Implementação de um Sistema de Blockchain para Garantir a Integridade de Informações de Contexto Distribuídas em Brokers Federados

Fabio Rogério Faria Barbosa[1]
Lucas Lima Giacomi[2]
Rafael Martinez Souza[3]
Ruann Oliveira Melgaço[4]
Fábio Henrique Cabrini[5]

Resumo:

Este trabalho tem o objetivo de implementar uma arquitetura de segurança baseada em *Blockchain* que possa garantir a integridade das informações no banco de dados do Helix Sandbox NG. Por isso, foi criada uma instalação de *Blockchain* para provar, através de uma federação de *Context Brokers*, como ela pode informar o responsável sobre tentativas de mudança de um dado na federação. Para isso, foi necessária uma análise do sistema responsável por reorientar as informações recebidas.

Palavras-chave: HelixSandboxNG. FIWARE. Blockchain. IoT, Context Brokers.

Abstract:

This work aims to implement a blockchain-based security architecture, which can guarantee the integrity of the information in the Helix Sandbox NG database. For this reason, a Blockchain installation was created to prove, through a federation of Context Brokers, how it could inform the responsible person about attempts to change data in the federation. For this, an analysis of the system responsible for redirecting the information received was necessary.

Keywords: HelixSandboxNG. FIWARE. Blockchain. IoT, Context Brokers.

Introdução

A Internet das Coisas (*Internet of Things* - IoT*)* têm crescido de maneira expressiva nos últimos anos (ABNIC, 2019). Sua principal característica é realizar a interconexão de dispositivos cotidianos à internet (CONNER, 2010), como sensores e atuadores utilizados para diferentes fins, por exemplo: monitoramento de *smart bikes*, patinetes, carros autônomos, ambientes inteligentes etc.

Estes dispositivos são extremamente vulneráveis a ataques (CIO, 2019) e o uso de *blockchain* pode ser uma forma promissora para defender a integridade das informações armazenada ou transmitidas por esses dispositivos. Uma *blockchain* atua como um livro-razão, permitindo que as

[1] Graduando em Tecnologia em Segurança da Informação. E-mail: fabio.barbosa16@fatec.sp.gov.br
[2] Tecnólogo em Segurança da Informação. E-mail: lucas.giacomi@fatec.sp.gov.br
[3] Tecnólogo em Segurança da Informação. E-mail: rafael.souza151@fatec.sp.gov.br
[4] Graduando em Tecnologia em Segurança da Informação. E-mail: ruann.melgaco@fatec.sp.gov.br
[5] Mestre em Engenharia Elétrica. E-mail: fabio.cabrini@fatec.sp.gov.br

informações sejam gravadas e transmitidas de modo descentralizado (COINBASE, 2017), validando as informações através de algoritmos de consenso.

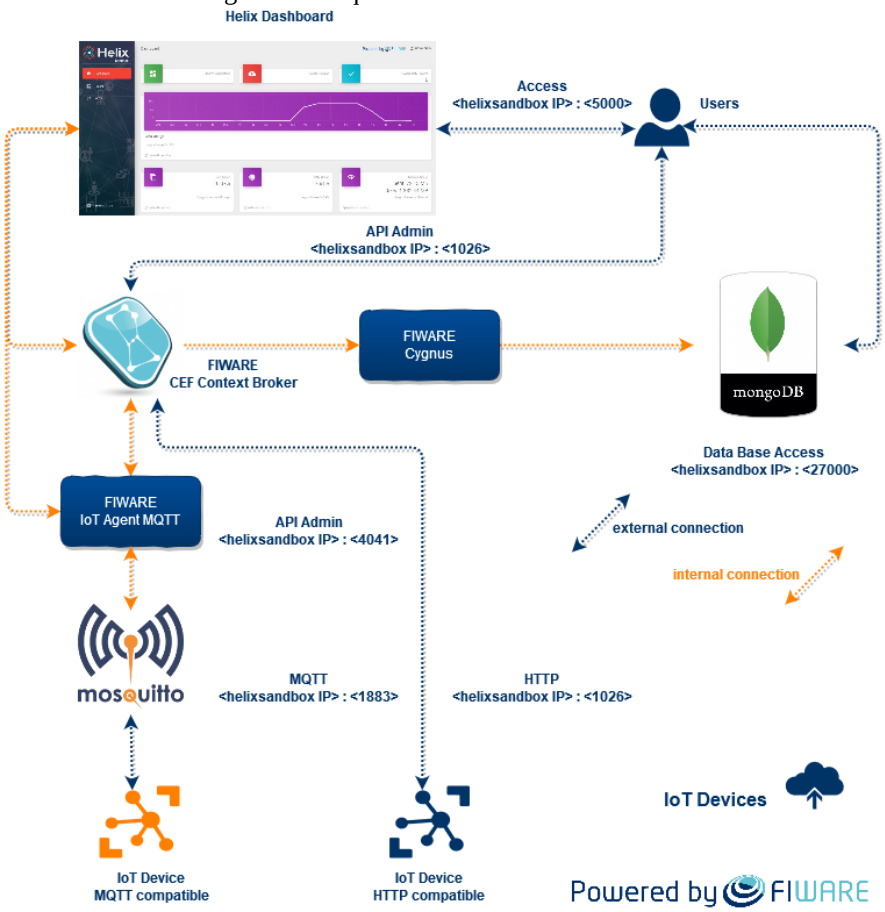
Estes algoritmos permitem que todas as máquinas da rede, chamadas de "*nodes*" entrem em um acordo sobre a veracidade de um dado adicionado à cadeia de informação, garantindo que a informação adicionada por um dos *nodes* foi a mesma repassada para os demais elementos da rede, ou seja, que a informação não tenha sido adulterada (LIN, 2017).

Helix SandBox NG

O Helix Sandbox NG (*Next Generation*) é uma plataforma de *back-end* desenvolvida para a prototipação rápida de aplicações para ambientes inteligentes e IoT na forma de POC (*Proof of Concept), MVP (Minimum Viable Product*) ou pesquisa acadêmica.

