UNIVERSIDADE FEDERAL RURAL DE PERNAMBUCO
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA APLICADA

Edgar Natanael de Vasconcelos Gregório

Transparência de Conectividade de Serviços Blockchain: Uma Proposta de Arquitetura

Dissertação de Mestrado

Recife, 2021

Edgar Natanael de Vasconcelos Gregório

Transparência de Conectividade de Serviços Blockchain: Uma Proposta de Arquitetura

Orientador: Prof. Dr. Fernando Antônio Aires Lins

Coorientador: Prof. Dr. Obionor de Oliveira Nóbrega

Dissertação apresentada ao Programa de Pós-Graduação em Informática Aplicada da Universidade Federal Rural de Pernambuco como requisito parcial para obtenção do título de Mestre em Informática Aplicada.

BANCA EXAMINADORA

_____

Prof Dr. Fernando Antônio Aires Lins
(UFRPE - Orientador)

_____

Prof Dr. Obionor de Oliveira Nóbrega
(UFRPE - Coorientador)

_____

Prof Dr. Marcos Antônio Domingues
(UFRPE - Examinador)

_____

Prof Dr. Jorge da Silva Correia Neto
(UFRPE - Examinador)

Recife, 2021

Dados Internacionais de Catalogação na Publicação
Universidade Federal Rural de Pernambuco
Sistema Integrado de Bibliotecas
Gerada automaticamente, mediante os dados fornecidos pelo(a) autor(a)

G821t Gregório, Edgar Natanael de Vasconcelos
Transparência de Conectividade de Serviços Blockchain: Uma Proposta de Arquitetura / Edgar Natanael de Vasconcelos Gregório. - 2021.
90 f. : il.

Orientador: Fernando Antonio Aires Lins.
Coorientador: Obionor de Oliveira Nobrega.
Inclui referências.

Dissertação (Mestrado) - Universidade Federal Rural de Pernambuco, Programa de Pós-Graduação em Informática Aplicada, Recife, 2021.

1. Internet das Coisas. 2. Blockchain. 3. Sistemas Distribuídos. 4. Transparência. I. Lins, Fernando Antonio Aires, orient. II. Nobrega, Obionor de Oliveira, coorient. III. Título

CDD 004

Dedico esta dissertação para a minha família, especialmente a minha esposa, pela paciência e incentivo na conclusão deste trabalho.

Obrigado!

AGRADECIMENTOS

Agradeço à Universidade Federal Rural de Pernambuco pela oportunidade de desenvolver este trabalho de pesquisa, incluindo todos os professores que fizeram parte da construção dos conhecimentos adquiridos em sala de aula. Toda gratidão aos Professores Fernando Antônio Aires Lins e Obionor de Oliveira Nóbrega, por serem parte importante da minha história na vida acadêmica.

Agradeço o apoio fundamental da minha família e das pessoas que acreditaram e não mediram esforços para me ajudar a chegar até aqui. Muitíssimo obrigado! Gostaria de agradecer a minha esposa Andréia Leite de Almeida por estar do meu lado em todos os momentos, incentivando e me substituindo em minhas obrigações como ESPOSO, para que pudesse me dedicar exclusivamente a esta pesquisa, por fim, agradeço aos professores da banca deste trabalho.

RESUMO

A Internet das Coisas criou um mundo conectado através dos mais diversos tipos de sensores, possibilitando que objetos de nossas vidas cotidianas estejam interligados à rede. Esta conectividade proporcionou uma forte interação com os dados gerados em tempo real pelos dispositivos. No entanto, à medida que a quantidade de dados gerados por esses dispositivos aumentam, o risco de roubo e adulteração deles se torna ainda mais relevante. Para melhorar o nível de segurança dentro da Internet das Coisas, o *Blockchain* se apresenta como uma potencial solução para a integridade dos dados. Contudo, a realização da conexão entre dispositivos IoT e *Blockchain* é uma tarefa de razoável complexidade. Neste contexto, este trabalho propõe uma arquitetura para transparência de conectividade de serviços *Blockchain* em Sistemas IoT. Esta arquitetura busca proporcionar ao usuário final a percepção da utilização do sistema proposto como um sistema único e totalmente centralizado, ocultando as complexidades desta integração. Para este fim, foi realizado um estudo aprofundado sobre as características das redes *Blockchain*, apresentando benefícios na implementação em sistemas de Internet das Coisas e um estudo dos tipos de plataformas e dos protocolos de comunicação mais utilizados. Para avaliar a arquitetura proposta, foi realizado estudo de caso que buscou analisar a percepção do usuário final na utilização do sistema proposto e consequentemente o nível de transparência aplicado.

Palavras-chave: Internet das coisas, Blockchain, Sistemas Distribuídos, Transparência.

ABSTRACT

The Internet of Things created a world connected through the most diverse types of sensors, making it possible for objects in our daily lives to be interconnected to the network. This connectivity provided a strong interaction with the data generated in real time by the devices. However, as the amount of data generated by these devices increases, the risk of theft and tampering with them becomes even more relevant. To improve the level of security within the Internet of Things, Blockchain presents itself as a potential solution for data integrity. However, making the connection between IoT and Blockchain devices is a task of reasonable complexity. In this context, this work proposes an architecture for transparency of connectivity of Blockchain services in IoT Systems. This architecture seeks to provide the end user with the perception of the use of the proposed system as a single and fully centralized system, hiding the complexities of this integration. To this end, an in-depth study on the characteristics of Blockchain networks was carried out, presenting benefits in the implementation in IoT systems and a study of the types of platforms and the most used communication protocols. To assess the proposed architecture, a case study was carried out that sought to analyze the perception of the end user in the use of the proposed system and consequently the level of transparency applied.

Keywords: Internet of things, Blockchain, Distributed Systems, Transparency.

Lista de Figuras

Lista de Tabelas

Lista de Quadros

Lista de Abreviaturas e Siglas

IOT	Internet das Coisas
BC	Blockchain
OWASP	Open Web Application Security Project
IDC	International Data Corporation
QoS	Qualidade de Serviços
QoE	Qualidade de Experiência

SUMÁRIO

1 INTRODUÇÃO

A Internet das Coisas (Internet of Things - IoT) é definida como a interconexão de dispositivos do nosso cotidiano com a Internet por meio de identificadores únicos (Id) e de suas respectivas propriedades, o que possibilita realizar a coleta de informações sobre o dispositivo e seu ambiente, assim como operações de alterações no seu estado atual a partir de qualquer lugar (KIN; RAMOS; MOHAMMED, 2017).

A IoT oferece possibilidades de integração entre os dispositivos, contudo, a segurança das informações coletadas por estes é um grande desafio. Com bilhões de dispositivos conectados, gerando informações com níveis variados de importância, a ameaça de roubo dessas informações é um fator relevante. A empresa International Data Corporation - IDC, um dos principais fornecedores globais de inteligência de mercado, acredita que até o ano de 2024 haverá um gasto de aproximadamente 742 bilhões em IoT, gerando 79,4 zettabytes (ZB) de dados (FRAMINGHAM, 2020). Desta forma, é imprescindível construir soluções que proporcionem níveis de segurança apropriados para os dispositivos IoT que se comunicam através de redes de comunicação.

Neste contexto surge o Blockchain, definido como sistema de cadeia de blocos criptografados, que oferece recursos relevantes para o fornecimento de privacidade e segurança distribuídas na Internet. A estrutura do Blockchain permite a criação de um banco de dados inviolável configurado em uma rede peer-to-peer (LEAL, 2017). Diante deste cenário, o Blockchain se torna um grande aliado para os dispositivos IoT, oferecendo serviços de segurança de dados como imutabilidade e não-repudiação.

Contudo, um fato relevante neste cenário é que os protocolos iniciais de redes Blockchain foram desenvolvidos para serem aplicados nas transações de criptomoedas e, portanto, a utilização no cenário da Internet das Coisas não foi seu objetivo primário (PANARELLO, 2018). Os desafios para realizar a implementação neste contexto tem motivado a proposição de novos trabalhos, os quais visam em sua grande maioria mitigar os problemas de interoperabilidade entre os dispositivos IoT e o Blockchain. Pode-se afirmar também que existe a necessidade de integração dessas soluções com o usuário final, uma vez que dificuldades de configuração ou usabilidade tendem a aumentar a não aceitação da solução pelo mercado consumidor (DEDEOGLU, 2020).

Diante das dificuldades de integração de dispositivos IoT e Blockchain, o desenvolvimento de um modelo arquitetural voltado para este fim pode proporcionar um serviço de integração transparente dessas tecnologias para o usuário final. Desta

forma, este trabalho visa propor uma arquitetura para transparência de conectividade de serviços Blockchain em sistema IoT.

Esta arquitetura objetiva diminuir a complexidade de integração para o usuário final, possibilitando uma melhor experiência no uso de dispositivos IoT em conjunto com Blockchain.

1.2 JUSTIFICATIVA

Dentro da temática desta dissertação, diversas iniciativas trazem propostas para o uso de Blockchain dentro do contexto da Internet das Coisas. Por exemplo, os autores ALI, DOLUI e ANTONELLI (2017) propõem uma arquitetura com objetivo de facilitar a comunicação dos dispositivos IoT com Blockchain utilizando o armazenamento de dados no formato do mecanismo peer-to-peer. A arquitetura é baseada em uma rede Blockchain privada de usuários e adaptável para diversos casos de uso de IoT.

Por sua vez, o autor KOŠŤÁL (2019) propõe uma arquitetura aprimorada com o objetivo de gerenciar e monitorar dispositivos IoT utilizando um Blockchain privado. A principal característica é o gerenciamento distribuído de arquivos de configurações de dispositivos IoT em redes Blockchain. A arquitetura permite que configurações dos dispositivos IoT sejam armazenadas no Blockchain, onde, na necessidade de atualização do firmware dos dispositivos IoT, essas configurações sejam baixadas.

Outra iniciativa de pesquisa relevante no contexto desta dissertação é apresentada por PINNO, GRÉGIO e DE BONA (2017), que propôs uma arquitetura totalmente descentralizada, transparente e compatível a uma variedade de modelos de dispositivos IoT. O objetivo proposto pelos autores é uma arquitetura baseada em Blockchain para autorização de acesso entre dispositivos IoT, permitindo estabelecer um relacionamento seguro e transparente entre eles.

Como observado anteriormente, diversos autores provêm arquiteturas para integrar dispositivos IoT em redes Blockchain, no entanto, suas soluções não permitem estabelecer a transparência de conectividade entre essas tecnologias com o usuário final. Por isso, este trabalho objetiva propor uma arquitetura para prover transparência de conectividade de serviços Blockchain em sistemas IoT, analisando a integração dessas tecnologias na visão do usuário final.

Sendo assim, a compreensão deste estudo decorre pelo desenvolvimento de um sistema de software para gerenciar métodos no processo de recebimento de dados

advindo de dispositivos IoT e encaminhá-lo para uma rede Blockchain. Por fim, está arquitetura irá proporcionar uma perspectiva real sobre a efetividade da sua utilização em soluções de problemas reais, identificando suas limitações com o objetivo que atenda cenários de larga escala.

1.3 OBJETIVOS E CONTRIBUIÇÕES

1.3.1 Objetivo Geral

O objetivo geral deste trabalho é propor uma arquitetura para transparência de conectividade de serviços Blockchain em sistemas IoT. Esta arquitetura deve proporcionar transparência de utilização para o usuário final.

1.3.2 Objetivos Específicos

Para atingir o objetivo geral, foram definidos alguns objetivos específicos a serem alcançados. Estes objetivos específicos se encontram descritos a seguir.

- Investigar os tipos de transparências utilizadas em Sistemas Distribuídos e selecionar dentre eles quais serão considerados dentro da arquitetura proposta;
- Elaborar algoritmo para controle, recepção, autenticação, validação e transmissão dos dados dos dispositivos IoT na Blockchain;
- Propor e executar estudos de casos para ilustrar e avaliar a arquitetura proposta nesta dissertação.

1.4 ESTRUTURA DA DISSERTAÇÃO

Este trabalho está dividido da forma que se segue.

Inicialmente, o capítulo 2 apresenta a fundamentação teórica desta dissertação. Dentre estes conceitos se destacam tópicos sobre Internet das Coisas e suas especificidades, assim como a estrutura de redes Blockchain. O capítulo 2 apresenta ainda uma visão sobre a transparência utilizada em sistemas distribuídos.

Por sua vez, o capítulo 3 apresenta os trabalhos relacionados com o objetivo central desta dissertação. Para isso, são detalhados os principais trabalhos sobre

conectividade de dispositivos IoT em serviços de Blockchain com o objetivo de apresentar e analisar o que já foi realizado em torno de soluções nesta área.

O capítulo 4 apresenta a contribuição principal deste trabalho, onde é detalhada toda a estrutura da solução proposta assim como a sua implementação. Por sua vez, o capítulo 5 detalha a avaliação da solução proposta e os resultados obtidos por meio de dois estudos de casos distintos, envolvendo análise do funcionamento do sistema.

Por fim, o capítulo 6 apresenta as conclusões finais desta dissertação, incluindo as limitações desta pesquisa e os trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo, serão abordados conceitos básicos sobre a temática deste trabalho. Inicialmente, na Seção 2.1, a Internet das Coisas será apresentada, considerando sua estrutura e os conceitos básicos que a fundamentam. Em seguida, na Seção 2.2, o Blockchain é apresentado a partir dos fundamentos e conceitos que permeiam esta tecnologia. Na Seção 2.3, conceitos básicos relacionados à transparência de Sistemas Distribuídos são abordados. Por fim, a Seção 2.4 apresenta as considerações finais deste capítulo.

2.1 INTERNET DAS COISAS

Apesar da Internet das Coisas não ser especificamente uma nova tecnologia, e sim um conceito que integra diversas tecnologias e plataformas, a sua existência e expansão dependem de um conjunto de normas e padrões. Esses padrões acabam por gerar parâmetros que permitem a melhoria técnica dos sistemas baseados em IoT, e, a partir disso, a criação de novos produtos, aplicações e plataformas (KHAN; SALAH, 2018).

Sistemas baseados na Internet das Coisas possuem uma variação expressiva de tamanho, dependendo da sua área de aplicação e da quantidade de sensores e atuadores conectados na rede. Através do uso de redes de comunicação é possível conectar diferentes tipos de dispositivos e armazená-los em nuvens ou servidores (HUREL; LOBATO, 2018).

Sistemas IoT podem ser compostos por um ou mais dispositivos (ou “coisas”), portanto, o problema de segurança está mais acentuado quando o cenário é de maior proporção, onde a escalabilidade torna a segurança mais complexa (MADAKAM, 2015).

Independentemente do aumento de dispositivos IoT conectados à rede, a aplicação dos princípios básicos de segurança devem ser considerados, como: Confidencialidade, Integridade e Disponibilidade (PRATES, 2018).

A confidencialidade dos dados consiste em garantir e restringir o acesso destes dados somente aos dispositivos e usuários cadastrados e devidamente autorizados de acordo com o nível de criticidade do dado. Para proteger os dados gerados no sistema, o

nível de proteção necessário poderá ser gerenciado de acordo com a importância de cada informação dentro da rede (GOMES, 2019).

Para obter a integridade dos dados é necessário a implementação de mecanismos de segurança para preservar os dados gerados pelos dispositivos IoT, fundamentando assim, o aumento da confiabilidade da rede de Internet das Coisas (PAUFERRO; DE PAIVA; LESSA, 2020). É importante utilizar recursos de segurança que possam verificar todas as modificações realizadas em arquivos com objetivo de buscar alterações indevidas aos dados dentro de um sistema IoT.

Já a disponibilidade dos dados gerados dentro da rede de Internet das Coisas precisa estar sempre acessível sempre que necessário por usuários e/ou outros dispositivos devidamente autorizados.

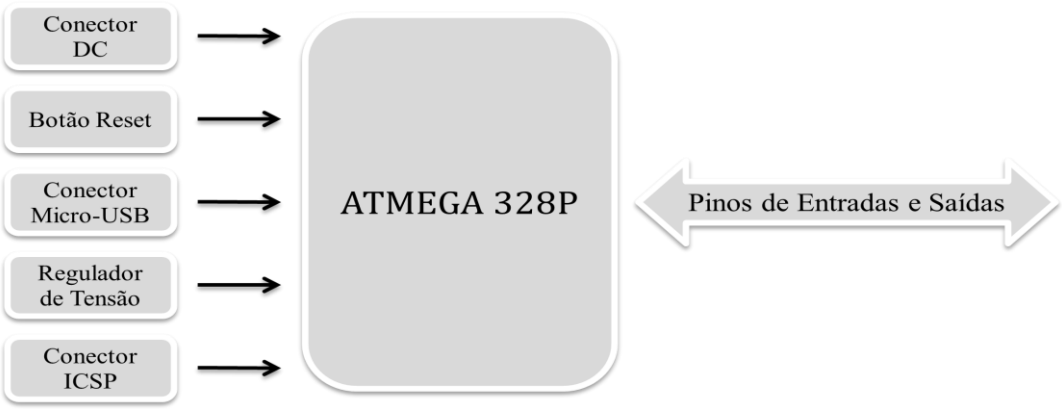
2.1.1 PLATAFORMAS PARA A INTERNET DAS COISAS

Para a conectividade de diversos tipos de objetos à internet, existem diversos tipos de plataformas que podem ser utilizadas para a realização da coleta, processamento e envio dos dados.

