Fabric implementado.



Fonte: Desenvolvido pelo autor

Figura 3 – Versões dos softwares utilizados para o funcionamento do Hyperledger Fabric

Linux CentOS 7 Linux 3.10.0-514.26.2.el7.x86_64	
Docker	Versão do Software Cliente: 17.07.0-ce/Versão do Software Servidor: 17.07.0-ce
Git	Versão do Software: 1.8.3.1
Golang	Versão do Software: 1.6.3
Node.js	Versão do Software: 6.11.1
NPM	Versão do Software: 3.10.10

Fonte: Desenvolvido pelo autor

3.4 CONTAINERS DOCKER

Para o funcionamento do blockchain Hyperledger Fabric, foram inicializados os containers que rodam os serviços necessários para a leitura e escrita (Fabric-Peer), ferramentas (Fabric-Tools), gerenciamento de identidade (Fabric-CA), serviço de consenso (Fabric-Orderer), base de dados (Fabric-CouchDB) e as regras de negócio (Chaincode) (Figura 4).

Figura 4 – Containers necessários para inicialização do blockchain Hyperledger Fabric.

Servidor Docker 17.07.0-ce	
Fabric-Peer	<p>Cliente Docker 17.07.0-ce</p> <p>- Leitura e escrita no Blockchain</p>
Fabric-Tools	<p>Cliente Docker 17.07.0-ce</p> <p>- Ferramentas do Cliente Hyperledger</p>
Fabric-CA	<p>Cliente Docker 17.07.0-ce</p> <p>- Registro, Renovações e Revogações de Identidades.</p>
Fabric-Orderer	<p>Cliente Docker 17.07.0-ce</p> <p>- Consenso baseado em Apache Kafka</p>
Fabric-CouchDB	<p>Cliente Docker 17.07.0-ce</p> <p>- Base de dados</p>
Chaincode	<p>Cliente Docker 17.07.0-ce</p> <p>- Regras de negócio do Blockchain</p>

Fonte: Desenvolvido pelo autor

3.5 ESTRUTURA DO CHAINCODE

O Chaincode é nativamente desenvolvido utilizando a linguagem Go, e permite a criação dos Contratos Inteligentes (*Smart Contracts*) Chaincode. A importação do pacote Shim (github.com/hyperledger/fabric/core/chaincode/shim) fornece APIs que permitem a comunicação com a base de serviços Hyperledger (Figura 5).

Figura 5: Estrutura básica de Chaincode para implementação do sistema.

```

package main
import (
    "bytes"
    "encoding/json"
    "fmt"
    "strconv"
    "github.com/hyperledger/fabric/core/chaincode/shim"
    sc "github.com/hyperledger/fabric/protos/peer"
)
type SmartContract struct {
}
type Person struct {
    PersonID      string `json:"personId"`
    PersonFirstName string `json:"personName"`
    PersonMiddleName string `json:"personMiddleName"`
    PersonLastName string `json:"personLastName"`
}
func (s *SmartContract) Init(APIStub shim.ChaincodeStubInterface) sc.Response {
    return shim.Success(nil)
}
func (s *SmartContract) Invoke(APIStub shim.ChaincodeStubInterface) sc.Response {
    if function == "initLedger" {
        return s.initLedger(APIStub)
    }
}
func (s *SmartContract) initLedger(APIStub shim.ChaincodeStubInterface) sc.Response {
    people := []Person{
        {
            PersonID: "0000000001",
            PersonFirstName: "Julio",
            PersonMiddleName: "Sergio",
            PersonLastName: "Quadros dos Santos"
        },
    }
    personIDAsBytes, _ := json.Marshal(people[0])
    APIStub.PutState("PERSON0001", personIDAsBytes)
    return shim.Success(nil)
}
func main() {
    shim.Start(new(SmartContract))
}

```

Fonte: Desenvolvido pelo autor.

3.5.1 Init

Para a inicialização do *Ledger* (Livro Razão), é necessário que o Chaincode informe os dados corretos de funcionamento. Por este motivo, a utilização do `Init` corresponde aos dados iniciais para a primeira utilização do Livro Razão.