Ele facilita a instalação e a configuração dos principais *Generic Enablers* (GEs) previstos na arquitetura FIWARE através do Helix Compose e sua arquitetura pode ser vista na Figura 1.

Figura 1 – Arquitetura do Helix Sandbox NG



Fonte: CABRINI, Fábio (2019).

Ele apresenta uma curva de aprendizagem rápida para começar a explorar a ferramenta e não demanda um computador muito potente. O Helix foi pensado para ser uma solução bastante leve podendo ser executado em diversos ambientes localmente ou na nuvem.

No Helix encontramos, por exemplo, o *Context Broker*, um GE que gerencia o ciclo de vida das informações.

Ele trabalha com sete camadas em sua arquitetura: *Infrastructure*; *Operating System*; *Container Platform*; *Services*; *Administration*; *Context Providers* e *Context Consumers*:

- **Infrastructure**: O Helix foi pensado para ter uma arquitetura agnóstica, trabalhando tanto nos virtualizadores mais conhecidos, quanto em plataformas *Openstack*, *Cloud Service Providers* (CSP) e em *Bare Metal*, reduzindo tempo e custo com infraestrutura e pessoal (CABRINI, Fábio, 2019);
- **Operating System**: Ele foi projetado para ser uma solução leve, precisa de recursos mínimos para seu funcionamento correto e permite sua utilização em qualquer distribuição Linux;
- **Container Platform**: Utiliza o Docker Engine e o Docker Compose, que permitem o gerenciamento e a comunicação de contêineres com o *Docker* Hub, responsável pela atualização dos componentes do Helix e o fornecimento e APIs para o gerenciamento e monitoramento destes contêineres;
- **Services**: Apresenta os módulos que compõem o Helix, sendo eles: Helix Compose, CEF Context Broker, Cygnus IoT Agent MQTT, Eclipse-Mosquitto e MongoDB;
- **Administration**: Permite o acesso externo pela conexão SSH, pela interface gráfica do Helix Compose e pelas APIs fornecidas pelos GEs;
- **Context Providers** e **Context Consumers**: Demonstram a comunicação entre diversos dispositivos, como dispositivos IoT, agentes IoT, Dashboards, aplicativos mobile e APIs de terceiros.

Ele é uma plataforma oficial e homologada pela Fiware Foundation que certificou a plataforma como sendo *Powered by* FIWARE em 2018.

Federação

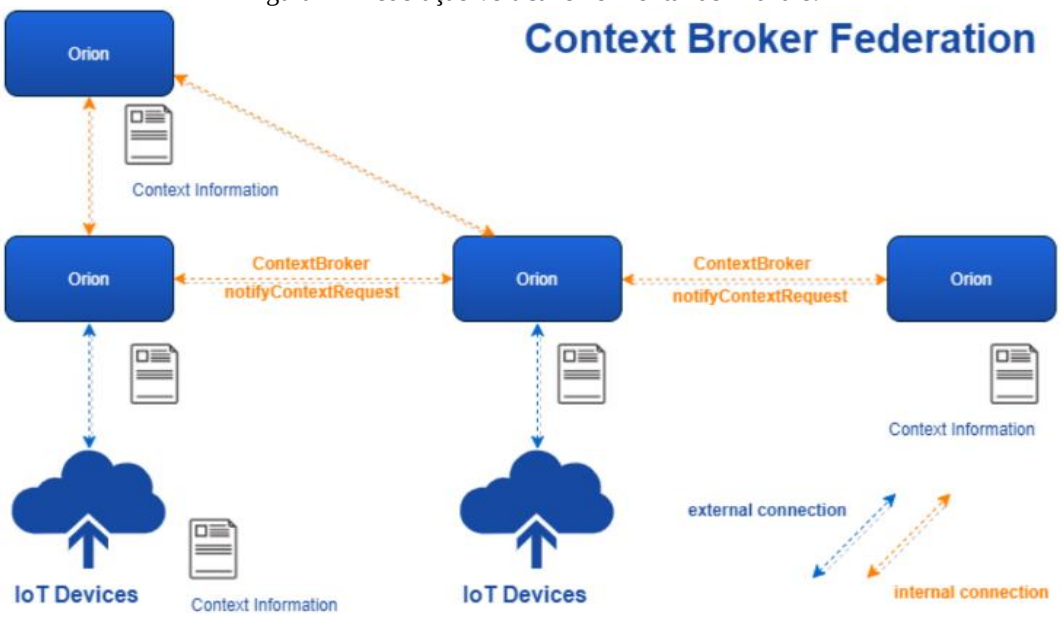
O termo pode ser usado ao descrever a interoperação de duas redes de telecomunicações distintas, formalmente desconectadas e que podem ter estruturas internas diferentes. (M. SERRANO, S. DAVY, M. JOHNSSON, W. DONNELLY, A. GALIS, 2011).

A federação no contexto do Helix é basicamente um encadeamento hierárquico de CEF Context Brokers e pode ser realizada através dos modos *push* e *pull* (FIWARE, 2019). Na federação do tipo *push*, as notificações de contexto enviadas por uma instância do CEF Context Broker são processadas e então enviadas para o próximo *Broker*.

Já na federação do tipo *pull*, as notificações de contexto são repassadas para a próxima instância sem que a primeira precise manter registros do dado. O processo de federação é feito através de assinaturas utilizando-se os parâmetros adequados.

Uma federação de *Context Brokers* pode ser criada "horizontalmente" no nível da nuvem ou do *fog*, ou ainda "verticalmente", interligando um ou mais *nodes* de *fog* uns com os outros e com a instância central na nuvem, cada qual executando sua instância do CEF Context Broker (CABRINI, Fabio H 2019).

Figura 2 – Federação vertical e horizontal de *Brokers*.



Fonte: CABRINI, Fábio H; V. FILHO, Filippo; T. KOFUJI, Sergio.