O Arduíno é uma dessas plataformas que estão sendo amplamente utilizada por diversos tipos de públicos, e o mesmo adota o conceito de hardware livre, onde usuários podem montá-lo, modificá-lo e personalizá-lo, partindo do mesmo hardware básico (MCROBERTS, 2018).

O Arduíno é composto por um “Microcontrolador Atmel” e circuitos de entradas e saídas, podendo ser programado através de um Integrated Development Environment IDE, baseado na linguagem de programação C / C++. A plataforma do Arduíno se encontra apresentada na Figura 2.1.

Figura 2. 1. Diagrama de blocos - Plataforma Arduino.



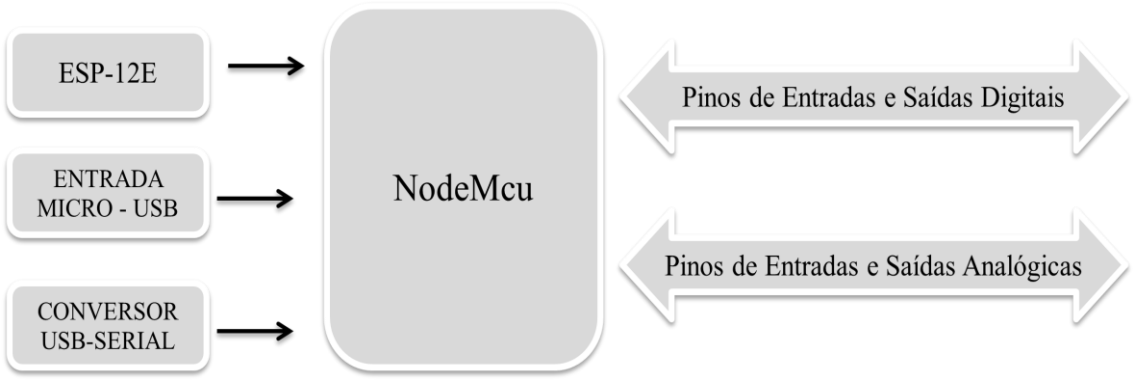
Fonte: Autor deste Trabalho.

Devido à necessidade de conectar o Arduino à Internet, a “Ethernet Shield W5100" foi desenvolvido. O módulo apresenta componentes necessários para a conexão com a Internet, onde poderá ser integrado a projetos de automação (SHIELD, 2016). O Ethernet Shield é composto pelo chip Wiznet W5100, suportando no máximo 4 conexões simultâneas.

É possível conectar sensores, atuadores ou outros dispositivos ao Arduino, podendo assim automatizar e controlar o meio em que está inserido. Sua portabilidade e relativo baixo custo tornam essa plataforma uma opção interessante para aplicações IoT (RAY, 2018).

A plataforma NodeMCU, outra plataforma disponível para sistemas IoT, é um dispositivo baseado na linguagem de programação “LUA”, possuindo como característica código aberto desenvolvido para trabalhar com o chip Wifi ESP8266 (ŠKRABA, 2017). O NodeMCU possui Wifi nativo, com a pilha do protocolo TCP/IP integrada, permitindo o acesso à rede Wifi com uma interface usb-serial e uma tensão regulador de 3.3 volts (KASHYAP, 2018). Os padrões wireless utilizados no chip ESP8266 são IEEE 802.11b, IEEE 805.11g e IEEE 805.11n, contento faixa de frequência 2.4 GHz e uma taxa de transmissão que varia entre 100 a 460 Mbps. Esta plataforma se encontra ilustrada na Figura 2.2.

Figura 2. 2. Plataforma NodeMcu.



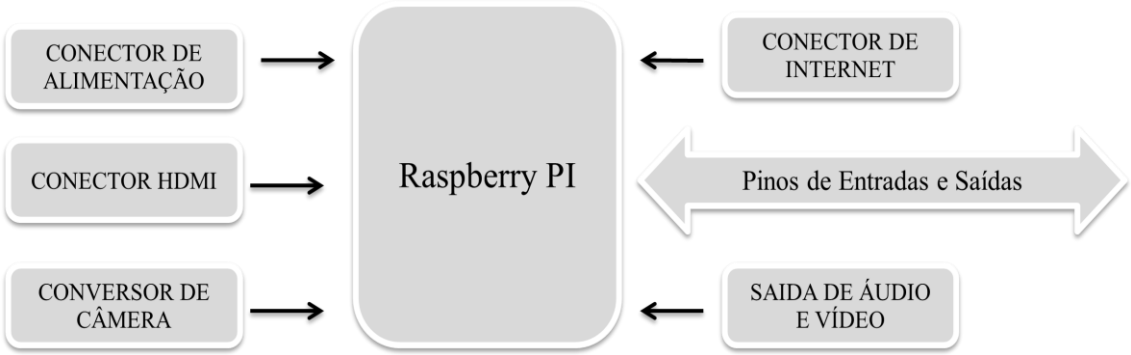
Fonte: Autor deste Trabalho.

Para atualização do firmware do NodeMCU é necessário conectar-se a uma das duas IDEs suportadas, ESPlorer IDE ou Arduíno IDE, permitindo que programas sejam executados e carregados em sua própria memória.

A utilização deste microcontrolador pode ocorrer em inúmeras aplicações, já que conta com poder de processamento e conexão Wifi embutido. O NodeMcu ESP8266 também poderá ser usado como um módulo com outros microcontroladores, a exemplo da placa arduíno, o que permite a recepção e o envio de dados para um aplicativo ou sistema Web.

A última plataforma a ser abordada nesta subseção é o Raspberry Pi, que é o nome de uma série de computadores de placas únicas desenvolvidas pela empresa "Raspberry Pi Foundation", lançando sua criação no ano de 2012 (RICHARDSON; WALLACE, 2012). Seus principais componentes são entrada para fonte de alimentação, entradas USB para conexão de periféricos (mouse, teclado, pendrive), entrada HDMI para conexão de monitor ou TV, saída para conexão de caixas de som, conexão para Internet, processador e memória, como ilustrado na Figura 2.3.

Figura 2. 3. Plataforma Raspberry Pi.



Fonte: Autor deste Trabalho.

Alguns recursos do Raspberry Pi podem variar dependendo do modelo da placa, podendo ainda executar sistemas operacionais convencionais e consequentemente trabalhar com diversas linguagens de programação (VUJOVIĆ, 2015).

O Raspberry Pi foi projetado para ser aplicado em diversas áreas da IoT, sendo uma plataforma portátil e de relativo baixo custo, podendo ser aplicável em residências, indústrias e áreas afins (DE OLIVEIRA, 2017).

2.1.2 PROTOCOLOS DE COMUNICAÇÕES

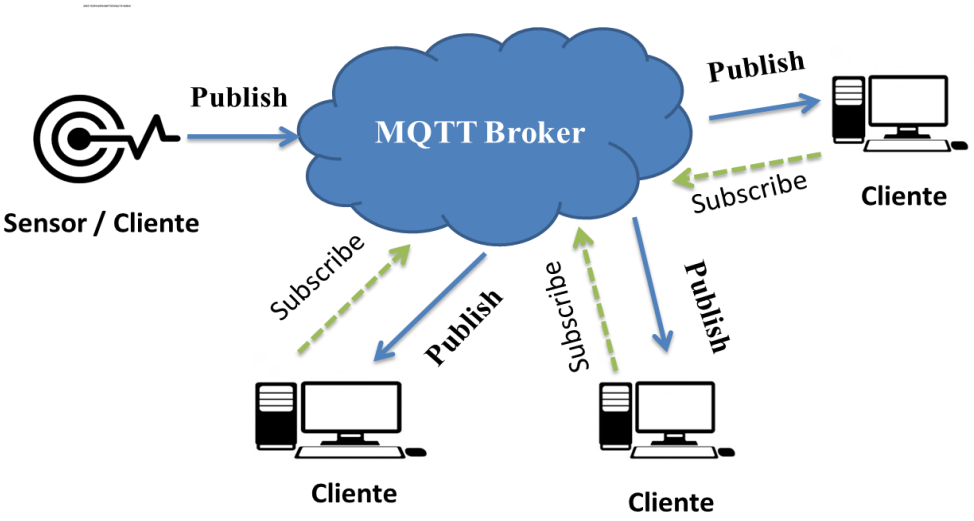
Com a necessidade de conectar uma variedade de dispositivos inteligentes entre si, foram implementados protocolos de comunicações com o intuito de uma maior abrangência e eficiência, como o MQTT e o REST. Os dois protocolos mais utilizados são amplamente aplicados por possuir leveza na transmissão de dados e na amplitude (COSMI; MOTA, 2019). Os protocolos na camada de aplicação IoT permitem uma melhor interação entre os dados coletados por sensores.

O protocolo Message Queue Telemetry Transport – MQTT é descrito como um protocolo destinado a comunicação IoT Machine-to-Machine (M2M) – Máquina à Máquina e sua principal característica é a leveza no transporte de mensagens (SONI; MAKWANA, 2017). Sua criação foi realizada no ano de 1990 pela empresa International Business Machines Corporation – IBM, com o objetivo de ser utilizado em sistemas de supervisões de dados realizando aquisições na área industrial (YASSEIN, 2017).

O conceito básico de aplicação deste protocolo é o modelo de “Publicação e Assinatura – Publish / Subscribe”, sua configuração de envio de mensagens são baseados no uso da criptografia a qualidade de serviços, Quality of Services – QoS.

Neste padrão “Publish / Subscribe”, quando um dispositivo da rede IoT deseja receber uma mensagem de outro dispositivo dentro da rede, ele a subscreve, fazendo uma requisição para outro dispositivo da rede onde irá gerir as publicações e subscrições (LIGHT, 2017). Este terceiro elemento é denominado de “Broker”, que será o intermediário no processo da comunicação entre os dispositivos (TORRES; ROCHA; DE SOUZA, 2016), ilustrado na Figura 2.4.

Figura 2. 4. Protocolo MQTT



Fonte: Autor deste Trabalho.

O MQTT está aplicado na camada Open System Interconnection – OSI como também o Hypertext Transfer Protocol – HTTP. A vantagem do MQTT sobre o HTTP é a segurança e apresentação de mais níveis de serviços, permitindo uma comunicação de 1 para N (KANG, 2017).

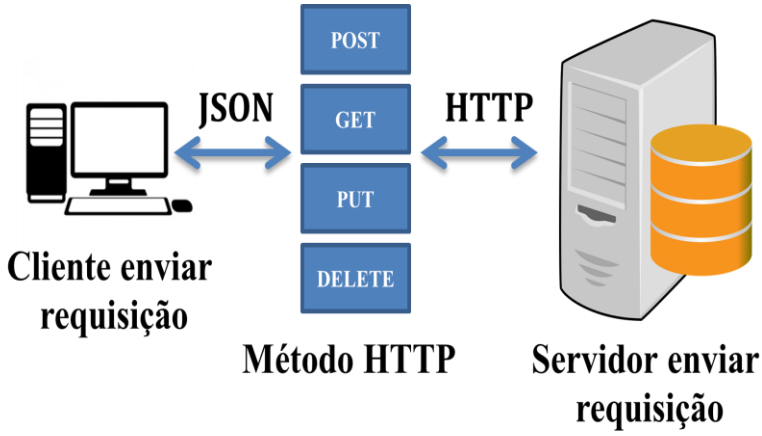
O protocolo Representational State Transfer – REST, Transferência de Estado Representacional, onde sua caracterização é abstração da arquitetura Web, com aplicação das principais operações POST, GET PUT e DELET (BENAVIDEZ GOMES, 2019). As requisições utilizadas para aplicação deste protocolo têm uma resposta dos dados no formato XML, HTML e JSON (CÂNDIDO, 2019).

Este protocolo foi criado no ano de 2000 por Roy Fielding, com objetivo de realizar uma comunicação em rede que também poderia ser utilizada em aplicações IoT

(CANGIANELLI, 2019). A conexão entre os dispositivos IoT utilizando o protocolo REST é utilizada na interface HTTP, podendo assim se comunicar com à internet e outros sistemas integrados, como ilustrado na Figura 2.5.

O REST é uma conexão unilateral e intermitente, ou seja, os dispositivos apenas se comunicam com o servidor quando necessário ou solicitado (PEREIRA, 2018). O servidor aguarda uma requisição do usuário/dispositivo para responder. Quando o dispositivo envia dados para o servidor, ele faz uma requisição de chamada HTTP POST e quando o dispositivo deseja obter dados do servidor ele faz uma chamada HTTP GET (DO ESPIRITO SANTO; ORDONEZ; RIBEIRO, 2018).

Figura 2. 5. Protocolo REST.



Fonte: Autor deste Trabalho.

Este protocolo REST facilita para que os dispositivos conectados na rede possam facilmente disponibilizar as informações de estados na rede, possibilitando uma padronização na criação, na leitura, atualização e na exclusão desses dados, sendo utilizado por vários servidores da Web (SOUZA, 2018).

2.2 BLOCKCHAIN

Blockchain pode ser visto como um registro público de todas as transações realizadas dentro da rede. A sua grande vantagem é a segurança associada a essas transações, pois as informações não podem ser alteradas ou excluídas (NOFER, 2017).

O Blockchain surgiu juntamente com o Bitcoin no ano de 2008, em um artigo publicado por Satoshi Nakamoto, mas apenas em 2009 foi lançado em código aberto para o público (GUPTA, 2017).

Cada bloco dentro de um Blockchain possui uma assinatura digital, que é conhecida como hash. Esse hash também funciona como um elo entre os blocos, dado que cada bloco possui o seu próprio hash e o hash do bloco anterior, ligando vários blocos como uma cadeia de informações (AHRAM, 2017).

O sistema Blockchain, para ter seus registros alterados, é necessário alterar mais de 50% dos hash's espalhados por toda a rede, realizando um ataque denominado 51% (KSHETRI; VOAS, 2018).

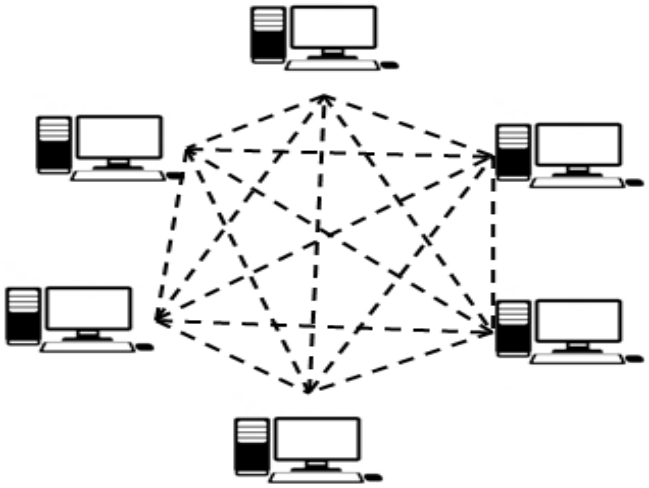
2.2.1 TIPOS DE BLOCKCHAIN

Atualmente, podem-se dividir os Blockchains em: Blockchain público, Blockchain privado e Blockchain consórcio (DE LIMA; MONTEIRO, 2020), onde serão descritos a seguir.

2.2.1.1 Blockchain Público

Na rede de Blockchain pública, os usuários não se conhecem. Portanto, o nível de confiança entre os usuários da rede é bastante baixo, tendo como característica uma rede aberta e as informações contidas dentro dela são de domínio público, como ilustrado na Figura 2.6.

Figura 2. 6. Estrutura Arquitetural da Blockchain Pública.



Fonte: Autor deste Trabalho.

Devido à liberdade de acesso à rede pública, qualquer nó contido nela poderá visualizar, ler e gravar dados, pois não necessita de autorização para poder participar e interagir com a rede de Blockchain pública (MOUGAYAR, 2018).

Nenhum participante de uma Blockchain pública possui o controle sobre os dados e, portanto, uma vez que os dados sejam inseridos na rede pública eles não poderão ser modificados ou excluídos depois de serem validados através do consenso na rede, garantindo a imutabilidade dos dados registrados nos blocos (LAVINA, 2018).

Segundo Fontalba (2019) as principais características da rede Blockchain Pública são: leitura e gravação aberta aos participantes, baixa velocidade nas transações, plenamente descentralizadas.

Uma Blockchain pública é distribuída e gerenciada por redes peer-to-peer (p2p), permitindo que os dados sejam registrados e armazenados em diversos servidores e sua qualidade de dados é mantida pela replicação do banco de dados e pela confiança computacional (GUINGO, 2020). Adicionalmente, a validação e a verificação das transações são demoradas, necessitando uma sobrecarga computacional mais elevada (BERNAL BLAY, 2018).

