Implementação de uma Arquitetura com Blockchain Hyperledger Fabric para Provimento de Serviços de Blockchain através de uma API REST

Julio S. Quadros dos Santos, Adilso N. de Souza, Élder F. F. Bernardi

[1]Instituto Federal de Educação Ciência e Tecnologia - IFSul
Perimetral Leste, 150 – 99064-440 – Passo Fundo – RS – Brasil

julioquadros@gmail.com, {elder.bernardi,adilso.souza}@passofundo.ifsul.edu.br

Abstract. *This work presents the results of an introductory and exploratory research on the needs to implement Hyperledger Fabric Blockchain, and integrate with other general purposes systems. As a case study, a deployment was performed using feasible technologies to developers in general, which proved to be functional, using code software components and open standards. It contributes, besides the empirical experience, an architecture to integrate blockchain technology with legacy systems.*

Resumo. *Este trabalho apresenta o resultado de uma pesquisa introdutória e exploratória sobre as necessidades tecnológicas para a implantação do* Blockchain Hyperledger Fabric *para fins gerais de integração com outros sistemas. Como estudo de caso realizou-se uma implantação utilizando-se de tecnologias acessíveis a desenvolvedores em geral, a qual mostrou-se funcional, utilizando-se de componentes de software de código e padrões abertos. Traz como contribuição, além do relato da experiência empírica, uma arquitetura para integração da tecnologia de blockchain em sistemas legados.*

1. Introdução

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 alternativas 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 têm a capacidade de viabilizar projetos em benefício da sociedade em áreas como saúde, educação, transportes, alimentos e meio ambiente [Tapscott 2016]. Por este motivo, a Linux Foundation que é referência em implementação de padrões, propôs o projeto Hyperledger como padronização para a indústria [Bradburry 2017].

A visibilidade alcançada por sistemas de criptomoedas serviu como um propulsor do interesse pelas tecnologias adjacentes que as suportam. Juntando-se então à necessidade por sistemas de armazenamento e de transações com alta confiabilidade e o alto interesse pela tecnologia Blockchain, vê-se a necessidade de buscar e experimentar soluções que possam ser aplicadas nos mais diversos ramos, baseando-se em provedores de serviços e padrões já estabelecidos, que possibilitem a adesão destas tecnologias por

corporações ou desenvolvedores em geral. Diante do exposto, o presente estudo tem como objetivo identificar as necessidades tecnológicas para a implementação do Blockchain Hyperledger Fabric [Cachin 2016] e experimentar uma solução com tais tecnologias. Inicialmente é apresentada uma arquitetura adequada para a implementação da tecnologia, seguida pela definição de tecnologias Open Source adequadas a esta implementação, provendo os serviços necessários para o funcionamento do Blockchain Hyperledger Fabric. Também se apresenta um middlware de integração entre o Blockchain Hyperledger Fabric e sistemas externos legados por meio de uma Application Programming Interface (API). Como forma de validação da arquitetura proposta, implementa-se uma aplicação, sendo realizado testes para avaliação da estrutura implementada.

2. Referencial Teórico e Trabalhos Relacionados

O processo de implementação do Blockchain Hyperledger Fabric depende da utilização de tecnologias relacionadas, como sistemas operacionais, sistemas de virtualização e softwares para o correto funcionamento [Hyperledger 2017].

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 [IBM 2017], que inicialmente foi a idealizadora do Hyperledger Fabric, cedendo o projeto posteriormente à comunidade de software livre.

Também foram avaliadas outras formas de implementação do Blockchain Hyperledger Fabric utilizando conceitos similares, como a utilização de Kubernetes, e Chaincodes escritos em Java. Em [Zhang 2017], apresenta-se uma forma simplificada para a criação do ambiente utilizando ferramentas para contêineres. Em [Perry 2017], são descritos procedimentos para a criação dos smart contracts utilizando a linguagem de programação Java.

2.1. Blockchain

Blockchain [Melanie 2015] é o livro razão que serve como registro em blocos de todas as transações já executadas em um determinado domínio. Tornou-se popular graças à Bitcoin [Norton 2017]. No Blockchain os blocos gerados são inclusos de forma linear, e cronológica, mantendo uma cópia completa dos dados em cada nó na sua cadeia, que é atualizada automaticamente quando um novo nó é inserido no sistema [Melanie 2015]. Para manter a confiabilidade em relação aos registros armazenados é necessária a escolha de um serviço de consenso adequado a aplicação.

Segundo [Tapscott 2016], 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, dados para rastreabilidade de alimentos, entre outras coisas que possam ser expressas em código de computador.

Entre as principais características que destacam o Hyperledger Fabric das tecnologias similares, tais como o blockchain da Bitcon e da Ethereum [Wood 2014], destaca-se que este não é baseado em criptomoedas, permitindo a criação de redes privadas com

gestão de usuários através do uso de certificados. Também se destaca sua modularidade, que permite agregar diferentes serviços como diferentes algoritmos de consenso ou diferentes bases de dados.