A Figura 2 apresenta o fluxo das informações dos dispositivos para a federação de *Brokers* que possibilitam o trânsito de informações a todos os elementos da arquitetura.

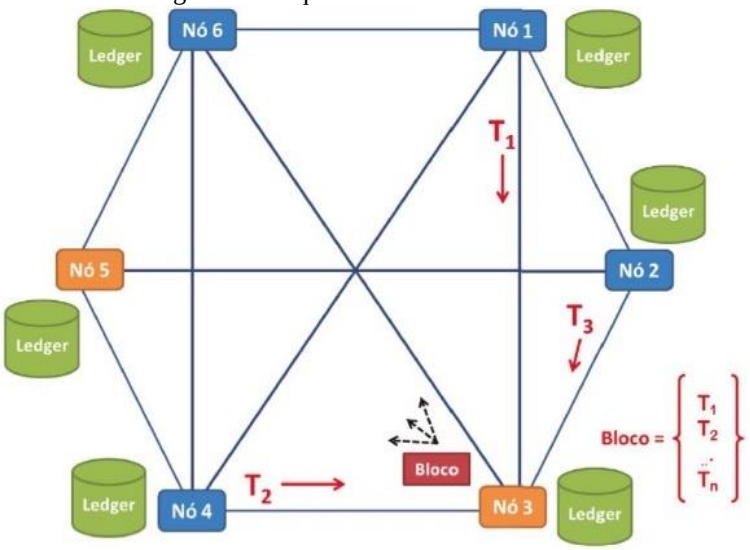
A tecnologia blockchain

Como visto anteriormente a *blockchain* é um livro-razão que garante registros confiáveis por serem imutáveis, ou seja, consiste em uma tecnologia que gera uma base de dados ordenados formada por blocos de transações criptografadas e fortemente interligadas que somente aceitam a inclusão de novos blocos, e nunca a remoção ou modificação de blocos existentes, respeitando assim a sua cronologia.

Como solução para segurança e intervenções governamentais, a *blockchain* trouxe um sistema descentralizado *Peer-to-Peer* (P2P) que preza pelo anonimato, garantindo assim a integridade total dos dados. A arquitetura de uma rede P2P é composta por *nodes*, onde todos exercem a função de cliente e/ou servidor, sem necessidade de um servidor central.

Para uma melhor compreensão da estrutura de uma *blockchain* é necessário contextualizar os *nodes* (nós) que são responsáveis por atuar como um ponto de comunicação que apresentam as seguintes funcionalidades: criação, armazenamento e validação de dados.

Figura 3 – Arquitetura de uma *blockchain*



Fonte: BRAGA, Alexandre 2017.

Há diversos tipos de *nodes*, neste trabalho será colocado em evidência o *Full Nodes* (Nós Completos), denominados também como *Full Validating Nodes* (Nós de Validação Total) que compreendem todas as funções supracitadas.

Ela também possui o *backup* de toda a cadeia de blocos, e tem como fim realizar a verificação da integridade de todos os blocos, sempre seguindo as normas dos algoritmos de consenso, conforme exposto na Figura 3.

A *blockchain* é mantida simultaneamente por todos os *nodes* da rede, não existindo local principal ou preferencial para armazenamento de uma base de dados original. Todo *node* tem a sua réplica da base de dados e todas são mantidas íntegras, consistentes e sincronizadas pelos protocolos de consenso.

Na Figura 3 também é demonstrada a base de dados que adota o termo ledger (coleção crescente de registros de transações).

Consenso distribuido

Consenso Distribuído é um termo utilizado em sistemas distribuídos sendo uma característica marcante nas técnicas de *blockchain* e utilizado em criptomoedas. O consenso ocorre entre os *nodes* da rede P2P por meio de métodos compostos por protocolos específicos e regras bem definidas.

Como exemplo temos o algoritmo de consenso bizantino (caracterizado pela necessidade de 3*n+1 *nodes* na rede P2P para tolerar n divergências no consenso), que foi utilizado como prova de trabalho no Bitcoin para mineração e como prova de participação na Ethereum.

Blocos e Transação

O bloco é formado por um cabeçalho contendo metadados, essa estrutura de dados que compreende as transações a serem incluídas na *blockchain* por intermédio dos *nodes* que validam os dados antes de incluírem, conforme mostra a Tabela 1.

Tabela 1 –Estrutura do Bloco.

Campo	Tamanho	Descrição
Índice	4 Bytes	Número do índice do bloco indica a posição em que ele foi criado.
Hash Anterior	32 Bytes	*Hash* do cabeçalho do bloco anterior (parente *block*) a este na *blockchain*.
Timestamp	4 Bytes	Hora exata da criação deste bloco em segundos no padrão UNIX.
Dados	Variável	Dado a ter sua integridade garantida.
Hash	32 Bytes	A função *hash* (resumo) deste bloco.
Dificuldade	4 Bytes	Dificuldade alvo do algoritmo de *proof-of-work* para este bloco.
Nonce	4 Bytes	Contador utilizado como *nonce* no algoritmo de *proof-of-work*.

Fonte: Autoria nossa.

O bloco será salvo se for validado pelos *nodes* seguindo o algoritmo de consenso da rede, caso contrário passam a não ter validade. A estrutura de criptomoedas, por exemplo, possui uma estrutura que se parece com um balancete contábil de débito e crédito e é composta dos seguintes elementos:

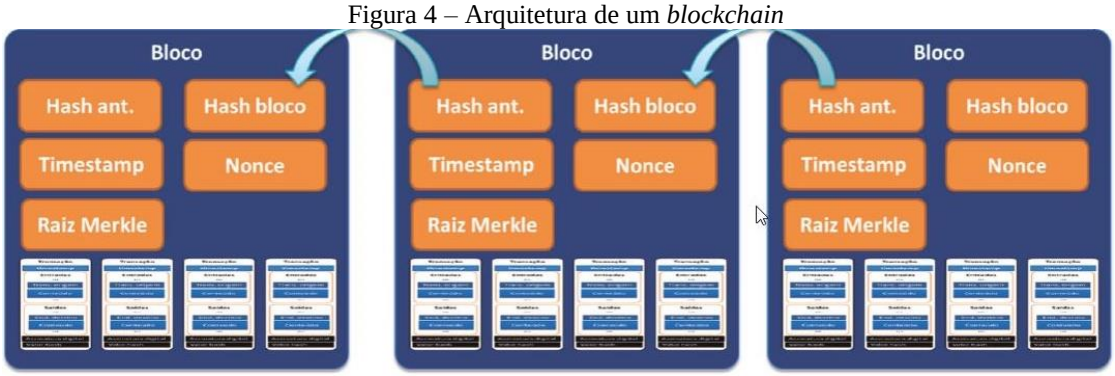
- ***Timestamp***; ***Hash***: O identificador da transação anterior de onde vemos valor de entrada (pode haver mais de um);
- **Valor de entrada**;
- **Valor de saída**;
- **Endereço de destino** (que vai receber o crédito);
- **Assinatura digital** feita com a chave privada do debitado.

A cadeia de blocos

Os *nodes* de uma rede *blockchain* validam consensualmente a veracidade em que os dados trafegados são registrados no *ledger*. Este consenso se dá com a maioria simples (50% + 1) da rede concordando com a integridade do bloco, para que o mesmo seja adicionado à cadeia.

As transações são incluídas em blocos que estão ordenados em uma cadeia, formando uma estrutura de dados conhecida em computação como “Lista Ligada”, conforme a Figura 4.

Figura 4 – Arquitetura de um *blockchain*

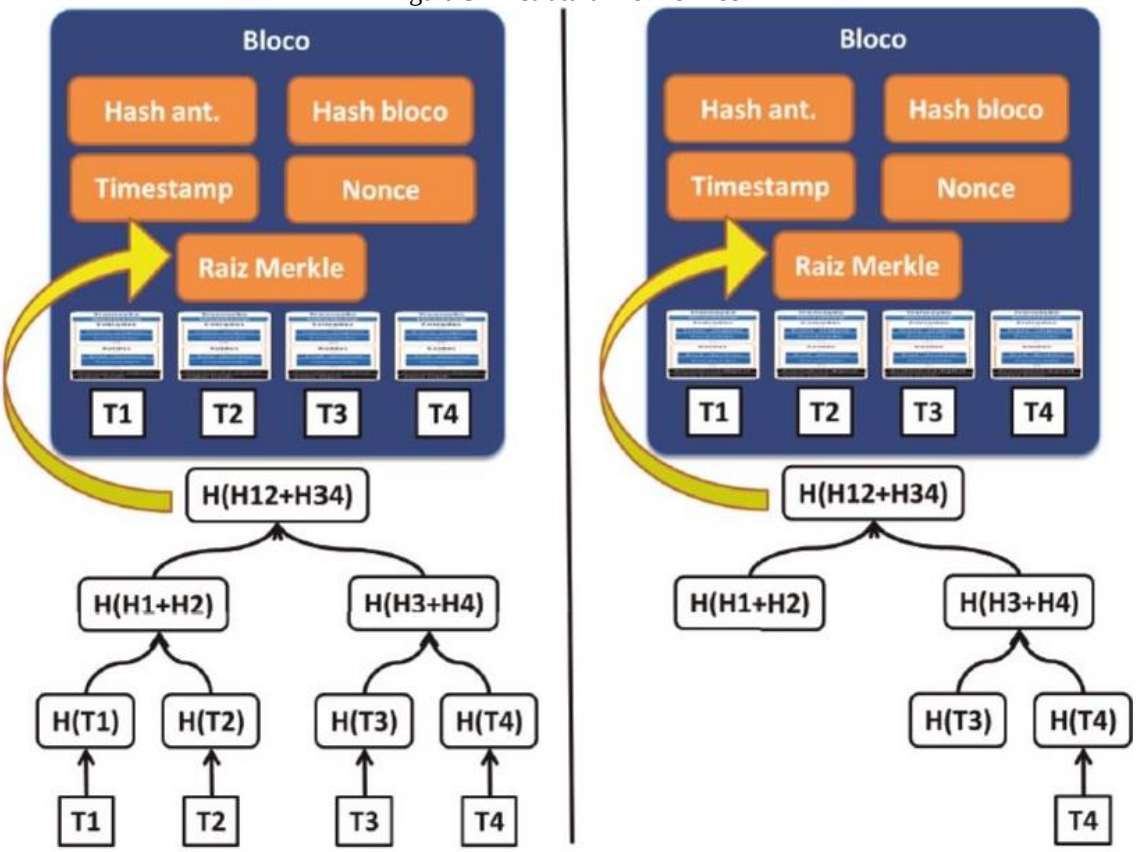


Fonte: BRAGA, Alexandre 2017.

O bloco mais recente é a cabeça (*head*) da cadeia. Cada bloco contém um conjunto de transações e um cabeçalho composto dos seguintes itens:

O *hash* do bloco anterior, um número pseudoaleatório único (*nonce*) e o *hash* da raiz da árvore de transações no bloco.

Figura 5 – Estrutura Merkle Tree



Fonte: BRAGA, Alexandre 2017.

Conforme a Figura 5 as transações dentro de um bloco estão ordenadas entre si de acordo com uma estrutura em árvore binária baseada em *hashes*, também conhecida como *Merkle Tree*. Ela ilustra a estrutura da árvore *Merkle* e a verificação de uma transação.

As folhas da árvore são os *hashes* das transações e os *hashes* dos pais são calculados com os *hashes* dos filhos. Os *hashes* dos ramos imediatos são calculados com os *hashes* das folhas, os *hashes* dos ramos intermediários são calculados com os *hashes* dos ramos imediatos, e assim sucessivamente até o cálculo do *hash* da raiz da árvore que é incluído no bloco.

Esta estrutura em árvore acelera a operação da verificação para saber se a transação pertence ao bloco, que pode ser feita em log(n) computações de *hash*, onde n é o tamanho da árvore. A verificação do *hash* de uma transação só usa o ramo da árvore (*Merkle branch*) em que a transação está localizada (BRAGA, Alexandre, 2017).