A validação dos blocos é dada através de algoritmos dedicados (proof of work) usados para verificar a legitimidade de uma determinada transação e, assim, produzir a criação da cadeia de blocos. Quando comparada com o Blockchain privado, o Blockchain público é considerado mais descentralizado. Contudo, esta maior

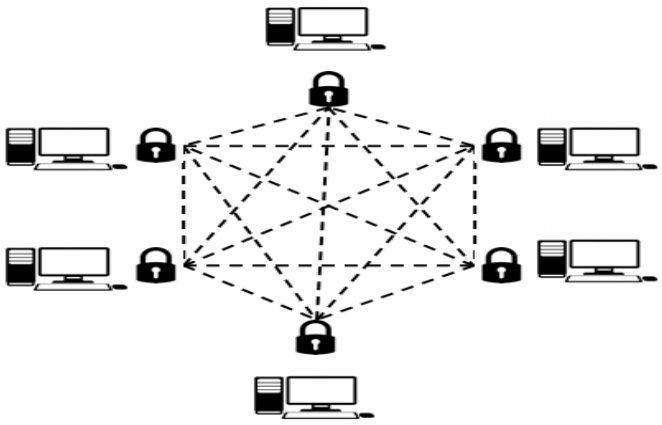
descentralização acaba o tornando mais lento que os Blockchain privados (CARVALHO, 2019).

2.2.1.2 Blockchain Privado

No Blockchain privado, o acesso à rede é restrito aos participantes cadastrados, ou seja, os participantes devem possuir uma permissão prévia (EVARISTO, 2019). Desta forma, a confiança entre os usuários da rede no Blockchain privado é mais elevada que no Blockchain público, podendo realizar uma quantidade maior de transações, dado que o compartilhamento e atualização de algoritmos são mais simples e rápidos (LINARES BARBERO, 2019).

No Blockchain privado, a realização da leitura, escrita e auditoria dos dados requer uma autorização específica para os participantes, como ilustra a Figura 2.7, garantindo um nível mais alto de segurança e privacidade dos dados (SILVA, 2018). De acordo com o contrato estabelecido entre os participantes na rede privada, os dados poderão ser exclusivos apenas para uma parte da rede.

Figura 2. 7. Estrutura Arquitetural de Blockchain Privado.



Fonte: Autor deste Trabalho.

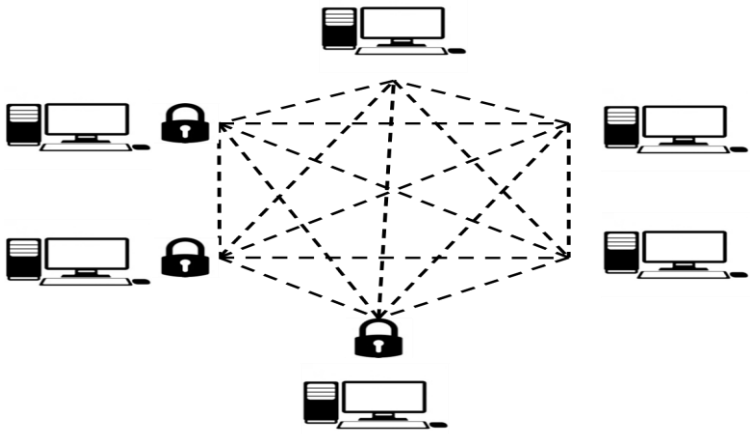
2.2.1.3 Blockchain Consórcio

O Blockchain consórcio é considerado uma junção dos Blockchain públicos com o privado. A principal diferença entre o Blockchain consórcio dos demais Blockchain é o nível de consenso a rede, dado que, o Blockchain público é considerado uma rede aberta em que qualquer um pode validar os blocos.

Por sua vez, no Blockchain privado apenas os participantes credenciados tem acesso a rede. Por fim, no Blockchain consórcio as regras de acesso a rede são flexíveis, sendo que a visibilidade dos dados na cadeia de blocos serem limitadas aos validadores e visíveis aos nós devidamente autorizados, como ilustra a Figura 2.8.

A aplicação do Blockchain consórcio é recomendada em cenários em que várias organizações operam em um mesmo setor e precisam realizar o compartilhamento das informações em ambiente comum (MIERS, 2019).

Figura 2. 8. Estrutura Arquitetural de Blockchain Consórcio.



Fonte: Autor deste Trabalho.

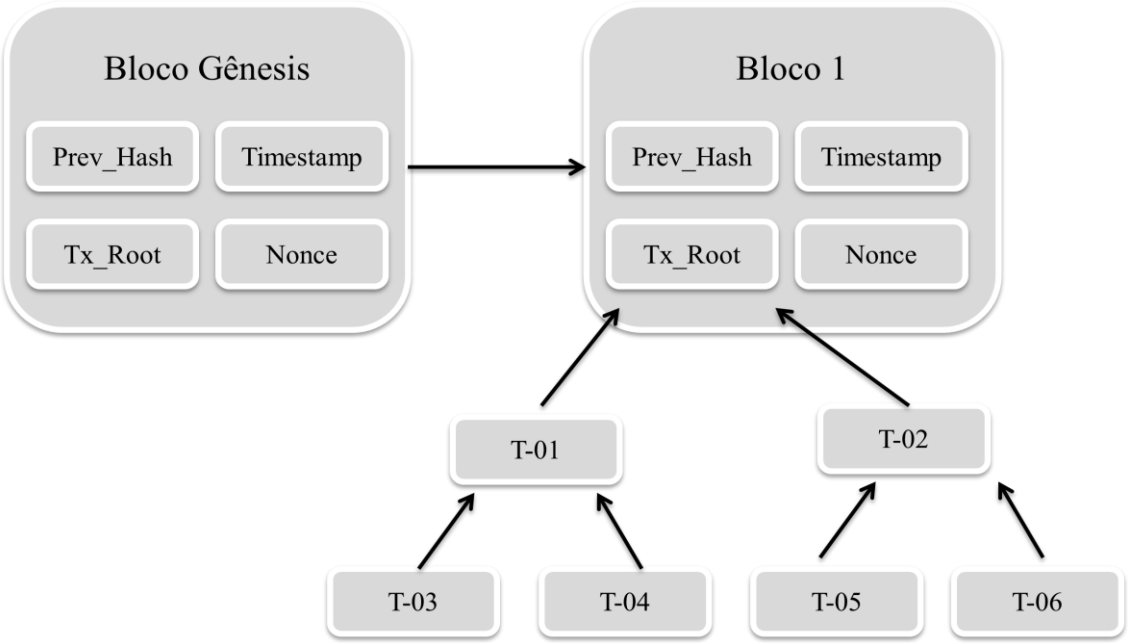
2.2.2 MINERAÇÃO DOS BLOCOS

A mineração é o processo responsável por adicionar transações ao Blockchain, através dos nós que compõe a rede, onde são chamados de mineradores. O objetivo dos mineradores é encontrar uma sequência numérica “Hash” compatível entre o bloco atual com outro bloco anterior já criado e, para isso, o algoritmo irá efetuar milhares de cálculos para encontrar esta sequência “Hash” (CHICARINO, 2017). Após o encontro dos valores compatíveis o minerador recebe uma determinada recompensa pelo bloco minerado, com o objetivo de incentivar o processo de mineração e manter a rede em funcionamento (RODRIGUES, 2018).

O minerador que primeiro encontrar o Hash deve informar para toda a rede com o intuito de validar sua mineração. Neste contexto, a aprovação de cada transação depende de um consenso entre os nós da rede, prevenindo assim, fraudes.

Os mineradores, depois de receber uma mensagem com uma nova transação adicionada na rede, ele armazena essa transação em uma base de dados de transações ainda não mineradas. Essas transações irão permanecer temporariamente em uma de fila de espera com prioridade, aguardando que seja incluída para validação em um bloco. Após selecionar quais transações serão incluídos nos blocos, será gerada uma árvore de merkle, onde as transações passam por Hash's sucessivos até que resulte em um único Hash, como ilustra a Figura 2.9.

Figura 2. 9. Estrutura de Mineração de Blocos.



Fonte: Autor deste Trabalho.

Depois da validação por todos nós da rede, o bloco é registrado dentro da cadeia de blocos (Blockchain) de forma cronológica, sendo digitalmente assinada para garantir a veracidade e integridade das informações. Uma cópia do Blockchain é distribuída para cada nó que compõe a rede, o que permite que esses nós possam comparar e validar as informações contidas no Blockchain de acordo com os dados que os nós possuem de cada registro.

A ação de minerar dados nos blocos é também conhecida como Proof of work ou prova de trabalho, que são algoritmos que processam a entrada de dados, como merkle root, timestamp, hash do bloco anterior e o nunce. É evidentemente que a prova de trabalho não o único algoritmo ou protocolo de consenso existente, como pode ser

citado o Proof of Stake, Delegated proof of stake, Proof of Authority, Proof of Importance e Delegated Byzantine Fault Tolerance, entre outros.

2.3 TRANSPARÊNCIA - SISTEMAS DISTRIBUÍDOS

Um sistema distribuído tem como objetivo facilitar o acesso ao usuário final dos recursos remotos existentes e compartilhar esses recursos de forma controlada (NETO; SANTOS; ARAUJO, 2017). Uma das principais características de um sistema distribuído é a transparência, que objetiva ocultar os processos e/ou recursos que estão distribuídos através de vários componentes (CREEL, 2020). A transparência aplicada no sistema deve possibilitar que o sistema se apresente para o usuário como um sistema único, otimizando o compartilhamento e gerenciamento de recursos (LOPER, 2019).

Segundo Coulouris (COULOURIS 2013), os tipos de transparências aplicadas em sistemas distribuídos são: de acesso, de localização, de concorrência, de replicação, de falhas, de mobilidade, de desempenho, de escalabilidade.

A transparência de acesso permite que o usuário acesse o sistema de forma que não haja nenhuma diferença entre os métodos utilizados, sejam esses métodos locais ou remotos.

Na transparência de localização os detalhes da topologia do sistema não devem ser conhecidos pelo usuário. Isto significa que o objeto dentro do sistema não deverá ser visível, como também a sua localização de persistência (seja dos dados ou dos recursos utilizados). Portanto, o acesso aos componentes deve ser acessível sem o conhecimento da sua localização física.

Porém, na transparência de concorrência a sua aplicação é dada quando dois ou mais usuários estão acessando os mesmos recursos do sistema sem que haja a percepção, em termos de desempenho, da utilização concorrente. Por sua vez, a transparência de replicação garante que os dados ou recursos do sistema sejam replicados sem a necessidade de conhecimento por parte do usuário.

No entanto, na transparência de falhas o sistema deve ter a capacidade de ocultar ou solucionar as falhas existentes de forma transparente para o usuário. Na transparência de mobilidade o objetivo é ocultar do usuário a migração dos recursos ou dados, de forma a não afetar a utilização do sistema.

Já no caso da transparência de desempenho, novos recursos são adicionados ao sistema para melhorar a sua performance sem que esta ação seja percebida pelo usuário. Por fim, a transparência de escalabilidade permite a expansão do sistema e seus recursos sem exigir mudanças significativas dentro da estrutura e de forma transparente ao usuário.

2.4 CONSIDERAÇÕES FINAIS DO CAPÍTULO

Neste capítulo, foram vistos os principais conceitos básicos referentes à temática desenvolvida nesta dissertação. Foram apresentados conceitos envolvendo a Internet das Coisas e suas características e aspectos importantes sobre os tipos de plataformas utilizadas. Após isso, foram apresentadas informações importantes acerca de Blockchain e, por fim, foram destacados aspectos importantes sobre os tipos de transparências aplicadas em sistemas distribuídos.

No próximo capítulo, serão apresentados e discutidos os principais trabalhos relacionados a esta dissertação.

3 TRABALHOS RELACIONADOS

Neste capítulo, serão abordados os principais trabalhos relacionados com o desenvolvimento desta dissertação, que se refere ao estado da arte sobre arquiteturas para integração entre dispositivos IoT e Blockchain. Deste modo, este capítulo está dividido da seguinte forma. Inicialmente, na Subseção 3.1, está uma visão geral das propostas apresentadas em trabalhos relacionados com arquiteturas para integração entre dispositivos IoT e Blockchain. Em seguida, na Subseção 3.2 é realizado um resumo comparativo dos principais trabalhos relacionados com esta pesquisa. Por fim, a Seção 3.3 apresenta as considerações finais deste capítulo.

3.1 TRABALHOS RELACIONADOS QUE FOCAM NA INTEGRAÇÃO DE DISPOSITIVOS IOT COM BLOCKCHAIN

Atualmente, existem diversas propostas de uso de Blockchain para sistemas IoT com o intuito de resolver os problemas de segurança dos dados. Diante desta necessidade, alguns autores trazem em suas pesquisas estas propostas como uma solução para integridade, confiabilidade e segurança das informações geradas dentro do sistema IoT.

Os autores SIMIĆ, SLADIĆ e MILOSAVLJEVIĆ (2017) definem um modelo arquitetural de Blockchain aplicado à IoT na área da saúde, onde o profissional da saúde poderá inserir e monitorar os dados clínicos dos pacientes de forma segura. O objetivo deste trabalho é mostrar uma possibilidade de armazenamento de dados de forma confidencial e segura, combinando Internet das Coisas, Blockchain e Big Data.

Os autores trazem os principais tópicos para o desenvolvimento teórico, destacando os benefícios mais relevantes dessas tecnologias aplicadas na área da saúde, como também descrevem como os dispositivos IoT podem ser utilizados como fonte para coletar dados dos pacientes e compartilhar entre as instituições de saúde. Por fim, os autores defendem a implementação e testes do sistema em um ambiente real, fazendo o uso de um Blockchain público ou privado.

Outro projeto relevante o de DORRI (2017), onde o mesmo se baseia em um modelo arquitetural de Blockchain privado aplicado em residências, gerenciando as informações externas e internas da casa. O trabalho apresenta como diferencial o uso do minerador interno, responsável por validar as informações de entrada no sistema Blockchain, tendo como os principais fundamentos a confiabilidade, a imutabilidade e a ininterruptibilidade do sistema.

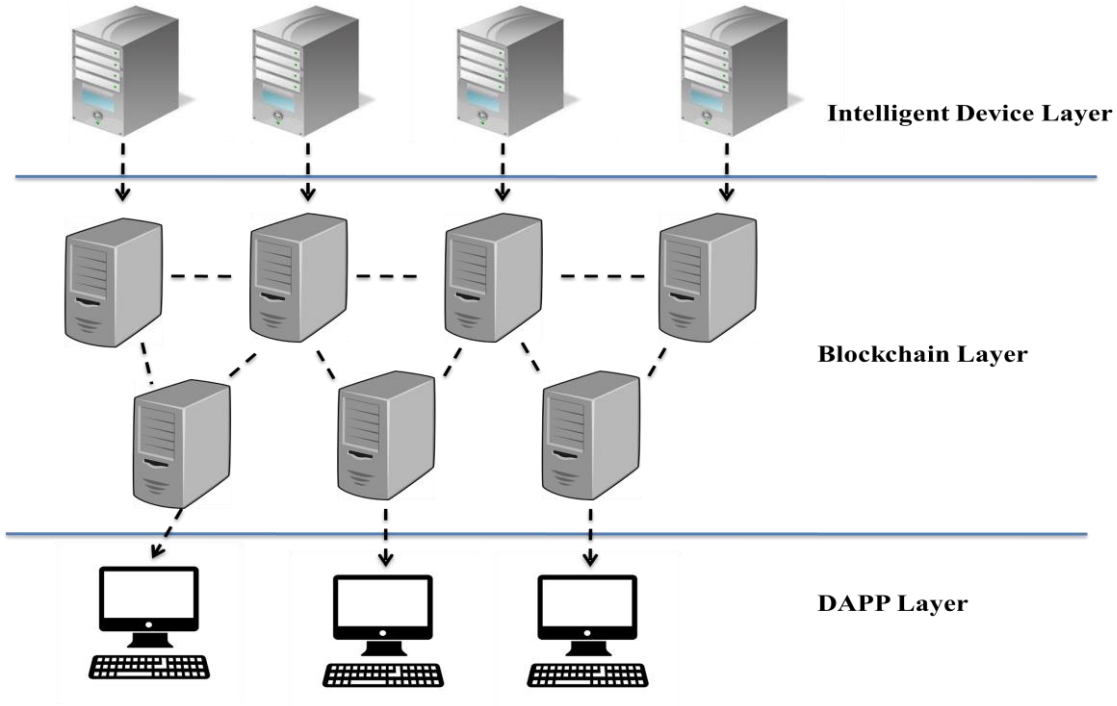
As informações geradas dentro do sistema são acessadas apenas por usuários devidamente credenciados, garantindo assim que essas informações sejam privativas unicamente ao grupo de usuários da rede Blockchain privada. Para avaliar o desempenho, o sistema foi simulado em um cenário de casa inteligente no COOJA (simulador de aplicações do sistema operacional Contiki), utilizando três sensores mote Z1 (que imitam dispositivos IoT domésticos), enviando dados a cada 10 segundos.

As métricas aplicadas para avaliação deste trabalho foram sobrecarga de pacote (comprimento da transmissão do pacote), a sobrecarga de tempo (tempo de processamento para cada transação no minerador) e o consumo de energia (energia consumida pelo minerador para processar as transações). Os resultados mostraram um bom desempenho nos resultados obtidos a partir de experimentos indicando a viabilidade de uso da arquitetura proposta.

Já YU (2018) propõe como principal contribuição o projeto de uma plataforma Blockchain para dispositivos inteligentes, utilizando arquitetura de rede distribuída e o mapeamento inteligente entre os nós de dispositivos inteligentes na rede. A plataforma projeta um algoritmo de consenso Blockchain para os dispositivos IoT, fornecendo maior eficiência no consenso e garantindo a descentralização, como também, proporcionando maior prestabilidade.