3.5.2 Invoke

O `Invoke` é responsável por receber as solicitações externas, tratando os fluxos necessários e respondendo aos sistemas externos os dados solicitados.

3.5.3 Func (Função)

As funções são criadas no Chaincode para simplificar as chamadas que podem ser executadas com maior frequência, tornando a configuração simplificada e podendo atender melhor as solicitações externas, que são acionadas através do Invoke.

3.6 NODE.JS

Para a comunicação entre os sistemas externos e o Fabric-Orderer foi desenvolvida uma API com o padrão REST, transmitindo os dados em formato JSON, o que possibilita a integração com sistemas externos que possibilitem a utilização de comunicação no padrão REST.

Para o desenvolvimento em Node.js é necessária a utilização do SDK Hyperledger Node.js, disponibilizado pela Hyperledger. Para a implementação do código, são necessárias também configurações de conexão do Node.js com os serviços do Hyperledger Fabric (Figura 6), a inicialização do serviço para responder as requisições externas (Figura 7), e a função de consulta de pessoas (Figura 8), e.

Figura 6: Definição das variáveis utilizadas para conexão com os serviços Hyperledger Fabric

```
'use strict';

var hfc = require('fabric-client');
var path = require('path');
var bodyParser = require('body-parser');
var util = require('util');
var express = require('express');
var app = express();

var options = {
  wallet_path: path.join(__dirname, './creds'),
  user_id: 'PeerAdmin',
  channel_id: 'mychannel',
  chaincode_id: 'cadastro_pessoa',
  peer_url: 'grpc://localhost:7051',
  event_url: 'grpc://localhost:7053',
  orderer_url: 'grpc://localhost:7050',
  network_url: 'grpc://localhost:7051',
};

var channel = {};
var client = null;
var targets = [];
var tx_id = null;

var app_port = 3000;
```

Fonte: Desenvolvido pelo autor.

Figura 7: Inicialização da API para ouvir as solicitações externas na porta 3000.

```
function startListener() {
  console.log("Inicializando o Serviço API REST na porta " + app_port);
  app.listen(app_port);
  console.log("A API REST agora está rodando na porta " + app_port + "\n");
}

startListener();
```

Fonte: Desenvolvido pelo autor.

Figura 8: Função para consulta do registro de uma pessoa pelo código cadastrado.

```
app.get("/consultaPessoa/:var", function(req, res) {
  var stateVar = req.params.var;

  Promise.resolve().then(() => {
    client = new hfc();
    return hfc.newDefaultKeyValueStore({ path: options.wallet_path });
  }).then((wallet) => {
    client.setStateStore(wallet);
    return client.getUserContext(options.user_id, true);
  }).then((user) => {
    channel = client.newChannel(options.channel_id);
    channel.addPeer(client.newPeer(options.network_url));
    return;
  }).then(() => {
    var transaction_id = client.newTransactionID();
    const request = {
      chaincodeId: options.chaincode_id,
      txId: transaction_id,
      fcn: 'queryPerson',
      args: [stateVar]
    };
    return channel.queryByChaincode(request);
  }).then((query_responses) => {
    res.status(200).json({ "value": query_responses[0].toString() });
  }).catch((err) => {
    console.error("Erro: ", err);
  });
});
```