Parte experimental

A sessão seguinte apresenta os detalhes da implementação da Blockchain no Helix e também os testes realizados.

Instalação

Para a realização do experimento, foi utilizada a versão 2.4.0-Next do Helix Sandbox NG, seguindo os passos de instalação disponíveis em sua página no GitHub[6].

As máquinas utilizadas precisam ter os requisitos mínimos descritos no ***requirements*** (1 vCPU, 1GB RAM e 16GB HDD ou SSD), tal máquina será virtualizada no Amazon Web Services. Será utilizada a versão do Ubuntu Server 18.04.4 *Long Term Support* (LTS[7]), recomendada pelos desenvolvedores. O Postman é um software utilizado para a interação com o CEF Context Broker e para simular os dispositivos de IoT.

Para o correto funcionamento do Helix, as portas abaixo devem ser liberadas no firewall, conforme Tabela 2.

Tabela 2 – Portas de serviço do Helix

Port	**Transport**	**Protocol**
22	TCP	SSH
5000	TCP	Helix Web Interface
3030	TCP	Helix Orchestrator
22443	TCP	Helix Hardware Monitor
1026	TCP	CEF Context Broker
27000	TCP	MongoDB
5050	TCP	Cygnus
1883	TCP	Eclipse-Mosquitto
4041	TCP	IoT Agent MQTT

Fonte: CABRINI, Fábio (2019)

Depois de realizado o download disponível no GitHub execute o arquivo chamado *install.sh* localizado na pasta Sandbox-NG. Para acessar o *dashboard* insira o endereço http:// <IP_DO_HELIX>:5000 e então realize a criação do usuário administrador. Em seguida será necessário criar e executar o CEF Context Broker através do *dashboard*.

Todos os processos listados anteriormente estão disponíveis na página do Github do Helix acessível pela página oficial da plataforma (GETHELIX.ORG, 2019).

Criação de uma entidade

O primeiro passo é criar a entidade de contexto que é a representação lógica do dispositivo de IoT. O exemplo "*Room1*" utilizado é o mesmo encontrado no site da Fiware, desta forma será utilizado um sensor de temperatura para as explicações.

[6] Github Helix: https://github.com/Helix-Platform/Sandbox-NG

[7] Versão do sistema a qual possui suporte a longo prazo.

As entidades têm um *ID* (Identificador) que as identificam no formato JSON, além de atributos, tipos de dados e metadados que a descrevem, conforme a Figura 6.

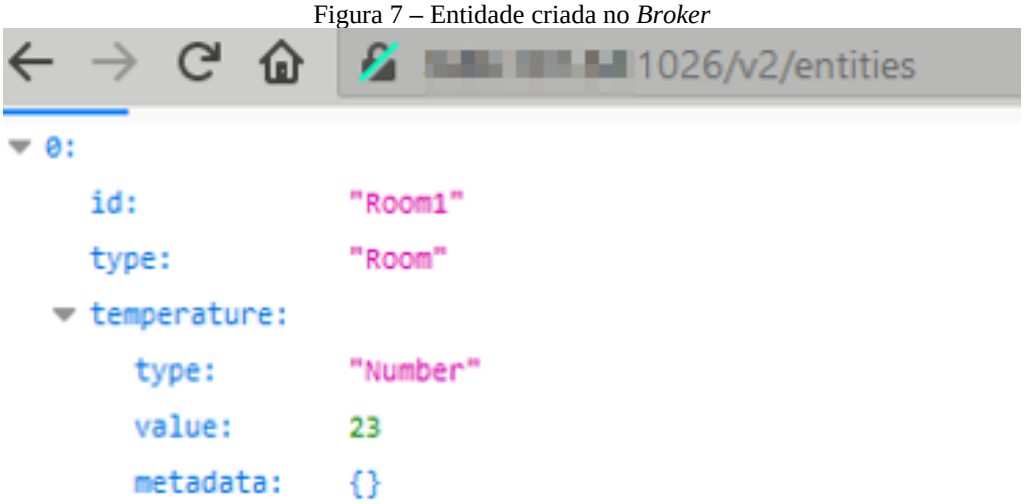
Figura 6 – Código para a criação da entidade

```
curl --location --request POST 'http://[redacted]
  [redacted]:1026/v2/entities' \
--header 'Content-Type: application/json' \
--data-raw '{
  "id": "Room1",
  "type": "Room",
  "level": {
  "value": "23",
  "type": "integer"
 }
}'
```

Fonte**:** Autoria nossa

A Figura 7 demonstra o resultado da criação da entidade dentro do *Broker*.

Figura 7 – Entidade criada no *Broker*



Fonte**:** Autoria nossa

Os metadados desta entidade estão de acordo com seu sensor, as informações podem variar em cada tipo de sensor.

Federação por meio da subscrição

A subscrição serve para enviar uma notificação a um *Broker* secundário, este processo é a federação propriamente dita. Pode-se observar que seu código é dividido em duas partes, sendo que a primeira parte informa qual entidade será federada definindo seu *ID* e Tipo.