Para o desenvolvimento deste trabalho, foi adotada uma arquitetura em três camadas (camada de dispositivos IoT, camada de Blockchain e camada DAPP). Os dados coletados do ambiente são inseridos e enviados da camada de dispositivos IoT para a camada de Blockchain, onde são compartilhados na rede que a compõem. Por fim, a camada DAPP (Aplicação Descentralizada) é responsável pelo fornecimento de serviços de interface de dados para os usuários da rede, conforme ilustra a Figura 3.1.

Figura 3. 1. Uma arquitetura de rede de plataforma de Blockchain de dispositivo inteligente.



Fonte: YU (2018)

Para avaliar o trabalho proposto, foi realizado um mapeamento da taxa de transferência de dados por todas as camadas da arquitetura proposta e comparado com o desempenho de Blockchain público da Ethereum e da Bitcoin. Os resultados mostraram que a arquitetura proposta tem maior taxa de transferência e menor latência de transação do que Bitcoin e Ethereum.

Os autores ALI, DOLUI e ANTONELLI (2017), por sua vez, desenvolveram uma arquitetura usando duas plataformas de aplicativos Blockchain existentes: Ethereum, uma plataforma de desenvolvimento Blockchain, e o Monax, uma plataforma de desenvolvimento de aplicativos Blockchain para ecossistemas de negócios.

A principal contribuição deste trabalho é inserir no Blockchain uma pilha de software de contratos inteligentes Blockchain e o armazenamento de arquivos ponto a ponto, para dar aos usuários de IoT autoridade sobre seus dados e eliminar a necessidade de gerenciamento centralizado. A arquitetura é destinada a ter privacidade embutida e ser adaptável para vários casos de uso da IoT.

Ao utilizar as duas plataformas para implementar a arquitetura proposta denominada “Arquitetura de Consórcio Modular”, foi construído um Blockchain do tipo consórcio, com o objetivo de validar os dados entre as plataformas Blockchain aplicadas.

As métricas de desempenho utilizadas para avaliar a arquitetura proposta foram a sobrecarga de processamento, a sobrecarga de tráfego de rede e o tempo de processamento dos blocos. Os resultados obtidos revelaram que a sobrecarga de tráfego da rede proveniente de Ethereum se eleva com o aumento de nós no consórcio, contudo, comparado com os resultados obtidos na plataforma Monax os dados são significativamente menores. Porém, quando usado no nível de sidechain, Monax supera Ethereum em termos de sobrecarga de processamento.

Em outra iniciativa de pesquisa relevante, os autores PINNO, GRÉGIO e DE BONA (2017.) propuseram uma arquitetura para controle de acesso baseada em Blockchain. A arquitetura é totalmente descentralizada, ou seja, sem necessidade de terceiros, escalável, transparente, tolerante a falhas e compatível com uma ampla gama de modelos de controle de acesso empregados na IoT.

A arquitetura denominada “ControlChain” foi desenvolvida a partir da junção de quatro arquiteturas de Blockchains diferentes e conectados entre si: o primeiro é o Blockchain de relacionamentos, que é responsável pelo armazenamento das credenciais públicas e relacionamentos de todas as entidades no sistema. Por sua vez, o Blockchain de contexto é responsável por registrar as informações obtidas a partir de sensores, dados processados e entradas manuais. Já o Blockchain tem por objetivo registrar as informações sobre permissões ou negar o acesso ao objeto. Por fim, o Blockchain de regras é aplicado para analisar, prestar contas, realizar auditoria de acessos e para verificar a estabilidade do sistema.

Para avaliar a arquitetura, os autores realizaram a comparação do ControlChain com outras arquiteturas (FairAccess, XACML, OAuth, UMA). As métricas utilizadas foram: scalability (avalia a capacidade de manipular uma quantidade crescente de dados de forma uniforme), faulttolerant (avalia o impacto causado por falhas em dispositivos ou links de comunicação), no third-parties (avalia dependência de terceiros na prevenção e a detecção de fraudes e interferências), new authorization (avalia a latência para fazer ou alterar uma autorização), getauthorization (avalia a latência para obter uma autorização), integratedrelationship (estuda a permissão de relacionamentos diretamente nas regras), compatibility (analisa a compatibilidade das arquiteturas com a abundância de modelos empregados atualmente na IoT), e, por fim, lowobject overhead (avalia o quanto o objeto está sobrecarregado pelo processo de autorização). O resultado das avaliações realizadas pode ser vista na Tabela 3.1.

Tabela 3. 1. Comparação de arquiteturas.

	XACML	Oauth	UMA	FairAccess	ControlChain
Scalability	-	-	-	+	+
Fault tolerant	-	-	-	+ -	+
No third-parties	-	-	-	+	+
New authorization	+	+	+	-	- (*)
Get authorization	+	+	+	- (*)	+
Integr. Relationship	-	-	-	-	+
Compatibility	+	-	-	-	+
Low object overhead	+	+	+	+	+

(*) **Exclusively dependent of the type of proof and dissemination speed of blocks**

Fonte: PINNO, GRÉGIO e DE BONA (2017)

Outro trabalho a ser destacado é de KOŠŤÁL (2019), que propõe uma arquitetura totalmente descentralizada, aplicando o conceito de segurança na verificação da integridade dos arquivos de configurações dos dispositivos IoT. Para evitar leituras indesejadas de outros participantes do Blockchain, uma função criptográfica é usada no chaincode (trecho de código escrito), tornando a rede privada.

Qualquer dispositivo de IoT conectado à arquitetura proposta pode ser gerenciado e monitorado. Se precisar de uma nova configuração, o usuário insere-a no Blockchain, no qual o dispositivo IoT poderá baixar as novas informações e assim, o usuário e/ou dispositivo IoT poderá ser notificado sobre uma configuração bem-sucedida ou mal sucedida.

Para realizar a implementação da arquitetura proposta, foi utilizado o Hyperledger Composer (conjunto de ferramentas para facilitar o desenvolvimento de aplicativos de Blockchain). Uma vantagem da utilização do Hyperledger Composer neste trabalho é gerenciar de forma automática a API Representational State Transfer (REST) de acordo com o modelo de negócios. Esta API é usada pelo servidor modificado Trivial File Transfer Protocol (TFTP) para consultar dados do Blockchain.

Com o objetivo de avaliar a arquitetura proposta, foi implementado um protótipo de gerenciamento de configurações em um ambiente Cisco. A arquitetura foi testada em dois ambientes distintos. Primeiramente, um cenário funcional foi usado para verificar a utilidade do projeto proposto; em seguida, um cenário foi usado para medir os tempos necessários para baixar, aceitar e instalar um arquivo atualizado de configuração. O

objetivo do primeiro cenário de teste é obter com sucesso a configuração do dispositivo via Servidor TFTP, que serviu de intermediário para a API REST do Hyperledger Composer. A nova configuração do dispositivo foi modificada por meio da Interface de Usuário (IU) do composer – playground e aplicada no dispositivo de destino, resultando diretamente na hipótese de poder usar a rede Blockchain para gerenciamento e monitoramento de dispositivos IoT.

O segundo cenário foi voltado para medição de desempenho e tempo. Ao avaliar os resultados, a média do tempo total necessário para o dispositivo fazer o download e atualizar a configuração foi de 402,8 ms, conforme a Tabela 3.2.

Tabela 3. 2. Medição do tempo necessário para obter e instalar o arquivo de configuração atualizado no roteador Cisco

Nr. of Measurement	Download (ms)	Setup (ms)	Total Time (ms)
1	9	394	403
2	10	392	402
3	9	391	400
4	11	404	415
5	11	396	407
6	10	395	405
7	8	399	407
8	11	377	388
9	10	389	399
10	12	390	402

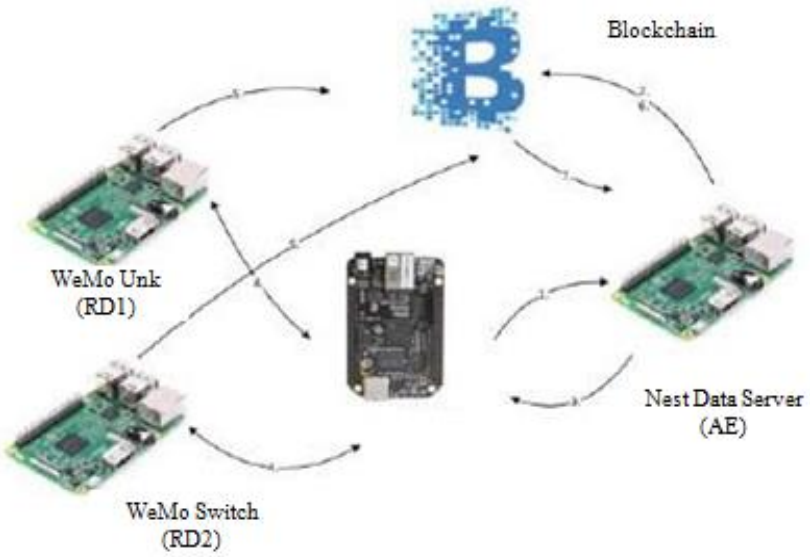
Fonte: KOŠŤÁL (2019).

Por fim, o trabalho apresentado por WU (2018) propõe uma estruturação de autenticação de dois fatores para dispositivos IoT baseados na infraestrutura Blockchain. O objetivo deste trabalho é realizar a autenticação de dois fatores que explora a relação dos dispositivos, suportado pela tecnologia Blockchain, fazendo com que a autenticação possa impedir o acesso de dispositivos maliciosos, mesmo se o primeiro fator falhar.

Foi implementado o sistema integrado IoT e Blockchain com Eris Blockchain (plataforma de desenvolvimento Blockchain). No desenvolvimento para implementação da arquitetura proposta foram utilizados os dispositivos Beaglebone Black e Raspberry Pi 3, ambas são plataformas de prototipagem amplamente utilizadas na Internet das Coisas e sistemas embarcados. Com o intuito de estabelecer o relacionamento dos dispositivos no Eris Blockchain foram estabelecidos dois tipos de relacionamentos:

primeiro foi criado o perfil do dispositivo e depois o emparelhamento de dispositivos, conforme ilustra a Figura 3.2.

Figura 3. 2. Configuração de ambiente experimental.



Fonte: (WU, 2018).

Para avaliar o desempenho da autenticação dos dispositivos baseada em Blockchain, foram medidos a memória e o uso da CPU de cada nó do sistema. Esses resultados foram obtidos usando o comando “top” do Linux para obter uma captura de tela em tempo de execução de cada processo Blockchain a cada 10 segundos e calculando o valor médio de 100 imagens consecutivas. O valor médio em porcentagem de sobrecarga de CPU com BeagleBone Black e os nós do Raspberry Pi 3 foram em média 29,55% e 13,35% respectivamente Os resultados experimentais indicam que a CPU e a sobrecarga de memória são aceitáveis, considerando que ocorrem apenas durante a fase de autenticação.

3.2 RESUMO COMPARATIVO DOS PRINCIPAIS TRABALHOS RELACIONADOS

Um resumo comparativo dos trabalhos relacionados com esta pesquisa pode ser visto no Quadro 3.1. Este resumo considera informações como uso de transparência e tipo de Blockchain adotado.

Quadro 3. 1. Tabela geral dos trabalhos relacionados.

Principais Trabalhos Relacionados	Aplicado em Estudo de Caso	Tipo de Blockchain		Arquitetura em Camadas	Aplicação da Transparência (Sistemas Distribuídos)	Objetivo da Proposta do Trabalho
		Público	Privado			
SIMIĆ;SLADIĆ; MILOSAVLJEVIĆ	NÃO	**NÃO**	NÃO	**NÃO**	**NÃO**	Segurança
DORRI	SIM	**NÃO**	SIM	**NÃO**	**NÃO**	Consumo de Energia
YU	**SIM**	SIM	NÃO	SIM	**NÃO**	Taxa de transferência de transações
ALI;DOLUI; ANTONELLI	**SIM**	**NÃO**	**SIM**	**NÃO**	**NÃO**	Sobrecarga de processamento, sobrecarga de tráfego de rede e tempos de processamento de bloco
PINNO; GRÉGIO; DE BONA	**SIM**	**NÃO**	**SIM**	**NÃO**	**NÃO**	Segurança
KOŠŤÁL	**SIM**	**NÃO**	**SIM**	**NÃO**	**NÃO**	Tempo de Upload de configuração
WU	**SIM**	**NÃO**	**SIM**	**NÃO**	**NÃO**	Segurança

Fonte: O autor deste Trabalho.

Os trabalhos apresentados apontam algumas semelhanças quanto ao uso de arquitetura para integrar dispositivos IoT em Blockchain. Ainda na tabela, são apresentados os objetivos centrais das propostas apontadas nos trabalhos. A partir deste resumo, é possível ver que o uso de arquitetura para integrar dispositivos IoT em Blockchain ainda está em aberto, sendo este um dos pontos explorados no desenvolvimento desta pesquisa.

Como pode ser visto no Quadro 3.1, nenhum dos trabalhos apresentados foca na aplicação da transparência para o usuário final em suas arquiteturas, como também, na usabilidade de padrões de projeto no desenvolvimento de arquiteturas em camadas. O

presente trabalho, que será descrito no próximo capítulo, busca contribuir nestes pontos citados.

3.3 CONSIDERAÇÕES FINAIS DO CAPÍTULO

Neste capítulo, foram abordados os trabalhos relacionados com a proposta desta pesquisa. Nestes trabalhos apresentados, foi possível observar a importância da proposta de uma arquitetura para integrar, de forma transparente, dispositivos IoT em Blockchain. Considerando o exposto, o próximo capítulo apresenta a contribuição principal desta dissertação, que é uma arquitetura para transparência de conectividade de serviços Blockchain em sistemas IoT.

4 PROPOSTA DE ARQUITETURA PARA TRANSPARÊNCIA DE CONECTIVIDADE DE SERVIÇOS BLOCKCHAIN EM SISTEMAS IOT

Este capítulo apresenta a principal contribuição deste trabalho, que é uma proposta de arquitetura para transparência de conectividade de serviços Blockchain em Sistemas IoT. Este capítulo está dividido da forma que segue. A seção 4.1 apresenta a metodologia através das camadas que compõem a arquitetura. Em seguida, a Seção 4.2 apresenta a implementação desta arquitetura (protótipo). Por fim, a Seção 4.3 apresentada às considerações finais deste capítulo.

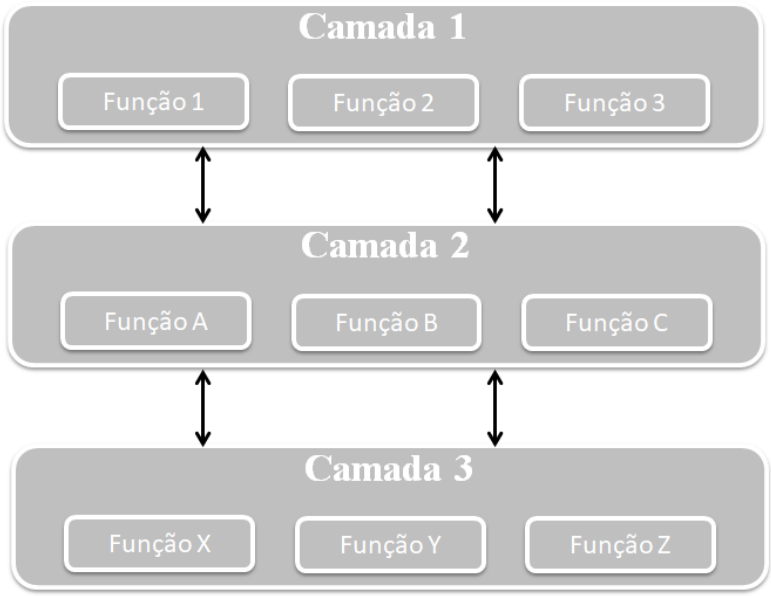
4.1 ARQUITETURA PROPOSTA

A arquitetura para transparência de conectividade de serviços Blockchain em Sistemas IoT proposta nesta dissertação consiste em um Módulo instalado dentro de um Gateway IoT, onde serão disponibilizados serviços para serem consumidos por usuários/dispositivos. O módulo será capaz de receber dados de dispositivos inteligentes que se comunicam com o Gateway IoT e enviá-los para um fornecedor de serviços Blockchain.

A arquitetura proposta está organizada em camadas, onde cada camada desempenha um papel específico. Arquitetura em camadas é um sistema Cliente-Servidor onde as camadas que a compõem são separadas, podendo ser compreendida como um processo de decomposição do sistema (ESTEVÃO; SANDMANN; SANTOS, 2018).

O uso de camadas na arquitetura é um fator importante para promover a independência entre as funcionalidades do sistema e esta independência é responsável por garantir a eficiência, a escalabilidade, a reutilização da estrutura e a facilidade da manutenção (COELHO, 2000), como mostra a Figura 4.1.

Figura 4. 1. Padrão de Arquitetura Aplicada na Proposta.