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 à necessidade de intermediários. Através da definição de regras de negócios em Smart Contracts Chaincode [Hyperledger 2017], os parâmetros de negociação definidos podem ser acompanhados pelo serviço de consenso do Blockchain Hyperledger Fabric, eliminando a necessidade de intermediários de negociação, permitindo com que o Blockchain possa detectar quando as regras são satisfeitas, e atuar conforme os contratos estabelecidos e aceitos pelas partes negociadas, como numa transação comercial, por exemplo.

2.2. Hyperledger Fabric

O Hyperledger Fabric é um framework com o propósito de criação da infraestrutura necessária para a implementação de blockchains privados. É mantido pela Linux Fundation.

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, ficam registrados para sempre. Além disso, as características do serviço de consenso Kafka [Kafka 2017], baseado em regras de votação por permissões, permitem que o Hyperledger Fabric opere com economia de recursos. Estes fatores tornam o Hyperledger a principal escolha para a proposta deste trabalho.

2.3. Smart Contract Chaincode

Chaincode [Hyperledger 2017] 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.

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 intermediada pelo blockchain Hyperledger Fabric.

2.4. Demais tecnologias empregadas

Além das tecnologias mencionadas, fez-se uso sistema operacional GNU/Linux, na distribuição Cent OS [Sicam et al. 2010]; da plataforma de backend Node.js [NODEJS 2017]; da linguagem de programação Go [GOLANG 2017]; do padrão arquitetural para integração Web REST [Fielding 2000]; e da notação para representação de dados JSON [JSON 2017]. Como solução de virtualização, utilizou-se as soluções Docker [Merkel 2014].

Apresentadas as tecnologias empregadas, segue a arquitetura proposta para suportar a implementação do Hyperledger Fabric.

3. Arquitetura Proposta

A fim de se avaliar a arquitetura proposta, optou-se por uma abordagem empírica com o objetivo de experimentar as APIs estudadas. Neste sentido, implementou-se uma aplicação que consiste no cadastro e consulta de registros no formato JSON através da API desenvolvida. Estes registros negociados remotamente são, por meio do *middleware* e plataformas implantadas, registrados de forma transparente no blockchain do Hyperledger Fabric.

A Figura 1 exibe o fluxo de operações que envolveram tal experimento. Para detalhar o processo, os componentes necessários para o funcionamento do blockchain foram ilustrados de forma simplificada, onde os sistemas externos, representados por um sensor, uma estação de trabalho e um *smartphone*, podem executar aplicações compatíveis com o padrão REST. Dessa forma, o fluxo de dados gerado por componentes externos pode ser enviado à API (Sequência 1), que verifica se os parâmetros recebidos estão corretos e encaminha aos *peers* uma proposta de transação (Sequência 2), caso maioria dos *peers* concorde que a transação está correta, o sistema devolve uma autorização para submissão da transação ao SDK (Sequência 3). Neste ponto, o SDK envia uma requisição de submissão de transação ao serviço Orderer (Sequência 4), o Orderer irá verificar se os usuários envolvidos na operação têm as permissões definidas pelas políticas de controle de acesso, e se as regras estabelecidas pelo Chaincode foram satisfeitas (Sequência 5) para finalizar o processo, submetendo a transação aos peers para persistir os dados na base de dados do Hyperledger Fabric (Sequência 6).

Cada um dos *peers* depende basicamente de serviços específicos para um correto funcionamento. O serviço fabric-peer é responsável pela comunicação entre os *peers* utilizando o protocolo de comunicação Gossip, e pela disseminando os dados para persistência nos outros *peers*. O Chaincode define as regras que devem ser satisfeitas para o correto armazenamento dos dados, e o CouchDB armazena os dados conforme especificação das regras do Chaincode. Todos os *peers* armazenam exatamente a mesma base de dados blockchain.

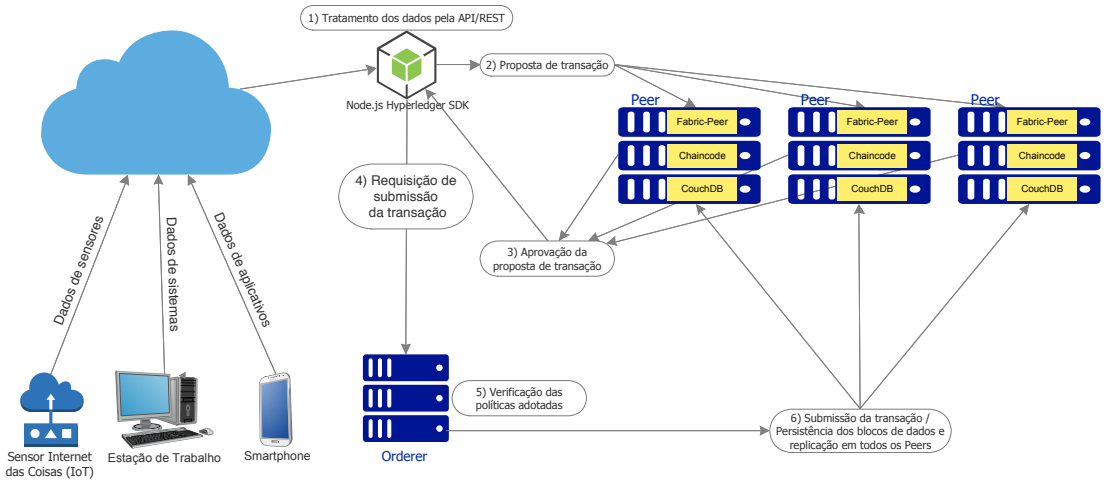

Figura 1. Visualização ampla do ambiente proposto na solução, com a demonstração dos fluxos necessários para a validação dos requisitos para a persistência de dados no blockchain.