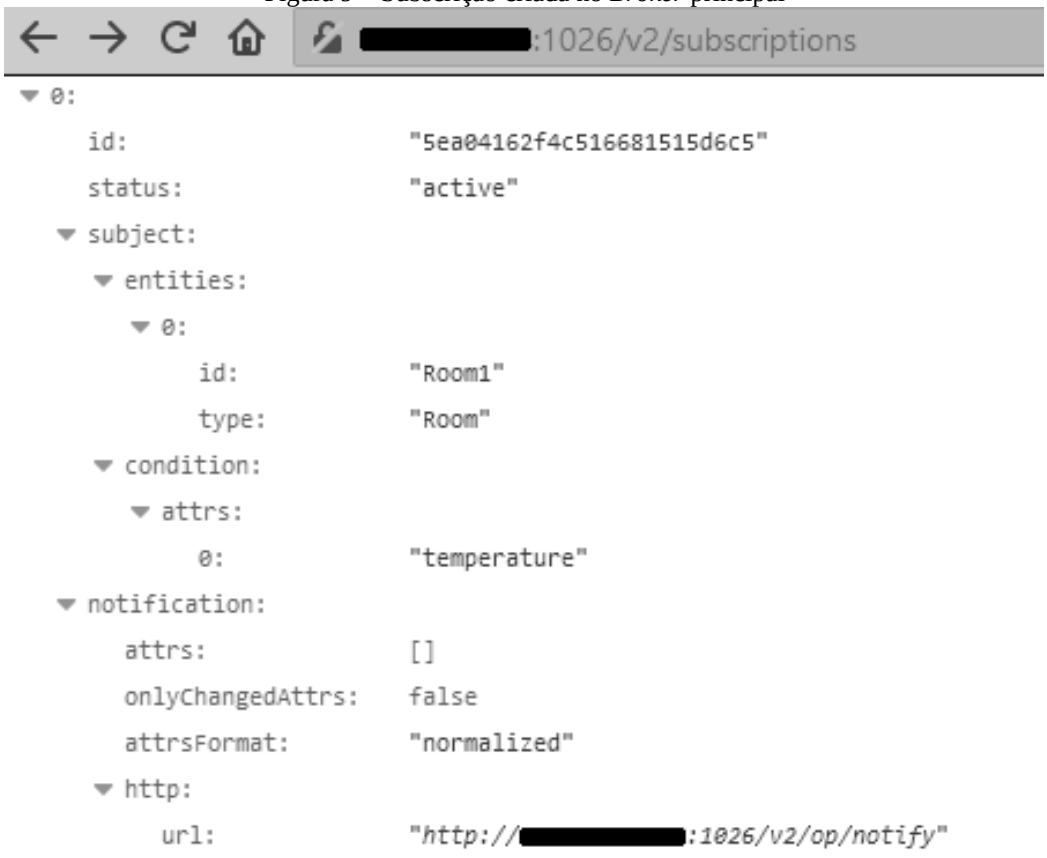
A segunda informa o *IP* do *Broker* que receberá a atualização dos dados, conforme a Figura 8 e 9.

Figura 8 – Código para a criação da subscrição.

```
curl --location --request POST 'http://[redacted]
  :1026/v2/subscriptions' \
--header 'Content-Type: application/json' \
--header 'Accept: application/json' \
--data-raw '{
  "description": "Notify Cygnus of all context changes",
  "subject": {
    "entities": [
      {
        "idPattern": ".*"
      }
    ]
  },
  "notification": {
    "http": {
      "url": "http://[redacted]1026/notify"
    },
    "attrsFormat": "legacy"
  },
  "throttling": 5
}'
```

Fonte: Autoria nossa

Figura 9 – Subscrição criada no *Broker* principal



Fonte: Autoria nossa

Como visto nas entidades a subscrição aqui também é composta por um *ID*, porém no lugar do Tipo há o s*tatus* da subscrição e os metadados.

Caso seja necessário federar o dispositivo para um terceiro *Broker* basta criar uma subscrição no segundo *Broker* apontando para o terceiro.

Atualização das entidades

Após a criação da entidade é possível atualizar os atributos através do método PUT, conforme a Figura 10:

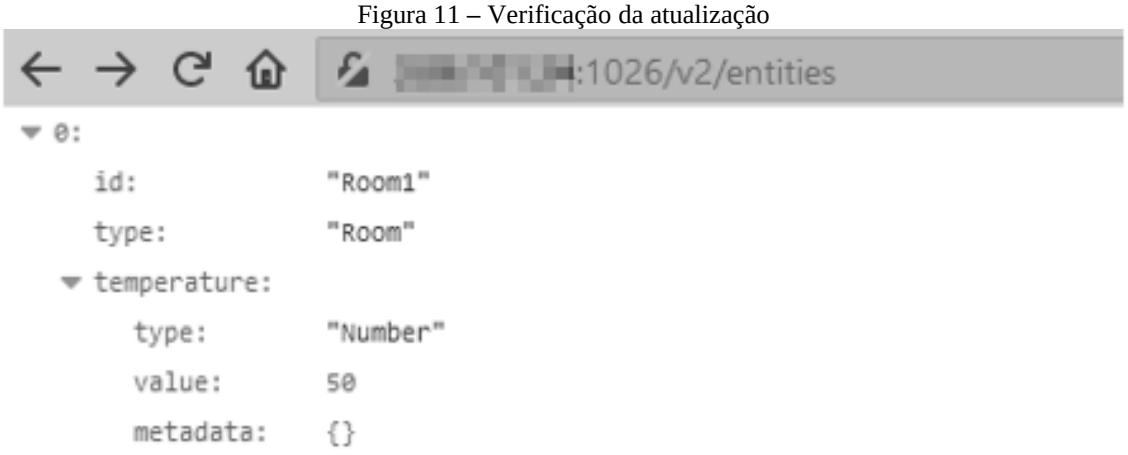
Figura 10 – Atualização do valor do sensor de temperatura da entidade

```
curl --location --request PUT 'http://[illegible]:1026/v2/entities/termometro/attrs/
    temperature/value' \
--header 'Content-Type: text/plain' \
--data-raw '1000'
```

Fonte: Autoria nossa

Note que é possível acessar o valor do sensor através de seu endereço. Os dados são estruturados em formato de árvore no *Broker*, conforme a Figura 11.

Figura 11 – Verificação da atualização



Fonte**:** Autoria nossa

Caso desejasse acessar apenas a entidade “Room1” bastaria digitar na barra de endereços <[IP_HELIX]:1026/v2/entities/Room1.