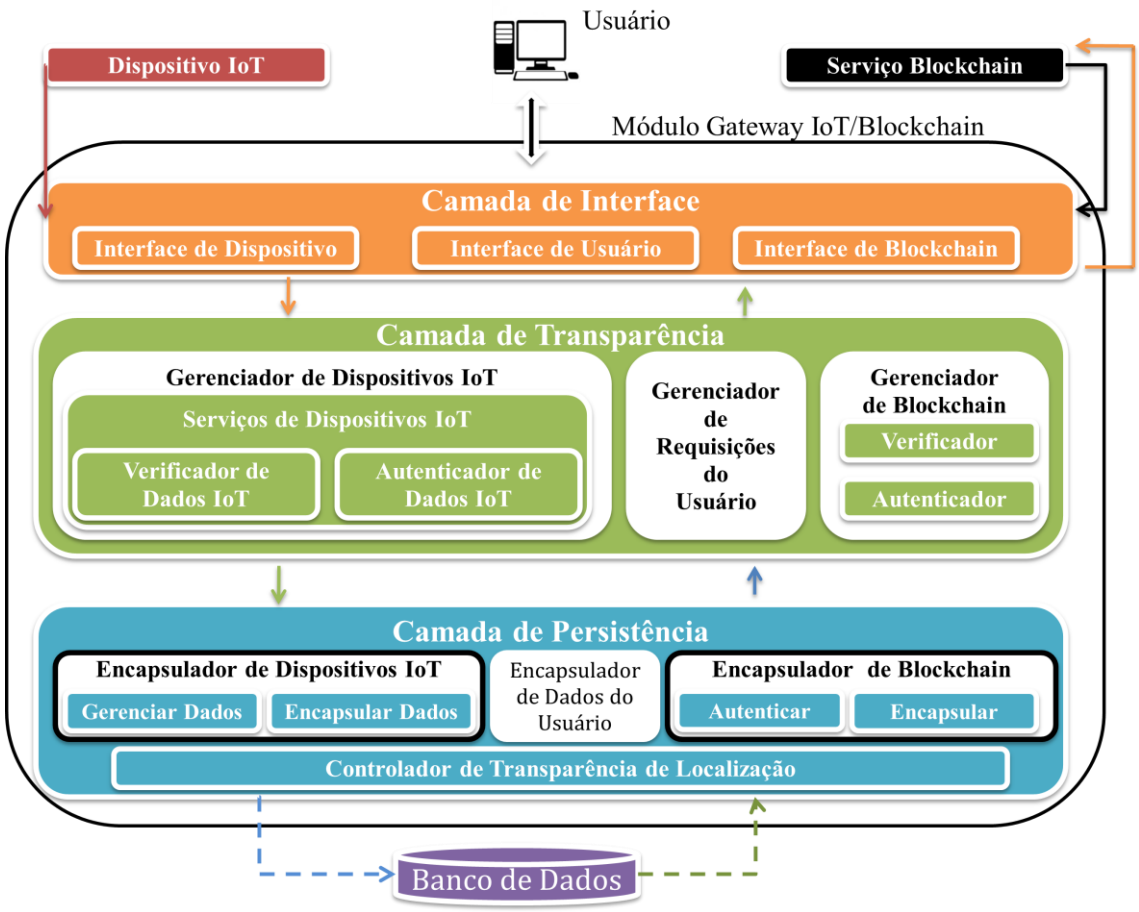
Fonte: Autor deste Trabalho.

O padrão de arquitetura mostrada na Figura 4.1, pode ser aplicado em uma variedade considerável de cenários, independentemente do tipo de plataforma de desenvolvimento ou da linguagem de desenvolvimento utilizada (LAPPALAINEN, 2017). O software em camadas possui a tendência a ter códigos limpos, facilitando seu entendimento e manutenção e, também é possível substituir ou acrescentar camadas na arquitetura sem comprometer as outras camadas (RODRIGUEZ, 2017).

Neste contexto, este trabalho propõe a arquitetura ilustrada na Figura 4.2. Esta arquitetura se fundamenta em três camadas, dividindo suas funcionalidades em: Camada de Interface, responsável pela apresentação dos dados para o usuário final; Camada de Transparência, que especifica as regras de negócio e determina como os dados e as requisições serão utilizadas e, por fim, a Camada de Persistência, que armazena e/ou recupera dados quando solicitado.

Nesta arquitetura, também estão presentes componentes responsáveis pelo funcionamento, segurança e abstração da comunicação entre as camadas.

Figura 4. 2. Arquitetura proposta.



Fonte: Autor deste Trabalho.

4.1.1 CAMADA DE INTERFACE

A camada de Interface tem por objetivo a interação com o usuário, exibindo as informações requisitadas. Nesta camada estão os clientes (usuários, dispositivos e Blockchain) ou aplicações que utilizarão os dados coletados através dos dispositivos IoT. Esses usuários podem ser páginas Web ou qualquer outro serviço habilitado com conexão à internet.

A interface do usuário “user interface” é o frontend onde o usuário final interage e realiza todas as ações do sistema, como cadastro e pesquisa dos dispositivos IoT, geração de relatórios e o cadastro dos dispositivos IoT em redes Blockchain, como também toda operação envolvendo a segurança e o acesso do usuário no sistema. As principais responsabilidades dos componentes desta camada são: apresentar uma interface inicial, para o acesso do usuário; receber requisições dos usuários; receber informações dos dispositivos IoT cadastrados no sistema, tais como suas características

(Id) e dados coletados (Value) e apresentar informações e relatórios aos usuários de forma personalizada.

O detalhamento de cada componente do modelo arquitetural da camada de interface do usuário é descrito no Quadro 4.1.

Quadro 4. 1. Detalhamento da Camada de Interface.

1. Componente: Interface de Dispositivo. 1.1 Descrição: Através desta interface o usuário poderá cadastrar dispositivos IoT ao sistema proposto, como também, requisitar informações sobre os dispositivos e seus status. 2. Componente: Interface de Usuário. 2.1 Descrição: Responsável por gerenciar o acesso do usuário no sistema. Este gerenciamento é realizado através do acrônimo da expressão “Create, Read, Update e Delete – CRUD”. 3. Componente: Interface de Blockchain. 3.1 Descrição: Esta interface é responsável por gerenciar a conexão dos dispositivos IoT com o Blockchain.

Fonte: Autor deste Trabalho.

4.1.2 CAMADA DE TRANSPARÊNCIA

A camada de Transparência é responsável por gerenciar as funcionalidades necessárias para conexões dos dispositivos IoT com os serviços de Blockchain, de forma que o usuário não tenha a percepção da complexidade e da estrutura dos métodos aplicados dentro desta camada, resultando em uma transparência para o usuário final.

Esta é a camada intermediária, onde são interpretados e processados os comandos realizados na camada de Interface, gerenciando as regras de negócios, como também realizando validações e autenticações dos dados de entrada, antes da camada de Transparência se comunicar com os métodos da camada de Persistência.

Para abstrair o conjunto de objetos/métodos da camada de transparência para o controle dos dispositivos IoT foi criada uma classe denominada “DispositivosController”. Dentro desta classe foi criados um conjunto de procedimentos (métodos) responsáveis por controlar o fluxo de requisições referente tanto ao acesso dos dispositivos IoT ao sistema proposto como também às entradas e saídas dos dados coletados.

Com o objetivo de listar todos os dispositivos IoT cadastrados no sistema proposto, foi criado dentro desta classe um método denominado "listarDispositivosAction()", como consta na linha 5 da Figura 4.3. Para a realização da listagem dos dispositivos IoT é necessário instanciar (compartilhar) uma classe da camada de persistência denominada "DispositivoModel()" criando um objeto ($o_Dispositivo), linha 7.

Este objeto está apontando para um método da classe "DispositivoModel()" denominada "_list()" também da camada de persistência, linha 10. Sendo assim, é possível listar e imprimir todos os dispositivos armazenados no banco de dados para o usuário final.

Figura 4. 3. Estrutura de Objetos da Camada de Transparência.

```
1   <?php
2   class DispositivosController{
3
4       // Criação do método
5       public function listarDispositivosAction(){
6
7       $o_Dispositivo = new DispositivoModel();
8
9           //Listando os contatos cadastrados
10      $v_ Dispositivo = $o_ Dispositivo ->_list();
11
12          //definindo qual o arquivo HTML que será usado para
13           $o_view = new View('views/listarDispositivo.html');
14
15          //Passando os dados do Dispositivo para a View
16      $o_view->setParams(array('v_ Dispositivos' => $v_ Dispositivo));
17
18          //Imprimindo código HTML
19      $o_view->showContents();
20      }
21  }
```

Fonte: Autor deste Trabalho.

As principais responsabilidades dos métodos dessa camada são: processar a informação provida pelo usuário, através dos métodos da classe da camada de transparência; validar e autenticar todas as requisições; e criptografar dados válidos antes de enviar para a camada de persistência.

O detalhamento de cada componente do modelo arquitetural da camada de transparência é descrito no Quadro 4.2 a seguir.

Quadro 4. 2. Detalhamento da Camada de Transparência.

1.Componente: Gerenciador de Dispositivos IoT. **1.1Descrição:** Responsável por receber as requisições da camada de interface do componente "Interface do Dispositivo IoT", realizando todo o processo de controle, fluxo e validações dos dados recebidos. **1.2Serviços:** **1.2.1 Serviços de Dispositivos IoT** - Responsável por receber as requisições dos dispositivos IoT. Este serviço é composto por dois principais subserviços; **1.2.1.1 Verificador de IoT** - Componente responsável por verificar se todos os dados dos dispositivos IoT foram completamente capturados dentro do método. **1.2.1.2 Autenticador de IoT** - Componente responsável por autenticar os dados dos dispositivos IoT, conforme protocolo estabelecido dentro do sistema; **2. Componente:** Gerenciador de Requisições do Usuário. **2.1 Descrição:** Este gerenciador recebe as requisições *"Create, Read, Update e Delete – CRUD"* da camada de interface do usuário, realizando o processo de filtragens e validações. Após aplicação desses processos, os dados são encaminhados para a "Camada de Persistência", onde serão autenticados. **3. Componente:** Gerenciador de *Blockchain*. **3.1Descrição:** Este gerenciador é responsável por receber as requisições da camada de interface e tem como objetivo validar e filtrar as requisições de cadastros e conexões dos dispositivos IoT no *Blockchain,* sendo composto por dois serviços principais. **3.2Serviços:** **3.2.1 Verificador** - Componente responsável por verificar se os dados dos dispositivos IoT para cadastro no serviço *Blockchain* foram capturados por completo. **3.2.1 Autenticador** - Componente é responsável por autenticar os dados para o cadastro dos dispositivos IoT no serviço *Blockchain*.

Fonte: Autor deste Trabalho.

O conceito de transparências aplicado na arquitetura proposta é responsável por gerenciar e manipular os arquivos e métodos das classes onde estão inseridas, com o objetivo de ocultar a localização dos recursos da arquitetura, executando os processos de leitura e escrita de arquivos de forma remota ou local, sem que haja diferença no processo, como também, visando ocultar a localização física dos arquivos.

4.1.2.1 Transparência de Acesso

A transparência de acesso objetiva ocultar para o usuário o modo no qual os recursos existentes dentro do sistema podem ser acessados (BOMBARDELLI, 2017). Esta transparência é um atributo que está relacionado diretamente com o formato de desenvolvimento da arquitetura proposta, e este atributo está configurado de modo que todos os recursos dos dispositivos IoT possam ser acessados, ocultando as diferenças entre as funcionalidades e possibilitando transparência para o usuário.

Diante deste contexto, a comunicação do usuário com o sistema está relacionada exclusivamente com a camada de interface do usuário, onde o usuário fará as requisições de serviços para dentro do sistema. Contudo, a abstração aplicada ocultará o conhecimento das demais camadas distribuídas e, desta forma, os objetos e dados gerados serão acessados de forma local ou remota usando operações idênticas.

4.1.2.2 Transparência de Localização

A transparência de localização é aplicada para ocultar do usuário o lugar onde os recursos ou dados do sistema estão localizados (COULOURIS, 2013).

A terceira camada da arquitetura proposta (Camada de Persistência) é constituída por um conjunto de classes e métodos responsáveis em prover a criação, remoção, alteração e recuperação dos dados nos sistemas de gerenciamento de bancos de dados (SGBD).

As classes e métodos são componentes do conceito de orientação a objetos que encapsulam a lógica de negócio para acessar uma fonte de dados (RAMOS, 2004), tendo por objetivo centralizar as funcionalidades e desacoplar a infraestrutura da arquitetura do sistema proposto da tecnologia do sistema de gerenciamento de bancos de dados, conforme Figura 4.4.

Figura 4. 4. Arquitetura em três camadas desacopladas do banco de dados.

Fonte: Autor deste Trabalho.

A camada de persistência fornece à camada de transparência a capacidade de acessar as funcionalidades e encaminhar os dados para a camada de interface do usuário.

A vantagem deste desacoplamento entre a camada de persistência do sistema de gerenciamento de banco de dados (SGBD) é dificultar uma comunicação direta. Sendo assim, é possível serem realizadas modificações no SGBD "A" para outro SGBD "B" sem afetar a arquitetura, a camada de persistência e principalmente a percepção do usuário em relação à localização física do SGBD.

4.1.3 CAMADA DE PERSISTÊNCIA

A camada de Persistência é responsável por gerenciar a manipulação dos dados dentro do sistema através da linguagem de manipulação de dados (ou DML, Data Manipulation Language). O DML é um conjunto de comandos dentro da linguagem de consulta estruturada (ou SQL, Structured Query Language), utilizado para recuperar, incluir, remover e modificar informações em bancos de dados (ALI; SHIBGHATULLAH, 2016).

Para realizar a manipulação dos dados a camada de persistência se conecta a um sistema de armazenamento de dados, onde através de um gerenciador de sistema de

banco de dados é possível aplicar o DML. Estes gerenciadores de banco de dados podem ser MySql; Oracle; PostgreSql; MongoDB, dentre outros (REPORT, 2020).

O detalhamento de cada componente do modelo arquitetural da camada de persistência é descrito no Quadro 4.3.

Quadro 4. 3. Detalhamento da Camada de Persistência.

1 Componente: Encapsulador de Dispositivo IoT.
1.1Descrição: Este gerenciador refere-se ao agrupamento de métodos necessários para aplicar as restrições diretas aos dados deste componente, onde é composto por dois serviços principais.
1.2 Serviços:
1.2.1 Gerenciar dados: Este componente é responsável por gerenciar o método para criptografar os dados do dispositivo IoT, como também, verificar a existência do dispositivo no banco de dados;
1.2.2 Encapsular dados: Responsável por ocultar o estado (comportamento) dos métodos da classe, evitando o acesso direto pelo usuário ou pelas demais camadas da arquitetura aos dados dos dispositivos IoT.
2 Componente: Encapsulador de Dados do Usuário.
2.1 Descrição: Este gerenciador tem por objetivo criptografar e gerenciar de forma oculta toda a estrutura de persistência dos dados dos usuários no banco de dados.
3 Componente: Encapsulador de Blockchain.
3.1 Descrição: Este gerenciador é composto por dois serviços.
3.2 Serviços:
3.2.1 Autenticar - Este serviço tem por objetivo confirmara origem dos dados da requisição do serviço *Blockchain*, estabelecendo a procedência do objeto selecionado;
3.2.2 Encapsular - É responsável pela realização da abstração, ocultação da visibilidade dos dados na camada de persistência do serviço *Blockchain* e realizar a criptografia dos dados.
4 Componente: Controlador de Transparência de Localização.
4.1 Descrição: Este gerenciador é responsável por configurar e gerenciar a conexão e o acesso ao banco de dados.

Fonte: Autor deste Trabalho.

4.2 IMPLEMENTAÇÃO

A arquitetura de três camadas fornece os protocolos necessários para conexão entre o cliente (camada de Interface) e as demais camadas existentes. Isto significa que o usuário não precisa instalar ou configurar a camada de Interface para ter acesso ao sistema, necessita apenas que tenha um navegador da Web disponível com conexão à internet.

Para o desenvolvimento da arquitetura para transparência de conectividade de serviços Blockchain em sistemas IoT foi aplicado o padrão cliente Web thin, que envolve padrões usados para sistemas com base na internet e sua principal característica é a configuração mínima do usuário, ou seja, o usuário necessita apenas de um navegador Web para acessar o sistema e toda lógica de negócio será executado dentro do servidor Web. Seus principais componentes são: navegador de cliente; servidor Web e conexão HTTP (DE LEMOS, 2013).