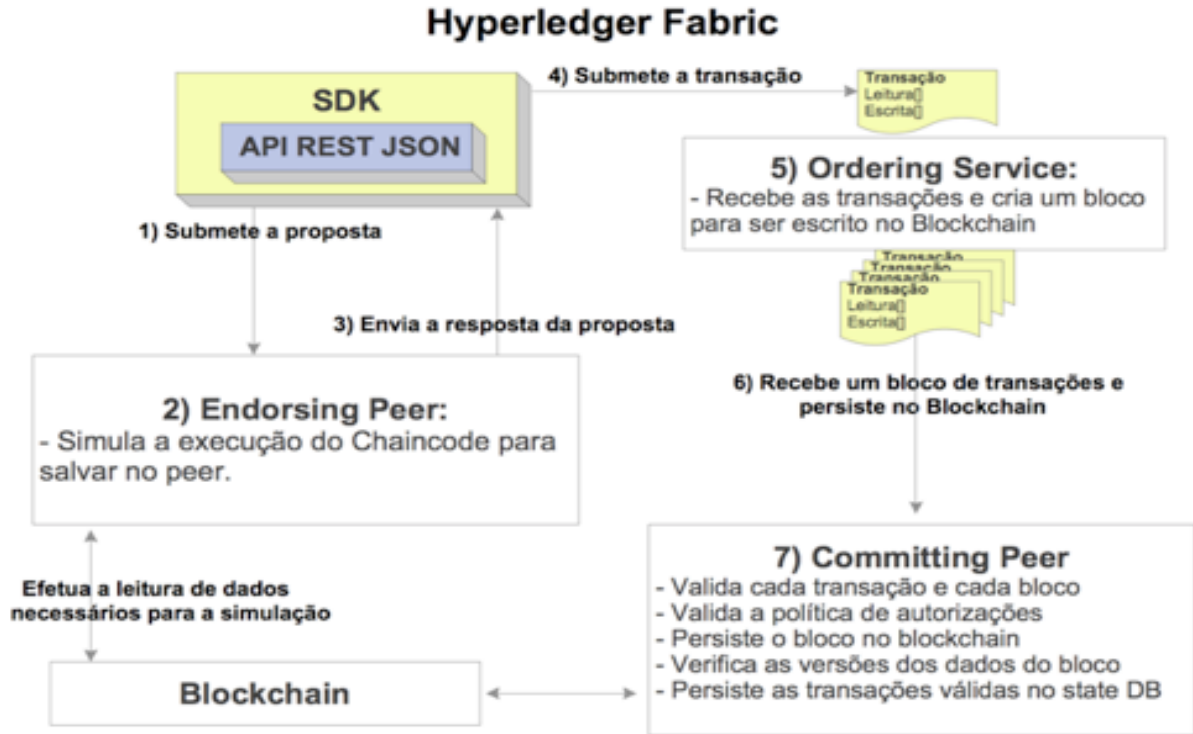
Fonte: Desenvolvido pelo autor

3.7 Considerações

Com o estudo da implementação, foi possível definir a estrutura de software e hardware necessária para o funcionamento do Blockchain Hyperledger Fabric em ambiente Linux, com a utilização de imagens Docker, e a identificação dos softwares e as respectivas versões necessários para o funcionamento do blockchain. Avançando para o desenvolvimento de aplicações, também foram identificadas as necessidades para a criação das regras de negócio e das estruturas de dados necessárias, através do uso de *Smart Contracts* Chaincode. Já na camada de interconexão, foram definidas as necessidades de desenvolvimento da API em Node.js para troca de informações com sistemas externos através de dados JSON (Figura 8). Foram realizados testes utilizando o software Insomnia como forma de

verificar a comunicação no padrão REST, enviando os dados necessários para o cadastro de uma pessoa (Figura 9), as mensagens geradas pela API no terminal Linux descrevem o sucesso das transações (Figura 10) e uma consulta do registro efetuado utilizando o software Insomnia passando o parâmetro do registro a ser consultado no padrão REST (Figura 11).

Figura 8: Dinâmica de funcionamento dos Serviços Hyperledger Fabric:



Fonte: Adaptado de Hyperledger Fabric Docs.

Figura 9: Cadastro de uma pessoa utilizando JSON através da API REST.

Insomnia – Arcani Blockchain

POST http://138.68.0.248:3000/api/cadastraPessoa Send 200 OK TIME 865 ms SIZE 120 B

JSON Auth Query 5 Header 1 Docs

URL PREVIEW

```
http://138.68.0.248:3000/api/cadastraPessoa?Key=PERSON20&personId=000000020&personFirstName=Ari&personMiddleName=João&personSurname=Ferreira
```

Key	PERSON20
personId	000000020
personFirstName	Ari
personMiddleName	João
personSurname	Ferreira

New name New value

Preview Header 8 Cookie Timeline

```
1 {  
2   "Key": "PERSON20",  
3   "personId": "000000020",  
4   "personFirstName": "Ari",  
5   "personMiddleName": "João",  
6   "personSurname": "Ferreira"  
7 }
```

Fonte: Desenvolvido pelo autor.