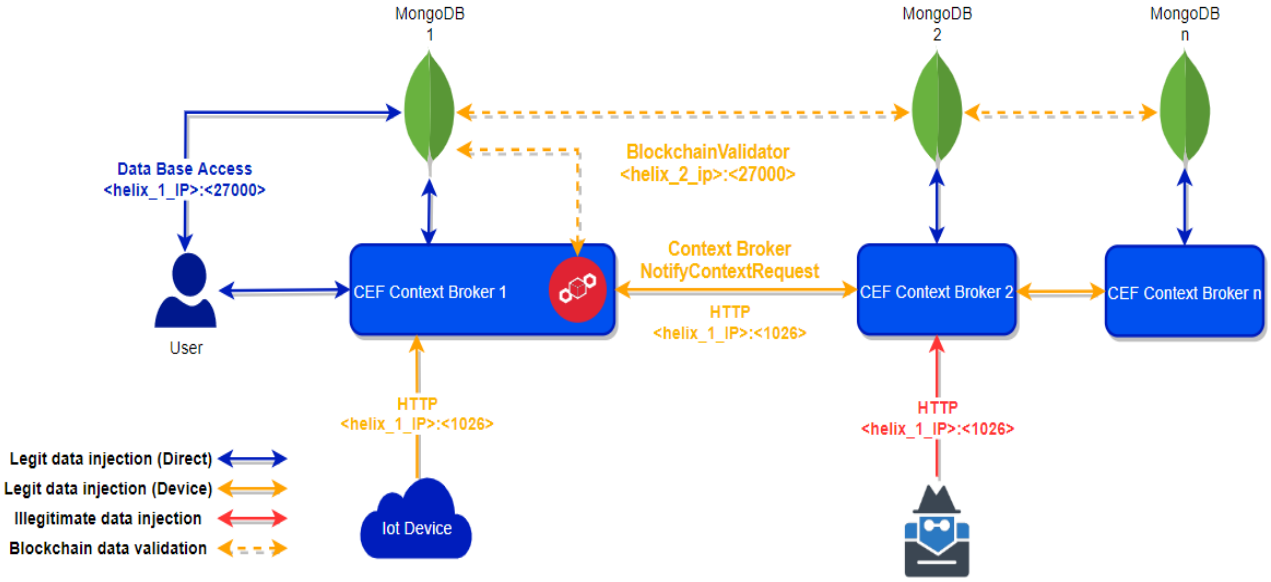
Aqui pode-se verificar uma vulnerabilidade do Helix um tanto problemática. É possível realizar a inserção de dados diretamente para o Broker sem qualquer tipo de validação, bastando apenas saber o IP do Helix e o dispositivo que se deseja alterar o valor.

Utilizando a Blockchain

A *Blockchain* criada faz a validação dos dados contidos nos *Brokers*. Após ser instalada no *Broker* principal, representada pelo ícone vermelho de *blockchain* no diagrama da Figura 13, ela verificará a primeira entidade cadastrada no *Broker* e a existência da federação.

Caso a federação exista a *Blockchain* tentará se conectar ao *Broker* secundário através da porta 27000 e então iniciará a comparação de valores entre o *Broker* principal e o secundário, conforme a Figura 13.

Figura 13 – Verificação da federação no *Broker* secundário



Fonte: Autoria nossa

Caso a federação não exista ou a porta se encontre bloqueada a *Blockchain* continuará funcionando, porém, a validação não será feita e consequentemente a mensagem de notificação no *prompt* da *Blockchain* não aparecerá.

Note que o Helix possui apenas um único MongoDB, por questões de otimização. Nele são guardados os registros de entidades, subscrições, registros e ainda, caso o armazenamento temporal dos dados seja ativado o módulo do Cygnus entra em operação.

Caso o armazenamento temporal não seja ativado o CEF Context Broker conversará diretamente com o MongoDB, conforme diagrama apresentado.

Para realizar o download da *Blockchain* é necessário acessar o endereço <*https://github.com/Martinez1991/helixBlockchain.git*>. Acesse a pasta da *Blockchain* e execute o código "*./install.sh*".

Após realizado o download execute o script "Principal.py". Conforme a Figura 14, temos a *Blockchain* rodando já com a atualização legítima de um dado.

Resultados obtidos

Conforme observado na Figura 14 existe a visão das primeiras informações obtidas pela *Blockchain*.

O índice 2 representa a primeira atualização dos dados, tendo em vista que o índice 0 é o bloco gênesis e o índice 1 representa a *Blockchain* obtendo os dados iniciais do sensor.

Figura 14**:** Bloco gênesis

```
Índice: 0
Hash anterior:
Timestamp: 1589549954282
Dados: Genesis block
Hash: e3438d3c39801e2bf64073b988d12dd21efe61630b2242d2070242aba8c8eecf
Dificuldade: 2
Nonce: 0

Índice: 1
Hash anterior: e3438d3c39801e2bf64073b988d12dd21efe61630b2242d2070242aba8c8eecf
Timestamp: 1589549956138
Dados: {'_id': {'id': 'Room1', 'type': 'Room', 'servicePath': '/'}, 'attrs': {'temperature': {'value': 15.0}}}
Hash: 0003e57c0c1215330994c33d453f563d31e3814c2bd002039030a7067d08abef
Dificuldade: 2
Nonce: 199

Índice: 2
Hash anterior: 0003e57c0c1215330994c33d453f563d31e3814c2bd002039030a7067d08abef
Timestamp: 1589549957005
Dados: {'_id': {'id': 'Room1', 'type': 'Room', 'servicePath': '/'}, 'attrNames': ['temperature'], 'attrs': {'temperature
': {'value': 15.0, 'type': 'Number', 'mdNames': [], 'creDate': 1589539644, 'modDate': 1589539948}}, 'creDate': 158953964
4, 'modDate': 1589539948, 'lastCorrelator': '2b62ae4c-969a-11ea-900d-0242ac120004'}
Hash: 00a4ba579a01268b127d448e31e98023a006fa855d411689febd712c6d6f4a6d
Dificuldade: 2
Nonce: 138
```

Fonte**:** Autoria nossa.