Para definir os requisitos de implementação do Hyperledger, buscou-se referência na arquitetura deste. De acordo com Cachin[Cachin 2016], pode-se desenhar uma organização de recursos composta pelos seguintes componentes:

Plataforma de implantação

Camada composta pela infraestrutura de hardware, sistema operacional e virtualização. Neste ponto destaca-se a necessidade desta plataforma estar acessível sob a abstração de serviços. Portanto, a implantação desta camada em uma infraestrutura de Nuvem foi adotada. A infraestrutura em núvem, para o estudo foi composta de hardware com 1 processador virtual, 2Gb de memória RAM e 40Gb de armazenamento SSD. Em relação ao software adotado, o sistema foi constituído, em sua base, com o Sistema Operacional Linux CentOS 7, as ferramentas de desenvolvimento Go Lang, Node.JS, NPM e versionamento Git. Para a disponibilização dos serviços em contêineres foram utilizados os servidores e clientes Docker, e para a rodar a API REST foi utilizado o SDK Hyperledger Node.js, como demonstrado na Figura 2

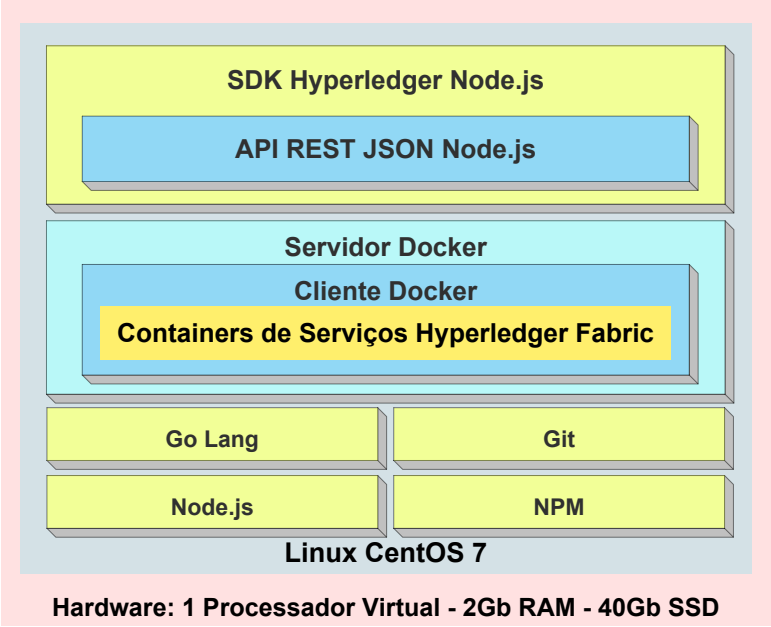

Figura 2. Estrutura utilizada para implementação dos serviços necessários para o funcionamento do Blockchain Hyperledger Fabric

Infraestrutura de Blockchain

Este é o principal requisito a ser implementado, pois trata-se do ponto principal da arquitetura proposta: a realização das tarefas de blockchain. Portanto, este componente trata especificamente da implantação do Hyperledger Fabric no nível de servidor.

Middleware de Integração

Necessária para prover os serviços de Blockchain para utilização em aplicações clientes. Tal middleware tem a funcionalidade de abstrair as particularidades de comunicação e representação de dados entre as partes heterogêneas, buscando interfaces e representações de dados comuns. Elencou-se as tecnologias REST e JSON, respectiva-

mente, para tal. O conjunto de tecnologias empregados nesta arquitetura, que implementam estes requisitos de maneira compatível com o propósito do blockchain foi definido através da criação de um Smart Contract Chaincode, como interface de comunicação com sistemas externos, o desenvolvimento de uma aplicação servidora no Node.JS.

API de Comunicação Externa

Esta API tem como propósito facilitar a integração de sistemas já estabelecidos no mercado e utilizados na indústria, desenvolvidos sem a previsão de integração com blockchain. Essa funcionalidade permite que sejam armazenadas ou replicadas informações sensíveis, como por exemplo, dados de registros de pessoas em cartórios de registro civil, registros financeiros, entre outros. Para isso, a arquitetura segue o padrão de aplicações Web corriqueiramente utilizado, provendo uma interface que possa receber dados e responder requisições no formato JSON em uma API com padrão REST de comunicação, utilizando o SDK Hyperledger Node.JS, como exemplificado na Figura 8.