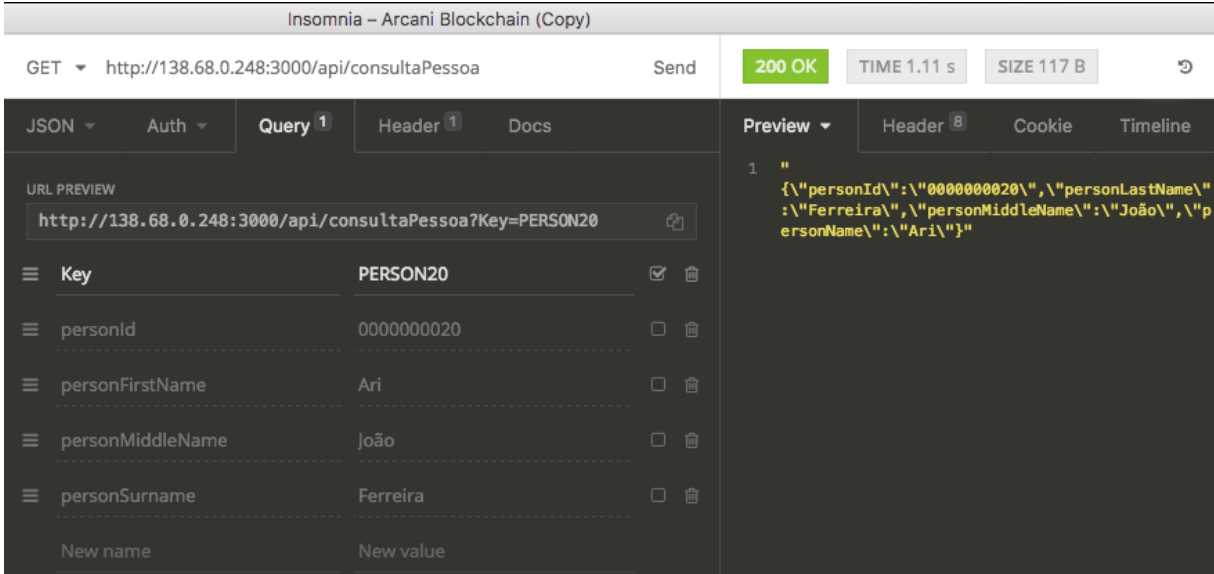
Figura 10: Mensagens geradas no console Linux pelo Node.js.

```
[root@arcaniBlockchain certidao_nascimento]# node cn_api_rest.js
Starting WebApp on port 3000
WebApp is now listening on port 3000

xRequisitado POST: PERSON20 - 0000000020 - Ari - João - Ferreira
Create a client and set the wallet location
Set wallet path, and associate user PeerAdmin with application
Check user is enrolled, and set a query URL in the network
Assigning transaction_id: 26c09c06f9e6406cd128438cd74e32a5bb31ddb4aa13e99f5aef6b28794784f
transaction proposal was good
Successfully sent Proposal and received ProposalResponse: Status - 200, message - "OK", metadata - "", endorsement signature: 0D 9
kZ" LgWk;e;
XnJ
info: [EventHub.js]: _connect - options {}
The transaction has been committed on peer localhost:7053
event promise all complete and testing complete
Successfully sent transaction to the orderer.
```

Fonte: Desenvolvido pelo autor.

Figura 11: Consulta de dados de uma pessoa através da API REST.



The screenshot shows a REST client interface with the following details:

- Method:** GET
- URL:** http://138.68.0.248:3000/api/consultaPessoa
- Status:** 200 OK
- Time:** 1.11 s
- Size:** 117 B
- Query:** Key=PERSON20
- Response (JSON):**

```
{
  "personId": "0000000020",
  "personLastName": "Ferreira",
  "personMiddleName": "João",
  "personName": "Ari"
}
```
- Table of Headers:**

Key	Value
Key	PERSON20
personId	0000000020
personFirstName	Ari
personMiddleName	João
personSurname	Ferreira
New name	New value

Fonte: Desenvolvido pelo autor.