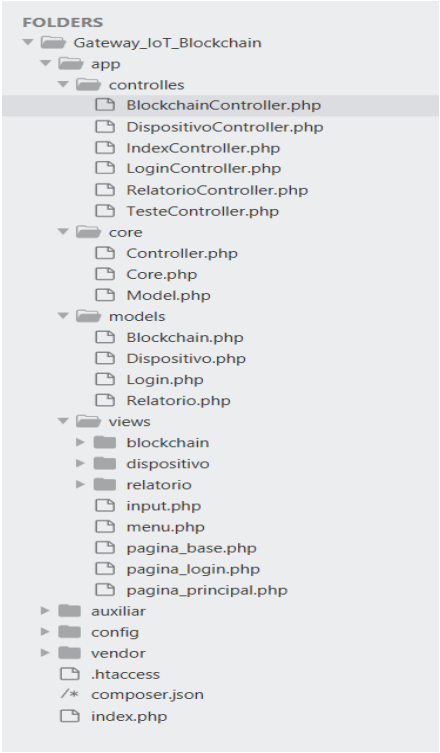
Para implementar o modelo arquitetural, foi utilizado como linguagem de desenvolvimento o PHP versão 7 (sete), visto a flexibilidade de sua utilização, tanto no lado do cliente “frontend”, quanto do lado do servidor “backend”. Através da robustez desta linguagem de desenvolvimento, foi possível criar as camadas que compõem a arquitetura proposta, como também os módulos e componentes de execução. Sendo uma linguagem opensource aplicada a uso geral na Web (NIEDERAUER, 2017), a linguagem de desenvolvimento PHP (Hypertext Preprocessor) é capaz de suportar grandes quantidades de dados, obtendo um bom desempenho na execução de muitas funções e consumindo muitos recursos ao mesmo tempo, sem comprometer o desempenho e a velocidade do servidor (DALL'OGLIO, 2018).

A compatibilidade com os principais banco de dados comerciais (MySQL, SQLite, Firebird, Interbase e Oracle) é uma grande vantagem na utilização da linguagem PHP em projetos Web (ROTERMEL; SOMMARIVA, 2016), motivo pelo qual adotamos esta linguagem de programação como padrão para o desenvolvimento para arquitetura proposta.

A estrutura de hierarquia dos diretórios e arquivos que caracteriza as três camadas da arquitetura proposta está descrita conforme a Figura 4.5. Esta estrutura está baseada no conceito de MVC; um acrônimo de “Model – View – Controller”. O MVC é

uma forma de estruturar os diretórios permitindo divisão das funcionalidades do sistema em camadas.

Figura 4. 5. Estrutura de Pacote da Arquitetura Proposta.



Fonte: Autor deste Trabalho.

Dentro do diretório da pasta raiz (Gateway_Iot_Blockchain) estão contidos os principais arquivos de configurações e as respectivas camadas para o funcionamento do sistema proposto. Cada camada tem sua estrutura administrativa de comunicação e de gerenciamento. O arquivo index.php é responsável por iniciar a aplicação do sistema no navegador Web. Por sua vez, a pasta config contém as configurações de conexão com o banco de dados que é utilizado para o armazenamento das informações sobre os dispositivos IoT, dados coletados pelos dispositivos e informações sobre a conectividade com o serviço Blockchain. Dentro da pasta auxiliar estão os arquivos de imagens utilizadas no sistema, como também, os arquivos de javascript (aplicação frontend) e arquivos para de estilo css (aplicação frontend).

A pasta Views é um pacote que contém arquivos dinâmicos criados para interagir com o usuário, realizando a visualização da informação de acordo com o tipo de requisições solicitadas pelo usuário final. Esta pasta representa a camada de interface.

A camada de Transparência é representada pela pasta Controller. Esta pasta foi desenvolvida com objetivo de armazenar todos os métodos (procedimentos) da aplicação, com objetivo de receber e processar as requisições da camada views (Camada de Interface). Cada tipo de método contido dentro da pasta Controller possui uma função específica que realizará uma determinada ação.

Por fim, esta camada é responsável por fazer o intermédio entre a Camada de Persistência (pasta models) e a Camada de Interface (pasta views), contendo todos os métodos (procedimentos) necessários para receber, processar, autenticar e validar as requisições do usuário e/ou dispositivos.

A pasta Models representa a Camada de Persistência e, esta pasta contém todos os métodos (procedimentos) para o recebimento das requisições da pasta Controller (Camada de Transparência). Através dos métodos da pasta Models são utilizados o conjunto de comandos SQL para gerenciar a persistência dos dados no banco de dados.

Considerando a necessidade de acessar ao sistema por diversos tipos de aparelhos eletrônicos, foi utilizado o framework Bootstrap, com o objetivo de otimizar a criação e a codificação de uma estrutura única e consistente de frontend.

O Bootstrap é um framework utilizado para o desenvolvimento Web de forma responsiva, adaptando o conteúdo ao tamanho das páginas (layout) de acordo com as suas dimensões (HESTERBERG, 2011). O Bootstrap possui componentes “plugins”, desenvolvidos em JavaScript que auxiliam na formatação do design (CHERNICK, 2011), podendo ser usado com qualquer IDE e qualquer linguagem de desenvolvimento do lado servidor, como ASP.NET;, PHP e RUBY.

Para desenvolvimento do layout da camada de interface foram criados arquivos para o desenvolvimento dos formulários de inserção de cadastro dos dispositivos IoT ao sistema proposto, como também, para edição dos dados, para os serviços de usuários e de serviço Blockchain.

Esses arquivos foram adicionados dentro do arquivo Bootstrap contendo as seguintes subpastas, conforme pode ser visto na Figura 4.6. O arquivo CSS – Cascading Style Sheet na versão 3 (três) na linha 3, responsável pelo o estilo da página; JAVASCRIPT - usado para a implementação de itens de alto nível como: animações, mapas e gráficos, sendo aplicado como “Front End”, contudo, na linha 13 e na linha 18 estão os tipos de fontes usadas para configurar textos.

Figura 4. 6. Estrutura de Arquivos do Bootstrap.

```
<?php
bootstrap/
├── css/
│   ├── bootstrap.css
│   ├── bootstrap.css.map
│   ├── bootstrap.min.css
│   ├── bootstrap-theme.css
│   ├── bootstrap-theme.css.map
│   └── bootstrap-theme.min.css
├── js/
│   ├── bootstrap.js
│   └── bootstrap.min.js
└── fonts/
    ├── glyphicons-halflings-regular.eot
    ├── glyphicons-halflings-regular.svg
    ├── glyphicons-halflings-regular.ttf
    ├── glyphicons-halflings-regular.woff
    └── glyphicons-halflings-regular.woff2
```

Fonte: Autor deste Trabalho.

4.2.1 IMPLEMENTAÇÃO DAS TRANSPARÊNCIAS

4.2.1.1 Transparência de Acesso

Na utilização do servidor Web Apache (TEIXEIRA, 2018), foi instalado dentro do diretório File Transfer Protocol - FTP um arquivo denominado .htaccess, permitindo o gerenciamento descentralizado das configurações. Este arquivo direciona e controla as páginas requisitadas pelo usuário e também configura a regravação da URL, desenvolvendo uma URL amigável.

As configurações do arquivo (.htaccess), utilizadas na configuração da URL, são mostradas conforme a Figura 4.7.

Figura 4. 7. Implementação do script *htaccess.

```
<?php
* .htaccess
RewriteEngine On
RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-d
RewriteRule ^(.*)$ index.php/$1 [L]

```

Fonte: Autor deste Trabalho.

Na configuração do arquivo .htaccess apresentado na Figura 4.7, foi inicialmente ativado o engine para a reescrita de URL, como consta na linha 3. Caso a requisição solicitada pelo usuário não seja um arquivo ou um diretório, a linha 4 e 5 realizará a negação de redirecionamento da página solicitada. Por fim, na linha 6, o script RewriteRule verifica o que foi solicitado pela URL e redireciona para a página index.php; no entanto, esse redirecionamento apenas será executado quando as regras da linha 4 e 5 estiverem satisfeitas.

Nesta configuração de acesso ao sistema, à função "autoload" irá carregar a classe do "Controller - Camada de Transparência" de forma automática, mostrando para o usuário o método contido na classe, ao invés do endereço da página requisitada, como mostra a Figura 4.8, linha 5. Desta forma, todos os endereços das páginas requisitadas pelo usuário serão ocultados e o que será mostrado em substituição serão os endereços de localização dos métodos das classes do Controller (Camada de Transparência).

Figura 4. 8. Implementação do script autoload.

```
1   <?php
2
3   // autoload.php @generated by Composer
4
5   require_once __DIR__ . '/composer/autoload_real.php';
6
7   return ComposerAutoloaderInit6fffeb2901d47a6fe38b2179fe3bd90c::getLoader();
```

Fonte: Autor deste Trabalho.

Na Figura 4.5, o diretório (app/core) é composto por três arquivos. O arquivo core.php é responsável por tratar a URL decidindo qual controlador e qual método deve ser executado. O arquivo controller.php é responsável por chamar a pasta model, view e controller através do método load, como consta na linha 7 até a linha 10 do Quadro A na Figura 4.9, onde são herdados para as classes no diretório App/controllers.

O arquivo app/core/controller.php fornece para as classes filhas (as classes em App/controllers/*) os métodos responsáveis para instanciar um model de forma dinâmica, exibindo uma determinada view passando para a mesma os dados em um array que podem ser acessado pela variável ($viewData = array()). Em casos de métodos ou controladores não encontrados (informado pelo usuário/dispositivo IoT/Serviço Blockchain), é chamado o método (Index) da class IndexController.

Por fim, o arquivo Model.php armazena a conexão com o banco de dados através de um método construtor na linha 7, onde instancia um objeto da classe “PDO”, ilustrado na Figura 4.9 no Quadro B.

Figura 4. 9. Diretórios de Arquivos (Camada de Interface).

Quadro A

```
1   <?php
2
3   namespace app\core;
4
5   class Controller{
6
7       public function load($viewName, $viewData=array()){
8           extract($viewData);
9           include "app/views/" . $viewName . ".php";
10      }
11
12  }
```

Quadro B

```
1   <?php
2   namespace app\core;
3
4   abstract class Model{
5       protected $database;
6
7       public function __construct() {
8           $this->database = new \PDO("mysql:dbname=".BANCO.";host=".SERVIDOR,USUARIO,SENHA);
9
10      }
11  }
```

Fonte: Autor deste Trabalho.

O diretório App/core/Core.php tem objetivo de obter a URL digitada no Browser pelo usuário e dividi-la por barra “/” obtendo assim um array como mostrado na Figura 4.10, da linha 19 até a linha 45, do método “Verificador”. O primeiro índice deste array contém o controller que será instanciado (linha 27); o segundo índice contém o método chamado de controller (instanciado na linha 28) e o restante são considerados como parâmetros do método (linha 39).

Figura 4. 10. Script de Tratamento da URL.

```php
<?php


    class Core{

        private $controller;
        private $metodo;
        private $parametro = array();

        public function __construct(){ $this->Verificador(); }

        public function run(){

            $controller = $this->getController();
            $control = new $controller();
            call_user_func_array(array($control, $this->getMetodo()), $this->getParametro());
        }

        public function Verificador(){
            $url =  explode("index.php", $_SERVER["PHP_SELF"]);
            $novo_url = end($url);

            if($novo_url != ""):
                $url_novo = explode("/", $novo_url);

                //Separa o Controller
                array_shift($url_novo);// excluir o ultimo valor do Array
                $this->controller = ucfirst($url_novo[0] . "Controller");

                //Separa o Metodo
                array_shift($url_novo);// excluir o ultimo valor do Array
                if(isset($url_novo[0])):
                    $this->metodo = $url_novo[0];
                endif;

                //Separa o Metodo
                array_shift($url_novo);// excluir o ultimo valor do Array
                if(isset($url_novo[0])):
                    $this->parametro = array_filter($url_novo);
                endif;

            else:
                $this->controller = ucfirst(CONTROLLER_PADRAO) . "Controller";
            endif;
        }//metodo


        public function getController(){
            if(class_exists(NAMESPACE_CONTROLLER . $this->controller)):
                return NAMESPACE_CONTROLLER . $this->controller;
            else:
                return NAMESPACE_CONTROLLER . ucfirst(CONTROLLER_PADRAO) . "Controller";
            endif;
        }

        public function getMetodo(){

            if(method_exists(NAMESPACE_CONTROLLER . $this->controller, $this->metodo)):
                return $this->metodo;
            endif;

            return METODO_PADRAO;

        }

        public function getParametro(){
            return $this->parametro;
        }

    }//class
```

Fonte: Autor deste Trabalho.

4.2.1.2 Transparência de Localização

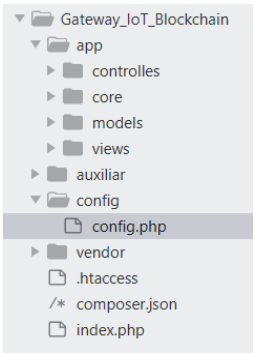
A utilização da transparência de localização consiste na forma de comunicação entre a própria arquitetura proposta com o sistema de gerenciamento de banco de dados - SGBD.

A última camada da arquitetura proposta, a camada de persistência, possui métodos com diretrizes para realizar a comunicação com diferentes tipos de SGBD. Logo, a camada de persistência isola toda a comunicação com o Sistema de Gerenciamento de Banco de Dados – SGBD do restante das camadas. Este isolamento garante que se houver qualquer modificação no tipo do SGBD, incluindo a forma de comunicação do SGBD, a camada de Interface não será afetada e o usuário não terá conhecimento da localização do armazenamento dos dados.

A Figura 4.11, Quadro A, apresenta a pasta “config”, responsável por configurar o SGBD que irá se conectar com a arquitetura proposta. Esta pasta foi criada dentro da pasta raiz do sistema app, separado das camadas que compõem a arquitetura, para isolar a forma de conexão.

Figura 4. 11. Arquivo config /Conexão com SGBD.

A



B

```
1   <?php
2   //configuração para o database
3   define("SERVIDOR", "localhost");
4   define("BANCO", "off_chain");
5   define("USUARIO", "root");
6   define("SENHA", "");
7
8   define('CONTROLLER_PADRAO', 'index');
9   define('METODO_PADRAO', 'index');
10  define('NAMESPACE_CONTROLLER', 'app\\controlles\\');
11  define('URL_BASE', 'http://127.0.0.1/Gateway_IoT_Blockchain/');
```

Fonte: Autor deste Trabalho.

Em relação ao Quadro B da Figura 4.11, o script ilustrado na linha 3 é informar o nome do servidor que se conectará com sistema, por sua vez, a linha 4 informa o nome do banco de dados criado para armazenamento dos dados. As linhas 5 e 6 informam o login e a senha de acesso. Por fim, as linhas 8 até 11 configuram a URL que será acessada quando o arquivo index.php for chamado.

4.2.2 IMPLEMENTAÇÃO DO BLOCKCHAIN

Com o objetivo de ilustrar a arquitetura proposta, um Blockchain foi desenvolvido exclusivamente para adaptar-se à estrutura desenvolvida para esta arquitetura. Para o desenvolvimento do Blockchain foi utilizado a linguagem de desenvolvimento PHP, contendo todas as características e padrões que validam uma rede Blockchain. Primeiramente foram definidos a estrutura dos blocos e o tipo do arquivo que irá armazenar o Blockchain, o JSON (modelo para armazenamento e transmissão de informações no formato texto).

Os atributos criados para cada bloco contêm como características o índice, hash, timestamp, a prova de trabalho e o conteúdo. O Índice é um valor numérico inteiro, único e incremental, responsável por identificar cada bloco como único dentro da cadeia de blocos. O primeiro valor do hash do bloco é criado com um valor arbitrário. No entanto, os valores que irão compor o hash do bloco posterior serão calculados utilizando uma função específica, e esta função está apresentada na Figura 4.12. A linha 2 da Figura 4.12 retorna o valor calculado do hash do bloco anterior contendo o hash do bloco, o tempo, à hora e o índice.

Figura 4. 12. Função Hash.

```
1   public function calculateHash(){
2       return hash("sha256", $this->index.$this->previousHash.$this->timestamp.((string)$this->data).$this->nonce);
3   }
```

Fonte: Autor deste Trabalho.

O Timestamp tem por objetivo especificar a data e o horário de criação de cada bloco. A prova de trabalho pode ser compreendida como o resultado da solução de um determinado problema criptografado e complexo utilizado para autenticar a transação pelos demais nós existentes na rede. Por fim, o conteúdo é um atributo que especifica os dados recebidos dos dispositivos IoT.

O Blockchain foi desenvolvido em três etapas. Como descrito na Figura 4.13, a primeira etapa foi o desenvolvimento da classe block “classBlock”, onde um método construtor é invocado automaticamente toda vez que um objeto ou bloco estiver sendo criado dentro da cadeia de blocos. O método construtor (linha 6 à linha 14) tem como objetivo receber por parâmetro os valores referentes ao horário de criação de cada

bloco, a data e o índice e repassar esses dados para o método “calculateHash()”, onde será instanciado um hash para um novo bloco criado.

Figura 4. 13. Classe Block.

```php
<?php
class Block
{
    public $nonce;

    public function __construct($index, $timestamp, $data, $previousHash = null)
    {
        $this->index = $index;
        $this->timestamp = $timestamp;
        $this->data = $data;
        $this->previousHash = $previousHash;
        $this->hash = $this->calculateHash();
        $this->nonce = 0;
    }

    public function calculateHash()
    {
        return hash("sha256", $this->index.$this->previousHash.$this->timestamp.((string)$this->data).$this->nonce);
    }
}
```

Fonte: Autor deste Trabalho.

A segunda etapa foi o desenvolvimento da visualização da cadeia de blocos para o usuário final, onde cada bloco é criado no momento do envio dos dados pelos dispositivos IoT. Esta etapa está descrita na Figura 4.14, mais especificamente na linha 26.