A Figura 3 mostra as camadas da arquitetura proposta (à esquerda) com suas respectivas tecnologias e implementações (à direita). Também ilustra o deployment do lado servidor da arquitetura em serviços de nuvem e o acesso por clientes heterogêneos através da API fornecida.

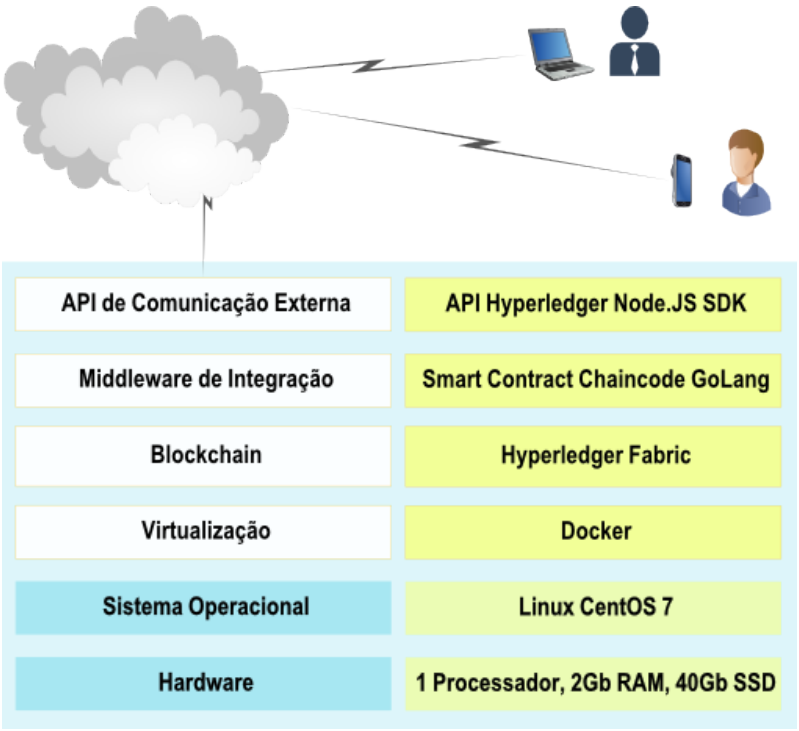

Figura 3. Modelo proposto para implementação do Blockchain Hyperledger Fabric

3.1. Funcionamento e Serviços Esperados

Dada organização proposta, espera-se que através dela seja possível realizar registros e consultas ao Blockchain por clientes heterogêneos. No tratamento destas requisições REST/JSON, converte-se os dados destas ao formato compatível com o Chaincode e leva-as a cabo ao Blockchain. Em relação à consulta de registros, o inverso é realizado: através

da API pode-se consultar os registros armazenados no formato Chaincode, sendo feita a conversão para JSON. Desta forma, cria-se um serviço de registro que pode ser integrado a qualquer aplicação que siga o padrão de Web Services e que, de forma transparente, realiza esses registros com os padrões do Blockchain como demonstrado pela Figura 4.

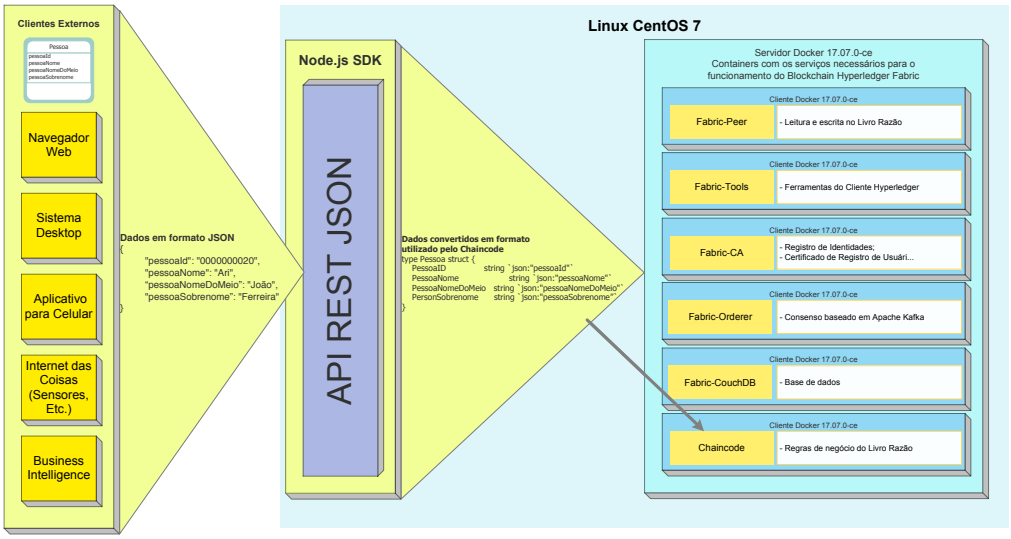

Figura 4. Funcionamento e serviços envolvidos no processo.