Com a estrutura implementada, é possível persistir e consultar os dados, tendo em vista que sistemas blockchain não permitem alterações das informações após a geração de um bloco. A comunicação entre sistemas externos e o blockchain, através da API REST utilizando dados JSON simplifica a integração de sistemas legados, sendo que maior parte das plataformas de desenvolvimento possui mecanismos de comunicação utilizando esse padrão.

4 CONSIDERAÇÕES FINAIS

Com o conhecimento adquirido no projeto, é possível iniciar o planejamento de sistemas relacionados ao conceito de Internet das Coisas, sendo esta área uma das mais beneficiadas com a utilização da tecnologia blockchain.

Existe a possibilidade de elaboração de artigos, descrevendo as estruturas necessárias para a implementação de serviços de armazenamento de dados obtidos através de sensores, e aspectos necessários para a implementação de comunicação M2M (Machine-to-Machine), utilizando as tecnologias disponibilizadas pela *Linux Foundation*.

5 ABSTRACT

Securely storing data is a concern in many sectors of the economy. In the current model of information security management, preventive measures are necessary to avoid fraud. Data storage systems in the blockchain pattern appear as an alternative to ensure immutability of information, providing centralized information, decentralized storage, auditing, and transparency in the storage process. Characteristics that have the capacity to make feasible projects for the benefit of society in areas such as health, education, transportation, food and environment. The Bitcoin blockchain and the Ethereum network are widely used in cryptocurrency systems, in the generation of digital coins and in the recording of financial transactions, but without defined standards for use in other sectors. The Linux Foundation, known for establishing software standards for use in the industry, has partnered with multinational companies through the blockchain infrastructure project Hyperledger, to present a wide range of frameworks and tools for implementing such technology. In this way, the present study aims to identify the technological needs for the implementation of Blockchain Hyperledger Fabric, the hardware structure used to implement the technology, the operating system, the virtualization system with the services necessary for its operation, and a brief introduction on the integration between Hyperledger Fabric and external systems through an Application Programming Interface (API).

Keywords: Information Storage, Security, Auditability.

6 REFERÊNCIAS

BRADBURY, Danny. *Hyperledger Goes to School*. Disponível em: <<http://www.nasdaq.com/article/hyperledger-goes-to-school-cm872546>>. Acesso em 10 nov. 2017.

DOCKER. Disponível em: <<https://www.docker.com/what-docker>>. Acesso em 15 nov. 2017.

FIELDING, Roy Thomas. *Architectural Styles and the Design of Network-based Software Architectures*. Doctoral dissertation, University of California, Irvine, 2000.

GOLANG. Disponível em: <<https://golang.org/>>. Acesso em 15 nov. 2017.

HYPERLEDGER. Disponível em: <<http://hyperledger-fabric.readthedocs.io/en/latest/chaincode4ade.html>>. Acesso em: 15 nov. 2017.

JSON. Disponível em: <<https://tools.ietf.org/html/rfc7159>>. Acesso em 15 nov. 2017.

LINUX. Disponível em: <<https://www.linux.com/what-is-linux>>. Acesso em 11 nov. 2017.

MOUGAYAR, William. *The Business Blockchain: Promise, Practice, and Application of the Next Internet Technology*. John Wiley & Sons, 2016.

NAKAMOTO, S. *Bitcoin: A peer-to-peer electronic cash system*. 2008.

NODEJS. Disponível em: <<https://nodejs.org/>>. Acesso em: 15 nov. 2017.

NORTON, Christopher. *Blockchain: Everything You Need to Know About the Technology Behind Bitcoin*. Pronoun, 2017.

SWAN, Melanie. *Blockchain: Blueprint for a New Economy*. O'Reilly Media, Inc., 2015.

TAPSCOTT, Don; TAPSCOTT, Alex. *Blockchain Revolution: How the Technology Behind Bitcoin Is Changing Money, Business, and the World*. Penguin, 2016.