Enquanto a Figura 14 demonstra a atualização legítima do dado, a Figura 15 demonstra a tentativa de atualização indevida. Note que a informação não foi escrita pois foi impedida pela *Blockchain*.

Figura 15**:** *Blockchain* após alteração do dado

```
Índice: 0
Hash anterior:
Timestamp: 1589549954282
Dados: Genesis block
Hash: e3438d3c39801e2bf64073b988d12dd21efe61630b2242d2070242aba8c8eecf
Dificuldade: 2
Nonce: 0

Índice: 1
Hash anterior: e3438d3c39801e2bf64073b988d12dd21efe61630b2242d2070242aba8c8eecf
Timestamp: 1589549956138
Dados: {'_id': {'id': 'Room1', 'type': 'Room', 'servicePath': '/'}, 'attrs': {'temperature': {'value': 15.0}}}
Hash: 0003e57c0c1215330994c33d453f563d31e3814c2bd002039030a7067d08abef
Dificuldade: 2
Nonce: 199

Índice: 2
Hash anterior: 0003e57c0c1215330994c33d453f563d31e3814c2bd002039030a7067d08abef
Timestamp: 1589549957005
Dados: {'_id': {'id': 'Room1', 'type': 'Room', 'servicePath': '/'}, 'attrNames': ['temperature'], 'attrs': {'temperature
': {'value': 15.0, 'type': 'Number', 'mdNames': [], 'creDate': 1589539644, 'modDate': 1589539948}}, 'creDate': 158953964
4, 'modDate': 1589539948, 'lastCorrelator': '2b62ae4c-969a-11ea-900d-0242ac120004'}
Hash: 00a4ba579a01268b127d448e31e98023a006fa855d411689febd712c6d6f4a6d
Dificuldade: 2
Nonce: 138

['Room1']
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    Device Room1 Foi adulterado
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
[]
```

Fonte**:** Autoria nossa.

Note que ela descreve também o nome do dispositivo que foi adulterado, sendo um meio ainda rústico para notificar o usuário acerca da tentativa de adulteração.

Considerações finais

A *blockchain* desenvolvida neste trabalho possibilitou identificar alterações nos dados armazenados em uma federação de CEF Context Brokers. O custo computacional para sua execução é bastante baixo, uma vez que ela não executa funções complexas, ao exemplo de uma *blockchain* tradicional, exigindo assim recursos mínimos de *hardware*.

Sua execução é fortemente recomendada em ambientes acadêmicos para efeito de estudo e para o desenvolvimento de POCs e MVPs, incorporando recursos de segurança em aplicações de IoT distribuídas baseadas na plataforma Helix Sandbox NG.

Em atividades futuras será possível acrescentar a função para impedir a alteração indevida dos dados em todos os *brokers* da federação, sem a necessidade de instalar a *Blockchain* em cada um deles. Outro ponto importante, será a adição de algum tipo de autenticação de modo a identificar se a inserção dos dados está sendo realizada por um dispositivo de IoT ou manualmente por uma pessoa, adicionando uma camada a mais de segurança.

Referências

BANAFA, Ahmed; **IoT and *Blockchain* Convergence: Benefits and Challenges**. IEEE, 2017. Disponível em: <https://iot.ieee.org/newsletter/january-2017/iot-and-*blockchain*-convergence-benefits-and-challenges.html>.

BRAGA, Alexandre. 2017 **TECNOLOGIA *BLOCKCHAIN*: Fundamentos, Tecnologias de Segurança e Desenvolvimento de Software.** Disponível em: <https://www.cpqd.com.br/wp-cntent/uploads/2017/09/whitepaper_*Blockchain*_fundamentos_tecnologias_de_seguranca_e_desenvolvimento_de_softwar_FINAL.pdf>.

CABRINI, Fábio H. et al. **Helix: An Open Platform to Fast Prototype *Smart* Environments Applications**. In: **2019 IEEE 1st Sustainable Cities Latin America Conference (SCLA)**. IEEE, 2019. p. 1-6.

CABRINI, Fábio H; V. FILHO, Filippo; T. KOFUJI, Sergio. **Arquiteturas de Fog Computing para Internet das coisas nas plataformas Fiware e Helix**. São Paulo, Brasil 2019.

CABRINI, Fabio, 2019. **Helix.** Disponível em: <https://github.com/fabiocabrini/helix-Sandbox/blob/master/README.md>.

CONNER, Margery. Sensors Empower the Internet of Things. EDN. Disponível em <https://www.edn.com/sensors-empower-the-internet-of-things/>. Acesso em: 18 de agosto de 2019.

FREDERIC, Paul. **10 principais vulnerabilidades da Internet das Coisas**. CIO, 2019. Disponível em: <https://cio.com.br/10-principais-vulnerabilidades-da-internet-das-coisas/>

FIWARE, ***Context Broker* Federation**, 2019. Disponível em: <https://fiware-orion.readthedocs.io/en/master/user/federation/index.html >. Acesso em: 13 de Março de 2020.

FIWARE DATA MODELS, Disponível em: <https://fiware-datamodels.readthedocs.io/en/latest/index.html>, Acesso em: 27 de Maio de 2020.

HELIX, GetHelix.Org, Disponível em: <https://github.com/Helix-Platform/Sandbox-NG>. Acesso em: 7 de Maio de 2020.

IBM, 2011. **Introduction to InfoSphere Federation Server,** Disponível em: <https://www.ibm.com/support/knowledgecenter/en/SSZJPZ_8.7.0/com.ibm.swg.im.iis.productization.iisinfsv.overview.doc/topics/cisofedintro.html>. Acesso em: 22 de Outubro de 2019.

M. Serrano, S. Davy, M. Johnsson, W. Donnelly, A. Galis - "Review and Designs of Federated Management in Future Internet Architectures" parte do livro "The Future Internet - Future Internet Assembly 2011: Achievements and Technological Promises" Vol. 6, pg 465, 4 Maio 2011.

ONGARO, Diego; OUSTERHOUT, John. **In Search of an Understandable Consensus Algorithm**. USENIX, Stanford University, 2014.