4. Implementação da Arquitetura

Para fins de ilustração e exemplo, o registro a ser gerenciado nos testes realizados corresponde à classe ”Pessoa”contendo os campos *pessoaId, pessoaNome, pessoaNomeDoMeio e pessoaSobrenome*. Tal simplificação é proposital no sentido de abstrair a complexidade de registros maiores e demonstrar um exemplo de uso geral [1].

Dito isto, segue a descrição da implementação dos componentes da arquitetura.

Estrutura básica

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 [NODEJS 2017], o servidor e cliente Docker e o SDK Hyperledger Node.js [sdk 2017], de forma a inicializar os serviços necessários para o funcionamento da estrutura necessária para o Blockchain Hyperledger Fabric. As versões dos softwares utilizados para o projeto foram o 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, demonstrados na Figura 5.

Containers Docker

o Hyperledger Fabric é composto por serviços que realizam as tarefas necessárias em containers distintos. Para o funcionamento, foram inicializados os containers que rodam os serviços necessários como o Fabric-Peer (responsável pela leitura e escrita na base

[1] O código-fonte da implementação realizada está disponível em: https://github.com/julioquadros/api-hyperledger-fabric

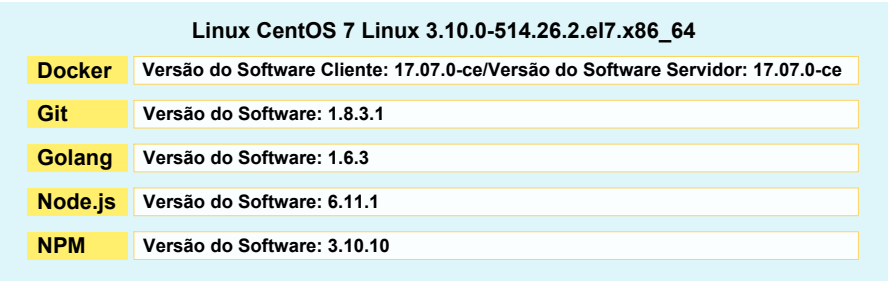

Figura 5. Softwares e respectivas versões utilizadas para a implementação

de dados), Fabric-Tools (ferramentas utilizadas para a configuração do cliente Hyperledger Fabric), Fabric-CA (necessário para o registro, renovações e revogações de identidades), Fabric-Orderer (responsável pelo consenso, utilizando Apache Kafka), Fabric-CouchDB (para a disponibilização da Base de dados, baseada em JSON), e o Chaincode (rodando as definições e regras de negócios necessárias para o consenso), apresentados na Figura 6. O sistema proposto utiliza containers Docker, previamente verificados para compatibilidade de versões. Para o estudo foram utilizados o Servidor versão 17.07.0-ce e Clientes Docker versão 17.07.0-ce. Como este componente da arquitetura está no modelo peer-to-peer, inicializou-se três peers para fins de consenso.

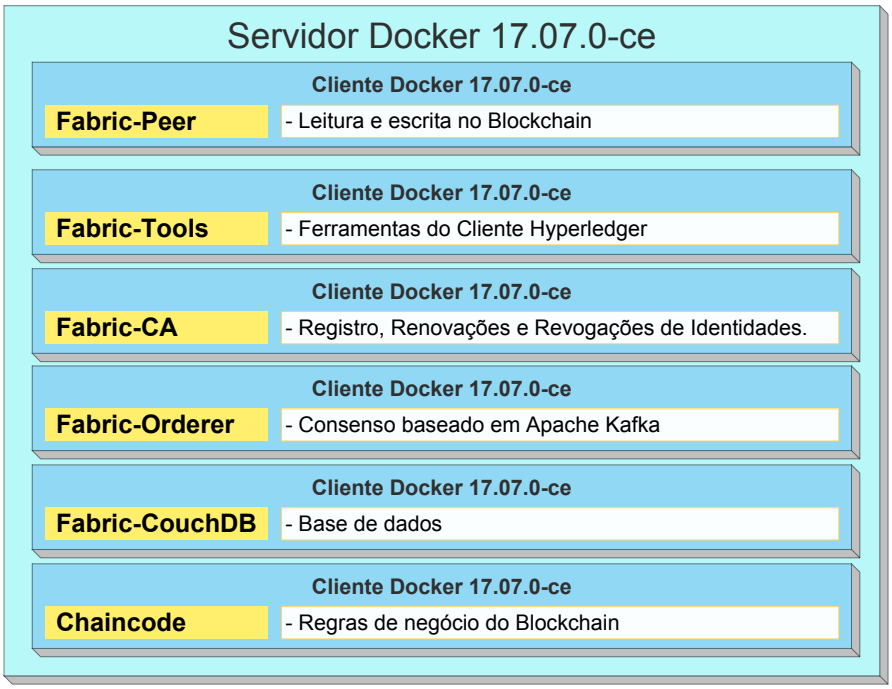

Figura 6. Containers Docker com os serviços necessários para o funcionamento do Blockchain Hyperledger Fabric.