Figura 4. 14. Impressão da Cadeia de Bloco.

```php

<?php

require_once('../blockchain.php');

$con = mysqli_connect("localhost","root","","sige");


    $result = mysqli_query($con,$sql);

        $i = 0;

        $testCoin = new BlockChain();

            while ($dados = mysqli_fetch_assoc($result)){

                $i = $i + 1;

                //echo "mining block ".$i."...\n";
                $testCoin->push(new Block($i, strtotime("now"), $dados['id']));
                //echo "<br/><br/>";

            }

        echo "<pre>";
            echo json_encode($testCoin, JSON_PRETTY_PRINT);
        echo "</pre>";
```

Fonte: Autor deste Trabalho.

A terceira etapa foi o desenvolvimento da classe blockchain. Esta classe possui todos os métodos para a criação da cadeia de blocos e toda a estrutura de validação, gerenciamento e configuração dos blocos, conforme descrito na Figura 4.15.

Figura 4. 15. Classe Blockchain.

```php
<?php
require_once("../block.php");

/**
 * Uma classe de blockchain simples com prova de trabalho (mineração).
 */
class BlockChain
{
    /**
     * Instantiates a new Blockchain.
     */
    public function __construct()
    {
        $this->chain = [$this->createGenesisBlock()];
        $this->difficulty = 4;
    }

    /**
     * Instancia um novo Blockchain.
     */
    private function createGenesisBlock()
    {
        return new Block(0, strtotime("2017-01-01"), "Genesis Block");
    }

    /**
     * Obtém o último bloco da cadeia.
     */
    public function getLastBlock()
    {
        return $this->chain[count($this->chain)-1];
    }

    /**
     * Empurra um novo bloco para a corrente.
     */
    public function push($block)
    {
        $block->previousHash = $this->getLastBlock()->hash;
        $this->mine($block);
        array_push($this->chain, $block);
    }
    /**
     * Minas um bloco.
     */
    public function mine($block)
    {
        while (substr($block->hash, 0, $this->difficulty) !== str_repeat("0", $this->difficulty)) {
            $block->nonce++;
            $block->hash = $block->calculateHash();
        }

        //echo "Block mined: ".$block->hash."\n";
    }

    /**
     * Valida a integridade do blockchain. Verdadeiro se o blockchain for válido, falso caso contrário
     */
    public function isValid()
    {
        for ($i = 1; $i < count($this->chain); $i++) {
            $currentBlock = $this->chain[$i];
            $previousBlock = $this->chain[$i-1];

            if ($currentBlock->hash != $currentBlock->calculateHash()) {
                return false;
            }

            if ($currentBlock->previousHash != $previousBlock->hash) {
                return false;
            }
        }

        return true;
    }
}
```

Fonte: Autor deste Trabalho

Na classe Blockchain está o método responsável pela criação do bloco gênesis (linhas 21 a 24) chamado “createGenesisBlock()”, onde será criado o primeiro bloco da cadeia. Nas linhas 29 a 32 está o método “getLastBlock()”, responsável por obter o último bloco gerado. Nas linhas 37 a 42 se encontra o método “push ($block)” que recebe o último bloco e o coloca na cadeia de blocos. O método “mine ($block)”, nas linhas 47 a 55, realiza a mineração de cada bloco antes de encaminhar para a cadeia de blocos, realizando a validação dos blocos. Por fim, nas linhas 60 a 76, ocorre a validação da cadeia de blocos.

4.3 CONSIDERAÇÕES FINAIS DO CAPÍTULO

Neste capítulo foi apresentada a principal contribuição desta dissertação, que é uma arquitetura para transparência de conectividade de serviços Blockchain em sistemas IoT. De forma resumida, foi detalhado um modelo arquitetural composto por três camadas. A primeira camada, views, representa a Camada de Interface, que é responsável pela interação do usuário com a Camada de Transparência. A segunda camada, controller, representa a Camada de Transparência, responsável por gerenciar todos os métodos da aplicação dos usuários, dos dispositivos IoT e dos serviços Blockchain. A terceira camada, models, representado a Camada de Persistência, responsável por interagir com o banco de dados e persistir os dados. Finalmente, foi desenvolvido um serviço Blockchain para ilustrar e avaliar o comportamento da arquitetura proposta neste trabalho.

5 AVALIAÇÃO

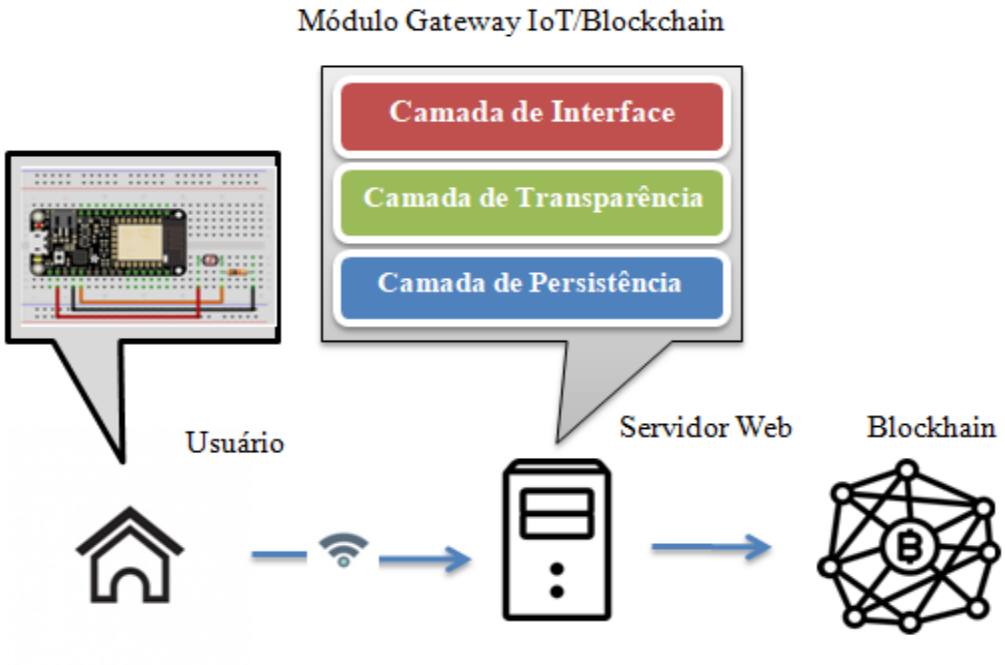
Neste capítulo, são apresentados dois estudos de casos que objetivam ilustrar e avaliar a utilização do modelo arquitetural proposto nesta pesquisa.

5.1 ESTUDO DE CASO 1: Casa Inteligente

Esta seção descreve como o modelo arquitetural proposto nesta dissertação pode ser inserido em um cenário real com o intuito de ilustrar e avaliar o uso da referida arquitetura. Este estudo de caso tem por objetivo avaliar o comportamento estrutural do sistema como um todo, onde os resultados serão obtidos através da execução completa do sistema.

Esta execução completa do sistema consiste desde a captação dos dados coletados do ambiente através dos dispositivos IoT até ao sistema proposto, onde esses dados serão processados e enviados ao serviço Blockchain,finalizando assim, na criação da cadeia de blocos (Blockchain), como ilustrado na Figura5.1.

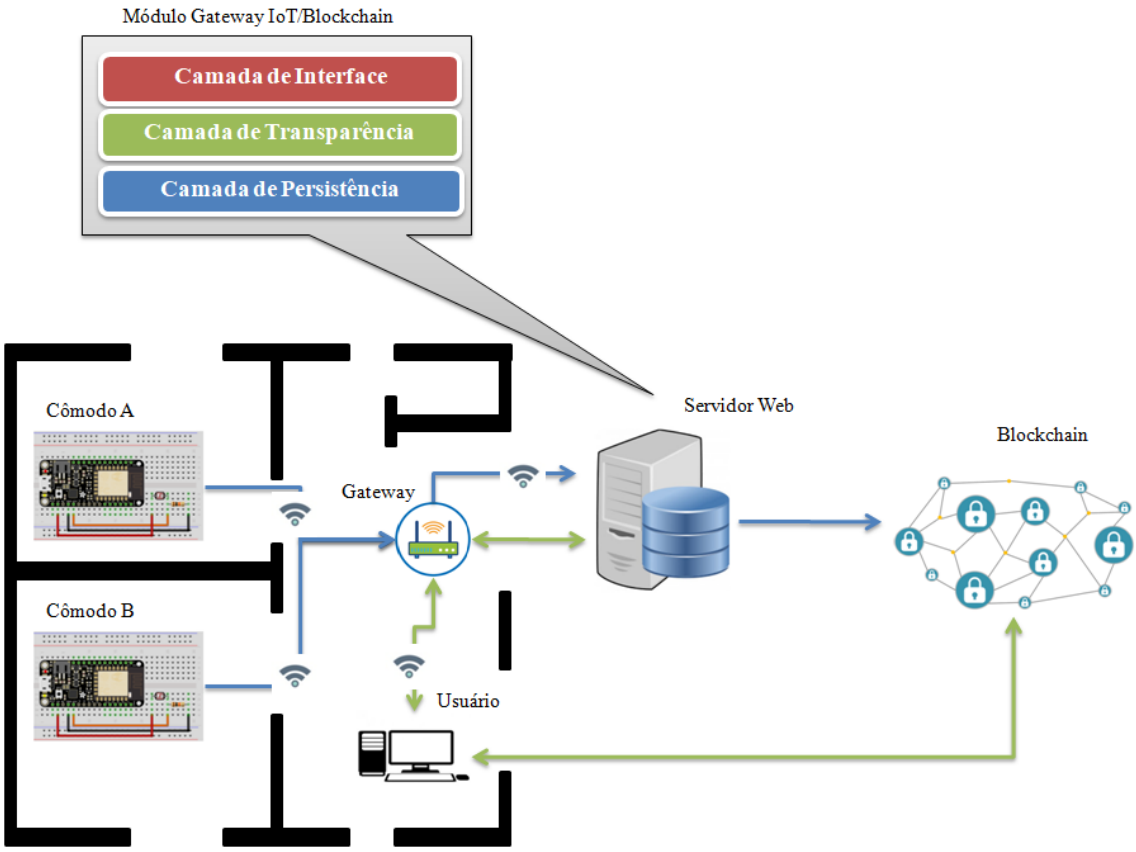
Figura 5. 1. Ciclo completo de execução do sistema proposto.



Fonte: Autor deste Trabalho.

O cenário escolhido foi inspirado em uma casa inteligente, pois esta é uma atual e relevante área no contexto da Internet das Coisas atualmente. A Figura 5.2 ilustra o fluxo do comportamento do cenário.

Figura 5. 2. Estudo de Caso 1.



Fonte: Autor deste Trabalho.

Os dados coletados através dos dispositivos IoT que se encontram nos cômodos A e B da casa inteligente são enviados para o servidor Web através do gateway. No servidor Web se encontram o sistema proposto como também a pasta com os arquivos contendo scripts de desenvolvimento do serviço Blockchain.

Dentro do sistema proposto, os dados enviados pelos dispositivos IoT são validados, como também, se verifica o cadastro dos dispositivos IoT no serviço Blockchain disponível. Após estes protocolos de validação e verificação, os dados são persistidos no banco de dados e enviados também para o Blockchain de acordo com a permissão de cada dispositivo.

O usuário poderá ter acesso aos dados dos dispositivos IoT através do sistema proposto, como também através da Interface de Programação de Aplicações - API do Blockchain.

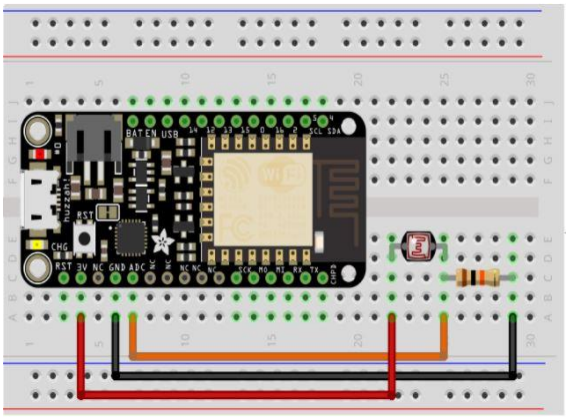
5.1.1 DEFINIÇÃO DAS COISAS: CENÁRIO 1

Considerando recursos da casa inteligente como cenário de estudo de caso para ilustrar a implantação do modelo arquitetural, em dois cômodos da casa inteligente foram instalados dois sensores de luminosidade (“Light Dependent Resistor ou Resistor Dependente de Luz – LDR) com o objetivo de ligar de forma automática as lâmpadas dos respectivos cômodos.

A aplicação dos sensores LDR no cenário do estudo de caso é motivada pelo objetivo de economia de energia e também pelo desejo de monitoramento e controle da luminosidade nos cômodos. Ao ler a incidência da luz dentro dos cômodos, os sensores LDR enviam os dados coletados para dois módulos NodeMCU ESP-12, que estão conectados em placas protoboards distintas.

Para a montagem do circuito entre o sensor LDR e o NodeMCU ESP-12, foram utilizadas duas placas de ensaio eletrônico conhecido como protoboards, dois resistores de 10K para limitar a corrente elétrica no sensor LDR e fios condutores para realizar as conexões (conhecidos como “Jumpers”), como mostra a Figura 5.3.

Figura 5. 3. Circuito LDR/NodemMcu ESP-12.



Fonte: Autor deste Trabalho.

O módulo NodeMCU ESP-12 é uma plataforma open source, criado com o objetivo de ser utilizado em projetos de IoT, composta basicamente por uma porta micro USB para alimentação e atualização do firmware, conversor USB integrado e Wifi

nativo (KASHYAP; SHARMA; GUPTA, 2018). As principais características dos módulos NodeMCU ESP-12 utilizados no cenário são: 4Mb de memória flash, WiFi nativo padrão 802.11b/g/n e ser compatível com a IDE do Arduino.

O desenvolvimento do script para a atualização do firmware dos módulos NodeMCU ESP-12 foi realizado através da IDE do Arduíno. Neste script foi especificado o endereço de rede para conexão entre os módulos e o sistema proposto como mostrado na Figura 5.4 (linha 3). A linha 5 contém a variável responsável por armazenar de forma temporária os dados coletados pelos dispositivos IoT antes de serem enviados para o sistema proposto via HTTP POST.

O intervalo de cada leitura nos sensores, diante do ambiente inserido, foi estabelecido em escala de tempo de 1000 milissegundos com apenas trinta iterações para o experimento, com o intuito de avaliar o comportamento e estrutura deste cenário, como mostra na linha 12.

Figura 5. 4. Configuração e Conexão de Envio dos Dados Para o Módulo Gateway IoT/Blockchain.

```
1   if(WiFi.status() == WL_CONNECTED){
2       HTTPClient http;
3       http.begin("http://192.168.0.103/Modulo_GatewayIoT_Blockchain/Validar_dados/");
4        http.addHeader("Content-Type", "application/x-www-form-urlencoded");
5       int httpCode = http.POST("temp=value");
6       String payload = http.getString();
7       Serial.println(httpCode);
8       Serial.println(payload);
9       http.end();
10  }
11  // tempo de 1 segundos ou 1000 milissegundos
12  delay(1000);
```

Fonte: Autor deste Trabalho.

A declaração dos parâmetros de conexão dos módulos NodeMCU ESP-12 com o gateway foi estabelecida através da configuração do “Identificador do Conjunto de Serviços - ssid” e do password, como mostra as linhas 2 e 3 da Figura 5.5.

Figura 5. 5. Configuração e Conexão com o Gateway.

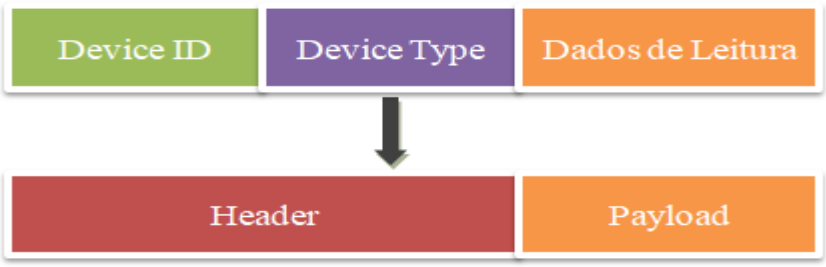
```
1   // Declaramos os parâmetros do WiFi, o usuário e a senha.
2   const char* ssid     = "PORTALNET_EDGAR";
3   const char* password = "Edgar2020!!";
4   void setup(){
5   Serial.begin(9600); //Iniciamos a comunicação serial com a velocidade de 9600
6   WiFi.begin(ssid, password); //Iniciamos a conexão WiFi
7   //mostra se houve a conexão
8   while(WiFi.status() != WL_CONNECTED){
9       delay(500);
10      Serial.println("Aguardando a Conexão");
11  }
```

Fonte: Autor deste Trabalho.

Após realizado as configurações dos dispositivos IoT, os dados obtidos dos ambientes em que os dispositivos IoT foram inseridos são enviados para o sistema proposto e são recepcionados pelos métodos da camada de transparência com o objetivo principal de verificar e autenticar se os dispositivos IoT estão cadastrados dentro do sistema. Esta verificação é realizada dentro da camada de persistência. Após o resultado positivo de cadastro dos dispositivos IoT, os métodos da camada de transparência (verificar e autenticar) irão validar os pacotes de dados obtidos pelos dispositivos IoT conforme os protocolos estabelecidos pelo sistema proposto, mostrados na Figura 5.6.