Estrutura do Chaincode

Para a implementação dessa estrutura, seguiu-se o desenvolvimento do Chaincode, como forma de possibilitar o tratamento dos dados da comunicação entre o SDK e o Chaincode. Com o intuito de facilitar o entendimento dos processos necessários para

consulta e armazenamento de informações no blockchain, foram seguidas as orientações da documentação oficial do projeto Hyperledger e utilizou-se do pacote Shim[2] para comunicação com os serviços Hyperledger.

Para o desenvolvimento em linguagem de programação Go pode ser utilizado um editor de texto simples. Inicialmente, cria-se um pacote ”main”, em seguida são importadas as dependências para as funções no ”import”, após isso é definida uma estrutura de Smart Contract, e também uma estrutura para pessoa chamada ”Person”com campos JSON, formato utilizado pelo CouchDB. Após as definições iniciais são definidas funções.

O Chaincode necessita duas funções básicas, ”Init”e ”Invoke”. A função ”Init”é utilizada para a inicialização do Ledger (Livro Razão), e para isso 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. A próxima função necessária, o ”Invoke”é responsável por receber as solicitações externas, tratando os fluxos necessários e respondendo aos sistemas externos os dados solicitados. No Chaincode básico foi utilizada a função ”InitLedger”para inclusão dos dados iniciais na base de dados. Finalizando, deve ser utilizada a função ”main”para inicializar o Chaincode como demonstrado na Figura 7. As funções definidas no Chaincode são chamadas por parâmetro pelo Middleware, que faz a integração entre o Hyperledger Fabric e a API.

```
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{
        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))
}
```

Figura 7. Modelo básico para a criação de Chaincode

[2]github.com/hyperledger/fabric/core/chaincode/shim

Middleware e API de Comunicação Externa

Para o desenvolvimento em Node.js foi 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, a inicialização do serviço para responder as requisições externas, e a função de consulta de registros. Para a utilização da API, são necessários parâmetros utilizados na comunicação com o Hyperledger Fabric, no exemplo da Figura 8 é demonstrado como são definidos os parâmetros, como a identificação de usuário utilizado pela API (`user_id`), o nome do canal utilizado (`channel_id`), o Smart Contract Chaincode (`chaincode_id`). Também são definidas as URL's utilizadas para comunicação com os serviços necessários para o funcionamento da API, como o peer, orderer, event e network. A comunicação entre os hosts separados em containers é realizada utilizando Remote Procedure Call (gRPC).

```
'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: 'certidao_nascimento',
    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;

```

Figura 8. Implementação da interface de comunicação externa utilizando Node.js

Para responder as requisições externas, foi necessário manter uma aplicação respondendo como um serviço em uma porta, a API desenvolvida para os testes funciona na porta 3000, com o código demonstrado na Figura 9, utilizando as definições anteriores da variável `app_port`.

```
function startListener() {
    console.log("Starting WebApp on port " + app_port);
    app.listen(app_port);
    console.log("WebApp is now listening on port " + app_port + "\n");
}

startListener();
```

Figura 9. Inicialização da API para ouvir as solicitações externas na porta 3000.

Comunicação entre sistemas externos e o Blockchain Hyperledger Fabric

Para demonstrar a comunicação entre sistemas externos e o Hyperledger Fabric, foram realizados testes utilizando o software Insomnia com padrão REST, enviando os dados necessários para o cadastro de uma pessoa como apresentado na Figura 10. No exemplo, são enviados parâmetros referentes ao cadastro de cliente utilizando o método POST, informando o IP do servidor e a porta da API (3000), o parâmetro /api/cadastraPessoa referente a função a ser chamada, e as variáveis referentes a chave, id, primeiro nome, nome do meio, e sobrenome.

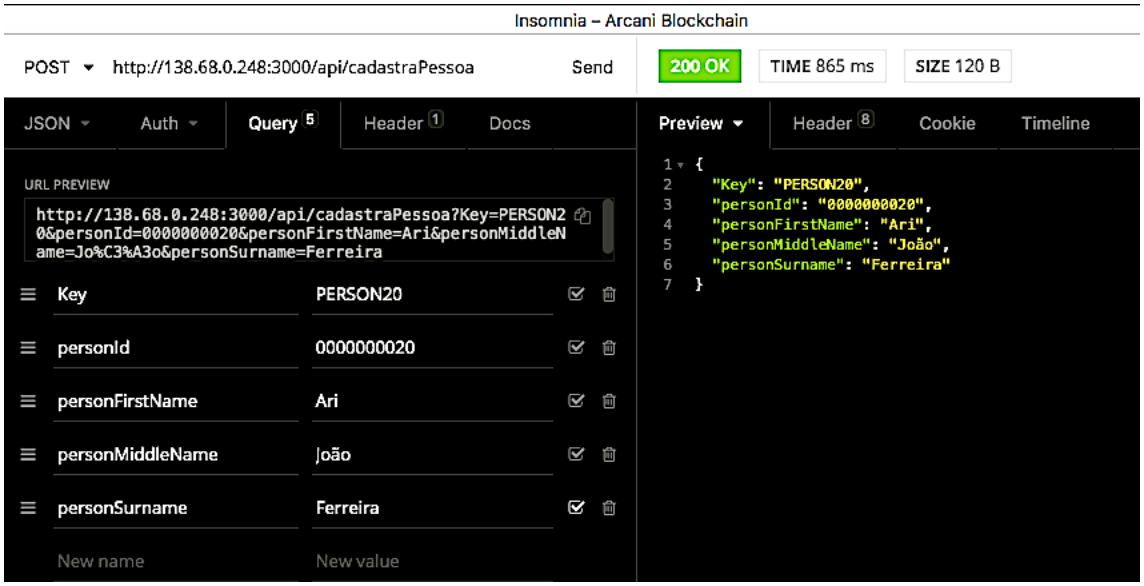

Figura 10. Cadastro de uma pessoa utilizando JSON através da API REST.

Para verificação do sucesso na comunicação, também foram monitorados os registros no Blockchain, as mensagens geradas pela API no terminal Linux demonstram o sucesso inserção do cadastro da pessoa (Figura 11). As mensagens demonstram que a proposta de inserção de cadastro foi satisfeita com sucesso pelo serviço de consenso utilizando os parâmetros do Chaincode, verificando se o usuário tem permissões para submissão da proposta, em seguida a mensagem informa que a transação foi submetida com sucesso, e os dados foram registrados no peer, na sequência os dados foram submetidos ao serviço de orderer para replicar os dados nos outros peers.

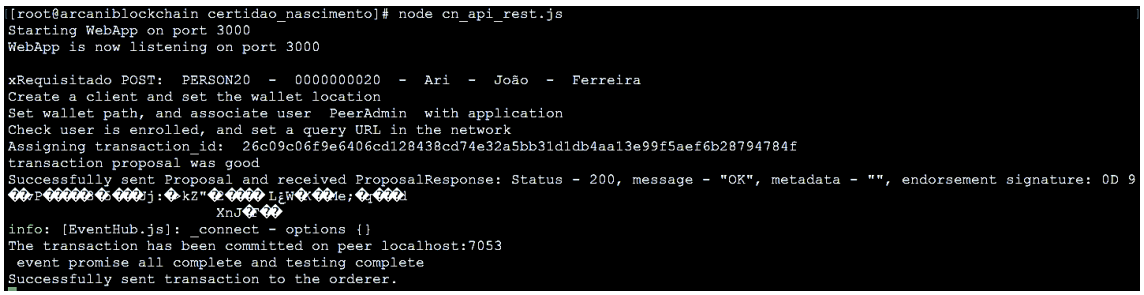

Figura 11. Mensagens geradas no console Linux pelo Node.js.

Após a constatação de sucesso no registro dos dados, uma consulta foi realizada

utilizando o software Insomnia passando o parâmetro do registro a ser consultado no padrão REST (Figura 12). O método utilizado para consulta foi o GET, informando o IP do servidor, utilizando a porta da API (3000). Para consulta de cadastro o único parâmetro necessário é a chave ”PERSON20”, seguindo o padrão de comunicação REST através do /api/consultaPessoa. A resposta da consulta corresponde ao registro enviado anteriormente.

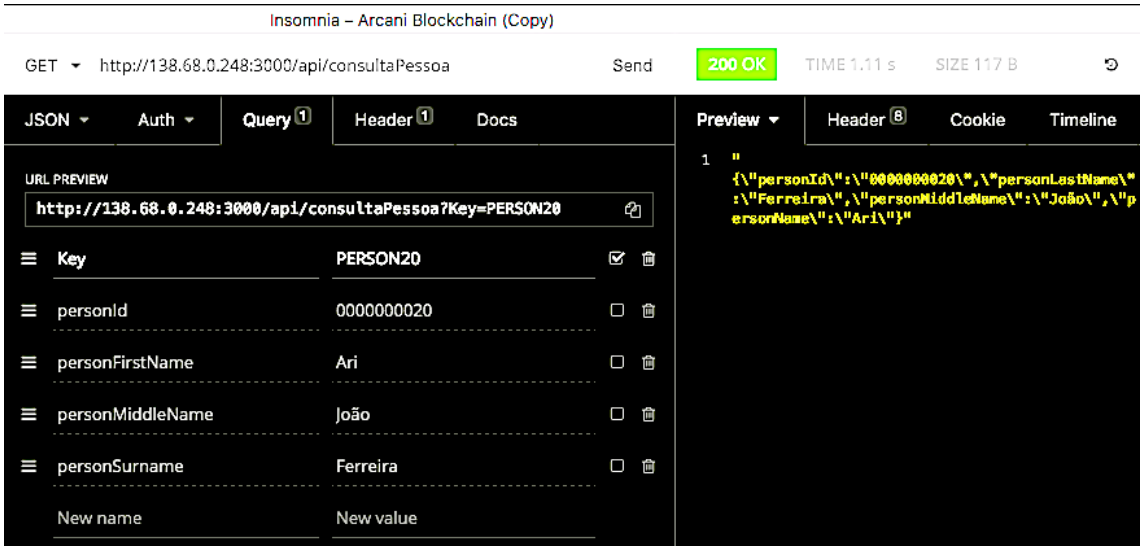

Figura 12. Consulta de dados de uma pessoa através da API REST.

5. Avaliação da Arquitetura

Através desta aplicação foi possível observar que a implementação da arquitetura proposta mostrou-se adequada, sendo todos os requisitos funcionais do sistema atendidos. Também, do ponto de vista não funcional, observou-se que a camada de transparência do formato de dados e acesso aos recursos do Hyperledger tornaram possível que a aplicação de teste pudesse ser organizada e implementada com um baixíssimo acoplamento à arquitetura proposta. Tal desacoplamento deu-se pelo uso das interfaces de comunicação padrões, especialmente em ambientes Web. Dito isto, pode-se inferir que o trabalho proposto atende como uma prova de conceito de integração de sistemas legados, ou seja, que foram planejados para armazenamento de registros sem Blockchain com o padrão REST, podem ser integrados a um serviço de armazenamento no modelo Blockchain.

Com o estudo para implementação, foi possível definir a estrutura de software e hardware necessária para o funcionamento do Blockchain Hyperledger Fabric em ambiente Linux. 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. Foram realizados testes utilizando o software Insomnia como forma de verificar a comunicação no padrão REST, enviando os dados necessários para o cadastro no formato JSON. Também efetuados testes de consulta do registro, passando o parâmetro do registro a ser consultado no padrão

REST. Os resultados são gerados pelo Chaincode, que retorna um resultado em formato JSON para o SDK, devolvendo os dados solicitados aos sistemas externos.

6. Considerações Finais

O trabalho desenvolvido apresentou uma arquitetura para integração da tecnologia Blockchain em sistemas legados. Pode-se experimentar uma implementação desta arquitetura utilizando-se de padrões e softwares de código aberto. A avaliação empírica demonstrou a viabilidade da arquitetura para atender a integração de sistemas legados ao Blockchain.

Como trabalhos futuros, pretende-se incrementar a aplicação para que de fato possam ser utilizadas os recursos disponibilizados pelo Blockchain, em relação a inteligência proporcionada pelos Smart Contracts Chaincode. Como por exemplo a verificação de regras de negócios, interigando dados gerados por sensores, ações de usuários registrados no sistema e respostas de sistemas externos como APIs de cotações de valores. Sendo essa implementação necessária para o desenvolvimento de um oráculo Blockchain.

Referências

(2017). Hyperledger Fabric SDK for node.js. `https://fabric-sdk-node.github.io/`. Acessado em: 12/2017.

Bradburry, D. (2017). Hyperledger goes to school. `http://www.nasdaq.com/article/hyperledger-goes-to-school-cm872546`. Acessado em: 12/2017.

Cachin, C. (2016). Architecture of the hyperledger blockchain fabric. In *Workshop on Distributed Cryptocurrencies and Consensus Ledgers*.

Fielding, R. T. (2000). Rest : Architectural styles and the design of network-based software architectures. *Doctoral dissertation, University of California*.

GOLANG (2017). `https://golang.org/`. Acessado em: 12/2017.

Hyperledger (2017). Chaincode for developers. `http://hyperledger-fabric.readthedocs.io/en/latest/chaincode4ade.html`. Acessado em: 12/2017.

IBM (2017). Ibm blockchain based on hyperledger fabric from the linux foundation. `https://www.ibm.com/blockchain/hyperledger.html`. Acessado em: 12/2017.

JSON (2017). The javascript object notation (json) data interchange format. `https://tools.ietf.org/html/rfc7159`. Acessado em: 12/2017.

Kafka (2017). Bringing up a kafka-based ordering service. `http://hyperledger-fabric.readthedocs.io/en/release-1.0/kafka.html`. Acessado em: 12/2017.

Melanie, S. (2015). Blockchain: Blueprint for a new economy. *Sebastopol: O'Reilly Media*.

Merkel, D. (2014). Docker: Lightweight linux containers for consistent development and deployment. *Linux J.*, 2014(239).

NODEJS (2017). Node.js. https://nodejs.org/. Acessado em: 12/2017.

Norton, C. (2017). *Blockchain: Everything You Need to Know About the Technology Behind Bitcoin*. Pronoun.

Perry, J. S. (2017). Blockchain chaincode for java developers. https://www.ibm.com/developerworks/library/j-chaincode-for-java-developers/j-chaincode-for-java-developers-pdf.pdf. Acessado em: 08/2018.

Sicam, C., Baclit, R., Membrey, P., and Newbigin, J. (2010). *Foundations of CentOS Linux: Enterprise Linux On the Cheap*. Springer.

Tapscott, Don; Tapscott, A. (2016). *Blockchain revolution: how the technology behind bitcoin is changing money, business, and the world*. Penguin.

Wood, G. (2014). Ethereum: A secure decentralised generalised transaction ledger. *Ethereum Project Yellow Paper*, 151:1–32.

Zhang, H. (2017). How to deploy hyperledger fabric on kubernetes. https://medium.com/@zhanghenry/how-to-deploy-hyperledger-fabric-on-kubernetes-2-751abf44c807. Acessado em: 08/2018.