O pacote de dados enviados pelos dispositivos IoT para o sistema proposto corresponde a um bloco de dados, composto genericamente de dois campos, o primeiro campo “Header – (Cabeçalho)” e o segundo campo denominado “Payload”. O primeiro campo, Header, contém a identificação do dispositivo, e tem o objetivo de identificar e garantir que as informações geradas foram de um dispositivo cadastrado no sistema proposto. O Header é composto por: Device Id – (Id do Dispositivo); DeviceType – (Tipo do Dispositivo). O segundo campo é o Payload, que é responsável por carregar os dados lidos do ambiente, “Data value - (Dados coletados no ambiente)”.

Figura 5. 6. Pacote de Dados.



Fonte: Autor deste Trabalho.

Após a confirmação de acesso dos dispositivos IoT no sistema proposto e a validação dos dados, os métodos da camada de transparência realizam o encaminhamento da solicitação de verificação de cadastro dos dispositivos IoT ao método autenticador de Blockchain na camada de persistência. Após a confirmação do cadastro do serviço Blockchain, o método autenticador de Blockchain envia a resposta para os métodos da camada de transparência, liberando assim o envio dos dados para o serviço Blockchain.

O sistema proposto foi instalado em um servidor Linux, emulando a execução do sistema dentro do gateway. O servidor Linux utilizado para armazenar o sistema proposto contém as seguintes configurações: processador Intel Xeon Quad Core 8GB, HD 2TB e placa de vídeo Geforce GT CorPCSafe. Para acessar o sistema foi utilizado um notebook contendo as seguintes características: processador Intel 8ª Geração Core i5-8565U, 8GB RAM, HD 1TB, placa de vídeo Intel® UHD Graphics 620 e Windows 10.

5.1.2 EXECUÇÃO E AVALIAÇÃO DO ESTUDO DE CASO

Conforme pode ser visto na Figura 5.7, após a execução de todo o processo para a realização dos experimentos na estrutura da arquitetura proposta, obtivemos como resultado a cadeia de blocos (Blockchain) gerada contendo o primeiro bloco denominado gênesis entre a linha 3 até a linha 10.

Na linha 4 o índice nonce indica o número de iterações usadas para encontrar o hash válido, ou seja, a quantidade de tentativas para a resolução da prova de trabalho. Na linha 5 o índice index mostra o número de identificação do bloco iniciando com o id de número zero. Na linha 6, o timestamp define a hora e a data de criação do bloco. Na linha 7 o índice data mostra a identificação da criação do bloco gênesis. Na linha 8 o índice previoushash é responsável por receber o valor do hash do bloco o anterior; ele é iniciado no bloco gênesis com o valor NULL. Por fim, a linha 9 contém o valor hash do bloco, indicando a validação da operação na criação do bloco.

Figura 5. 7. Cadeia de Blocos - Blockchain.


```
{
    "chain": [
        {
            "nonce": 0,
            "index": 0,
            "timestamp": 1483225200,
            "data": "Genesis Block",
            "previousHash": null,
            "hash": "34054c41850bdf3f6f62dd60ae824dd35b98c872d420ed0dde942d0e54cd94d6"
        },
        {
            "nonce": 35868,
            "index": 1,
            "timestamp": 1605365467,
            "data": "1",
            "previousHash": "34054c41850bdf3f6f62dd60ae824dd35b98c872d420ed0dde942d0e54cd94d6",
            "hash": "000063e411e0742b191a3c0468db9faadc5bb20224e49a86d654d60256fec2dc"
        },
        {
            "nonce": 26215,
            "index": 2,
            "timestamp": 1605365467,
            "data": "2",
            "previousHash": "000063e411e0742b191a3c0468db9faadc5bb20224e49a86d654d60256fec2dc",
            "hash": "000076d2d6fad61825176431ad972f1533ce626629a1db9d87c882200b5d1f66"
        },
        {
            "nonce": 12821,
            "index": 3,
            "timestamp": 1605365467,
            "data": "3",
            "previousHash": "000076d2d6fad61825176431ad972f1533ce626629a1db9d87c882200b5d1f66",
            "hash": "0000da7917c6b3d30af32a6ae7472e3a89e69795a034ce9d181f7d71203ef0ee"
        },
        {
            "nonce": 55395,
            "index": 4,
            "timestamp": 1605365467,
            "data": "4",
            "previousHash": "0000da7917c6b3d30af32a6ae7472e3a89e69795a034ce9d181f7d71203ef0ee",
            "hash": "000045ca7094d4fce5f51491ee81822e86c165631e26204a3e5852048f22b2fd"
        },

}
```


Fonte: Autor deste Trabalho.

A partir da execução do sistema foi possível avaliar o comportamento estrutural da arquitetura proposta. Considerando que foram obtidos resultados contendo informações sobre a criação da cadeia de blocos (Blockchain), é possível afirmar que a solução cumpriu o seu objetivo, dado que foi possível coletar dados do ambiente no qual os dispositivos IoT estavam inseridos e encaminhá-los para o sistema proposto. Uma vez os dados coletados dentro do sistema proposto, os mesmos foram tratados e encaminhados para o serviço Blockchain, realizando assim, o ciclo completo de uso da solução proposta.

5.2 ESTUDO DE CASO 2: Avaliação de Desempenho

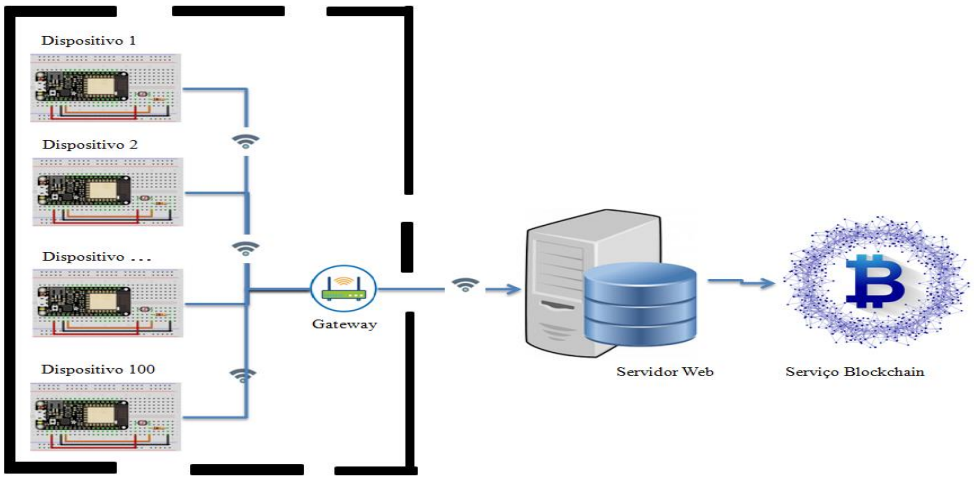
Segundo (MENDES, 2020), existem três aspectos primordiais para avaliar o desempenho de um sistema Web no contexto da experiência do usuário. O primeiro é o tempo gasto na realização da tarefa solicitada. O segundo é a resposta do navegador. Por fim, o terceiro é a visão do usuário final em relação à solicitação da página requisitada. Nesta avaliação, será focado o aspecto de tempo gasto para atender a solicitação do usuário.

A avaliação de desempenho é realizada aplicando cargas de trabalho em um ambiente virtualizado, com 4 tipos diferentes de clientes virtuais criados sobre a plataforma JMeter.

Para esta avaliação, serão adotados dois cenários. No primeiro cenário, o estudo de caso é executado sem a solução proposta (caso base), ou seja, sem a utilização do Módulo Gateway IoT/Blockchain. Já no segundo cenário, o estudo de caso é executado utilizando a solução proposta nesta dissertação (Módulo Gateway IoT/Blockchain), se efetuando inclusive a conexão com o Blockchain. O objetivo é comparar os dois cenários para mensurar a sobrecarga de desempenho imposta pela solução. Com isto, será possível analisar se a sobrecarga imposta se encontra em um limite aceitável tendo em vista a percepção do usuário.

Assim, o primeiro cenário consiste em avaliar o tempo de execução na geração e envio dos dados dos dispositivos IoT até o Servidor Web, onde os dados serão processados e enviados para o serviço Blockchain, como ilustra a Figura 5.8.

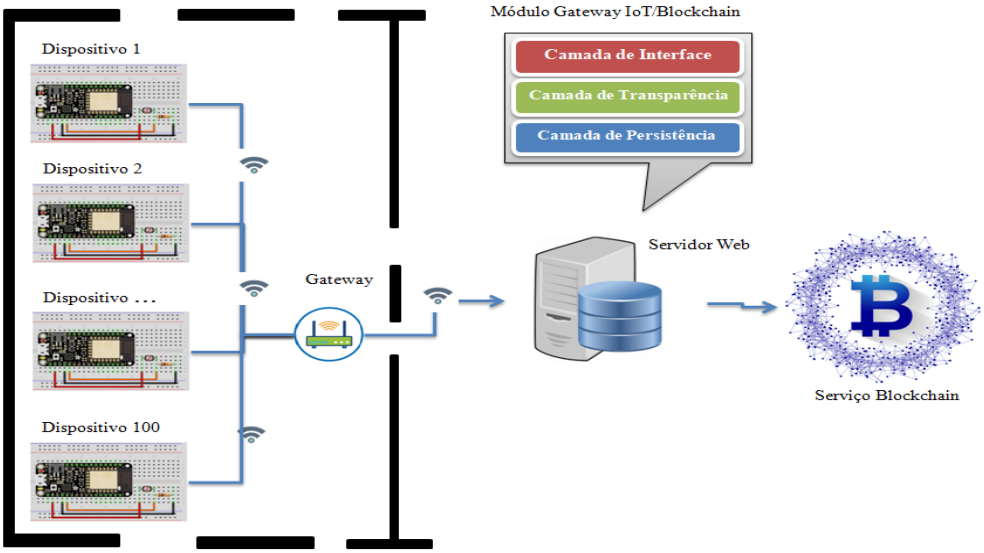
Figura 5. 8. Primeiro Cenário.



Fonte: Autor deste Trabalho.

Por sua vez, o segundo cenário consiste em avaliar o tempo de execução usando a solução proposta nesta dissertação, onde os dados serão processados e enviados para o serviço Blockchain por meio do Módulo Gateway IoT/Blockchain, como mostra a Figura 5.9.

Figura 5. 9. Segundo Cenário.



Fonte: Autor deste Trabalho.

A idéia dos dois cenários é avaliar a sobrecarga de desempenho imposta pela solução proposta nesta dissertação. Desta forma, o primeiro cenário não irá utilizar a solução, e o seu desempenho será medido. Depois, o segundo cenário é executado, e a solução é utilizada. Por fim, se avalia a diferença de desempenho observada nos dois cenários para poder avaliar se a solução proposta impõe uma sobrecarga de desempenho dentro de um limite aceitável.

Segundo os autores (CHEN; FARLEY; YE, 2004), o principal requisito que influencia negativamente os usuários em navegações de páginas da Web é o tempo de resposta (execução) das requisições realizadas. O tempo de resposta impacta diretamente na aceitação do usuário final se o valor no tempo de resposta for superior a 5 (cinco) segundos.

Devido ao conjunto de critérios estabelecidos, a métrica utilizada para avaliar o desempenho foi o tempo de execução. O tempo de execução é o tempo necessário para o atendimento de uma solicitação do usuário, onde sua unidade de tempo representativo é segundos.

5.2.1 DEFINIÇÃO DA CARGA DO EXPERIMENTO

Por cumprir os requisitos necessários para a execução da carga (requisições) nos dois experimentos, foi utilizada a ferramenta JMeter na versão 5.2.1. O teste de carga do experimento tem por objetivo avaliar se os resultados estão de acordo com o esperado, garantindo assim níveis aceitáveis de qualidade de serviço ao usuário final (SARZOSA CHOQUE, 2019).

Os valores aplicados para ambos os experimentos com clientes virtuais são de quantidades diferenciadas, iniciando com valor de um cliente virtual (Dispositivo IoT) e aumentando gradativamente, em uma variação de vinte e cinco clientes virtuais até o valor máximo de cem clientes virtuais, com o intuito de se encontrar um limite de capacidade do sistema proposto.

A execução dos experimentos foi realizada primeiramente com um cliente virtual disparando uma requisição por vez, sinalizando uma iteração. Este processo foi realizado até completar o valor de 30 iterações. Seguindo esta forma de coleta de dados, o processo para aquisição dos dados coletados dos demais clientes virtuais foi realizado de forma semelhante. O tempo utilizado para a inicialização das requisições foram no valor de 2 milissegundos.

O número de requisições realizadas para os experimentos pelos clientes virtuais é respectivamente igual à quantidade de clientes virtuais utilizados, como ilustra a Tabela 5.1.

Tabela 5. 1. Valores de Carga.

CLIENTES VIRTUAIS	REQUISIÇÕES	ITERAÇÕES	TEMPO DE INICIALIZAÇÃO
01	01	30	2 milissegundos
25	25		
50	50		
100	100		

Fonte: Autor deste Trabalho.

Foi pré-definida a quantidade de 30 repetições/iterações para cada variação de cliente virtual (Dispositivos IoT) aplicado na simulação, com o objetivo de observar o comportamento da arquitetura proposta sobre altas demandas, de forma que seja representada sua utilização em situação normal de uso. Segundo o autor (MORAIS;

CALLOU; LINS, 2020), a quantidade de 30 repetições traz um intervalo de confiança maior nos resultados, ficando assim mais próximo de uma situação real.

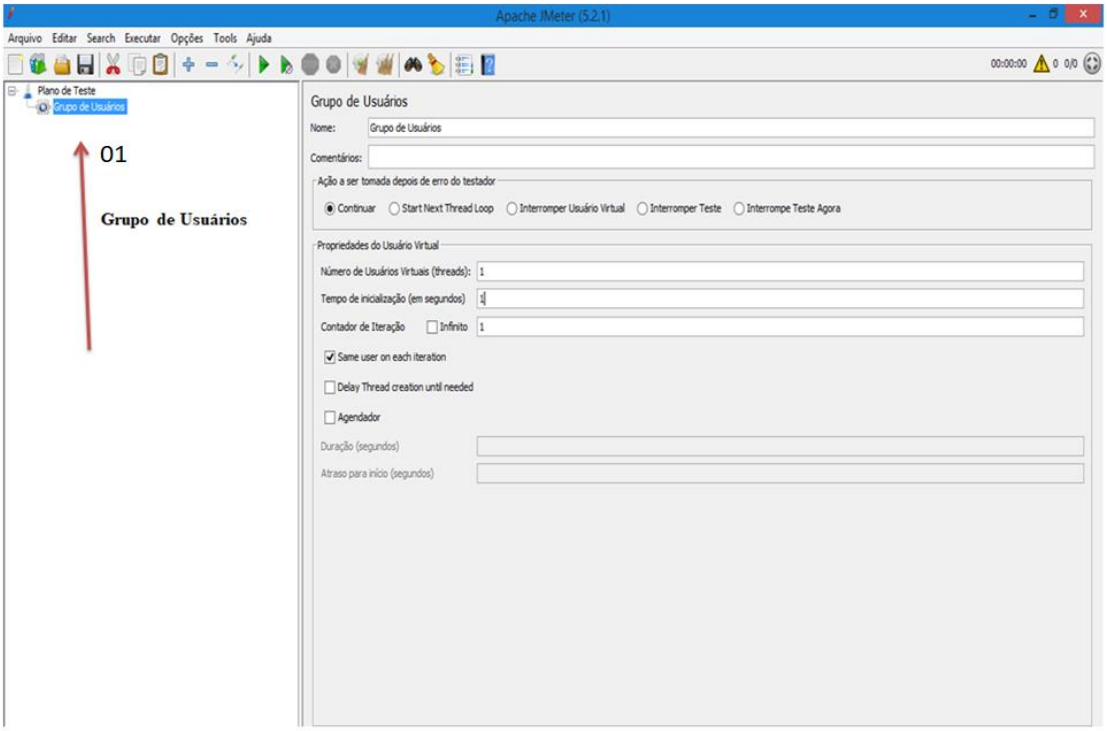
Inicialmente, foi criado o Plano de Testes dentro do JMeter, que é responsável pelo controle e execução dos grupos de usuários, ilustrado na Figura 5.10, no Quadro A(seta 01). Dentro dos grupos de usuários mostrados na figura, foram configuradas as execuções dos processos dos testes, definindo se cada grupo de usuário deverá ser executado de forma simultânea ou não, que tipo de controle será utilizado e qual será a definição da quantidade de usuários que irão interagir com arquitetura proposta e por quanto tempo durará esta interação.

No grupo de usuários foi utilizado inicialmente um único cliente virtual, significando que houve o envio de apenas uma única requisição, com o tempo de inicialização de dois milissegundos com apenas uma repetição, conforme a Figura 5.10 no Quadro B(seta 02). Os valores apresentados das métricas foram plotados na escala de tempo de milissegundos para padronizar com as unidades de tempo comumente usadas nos dispositivos IoT.

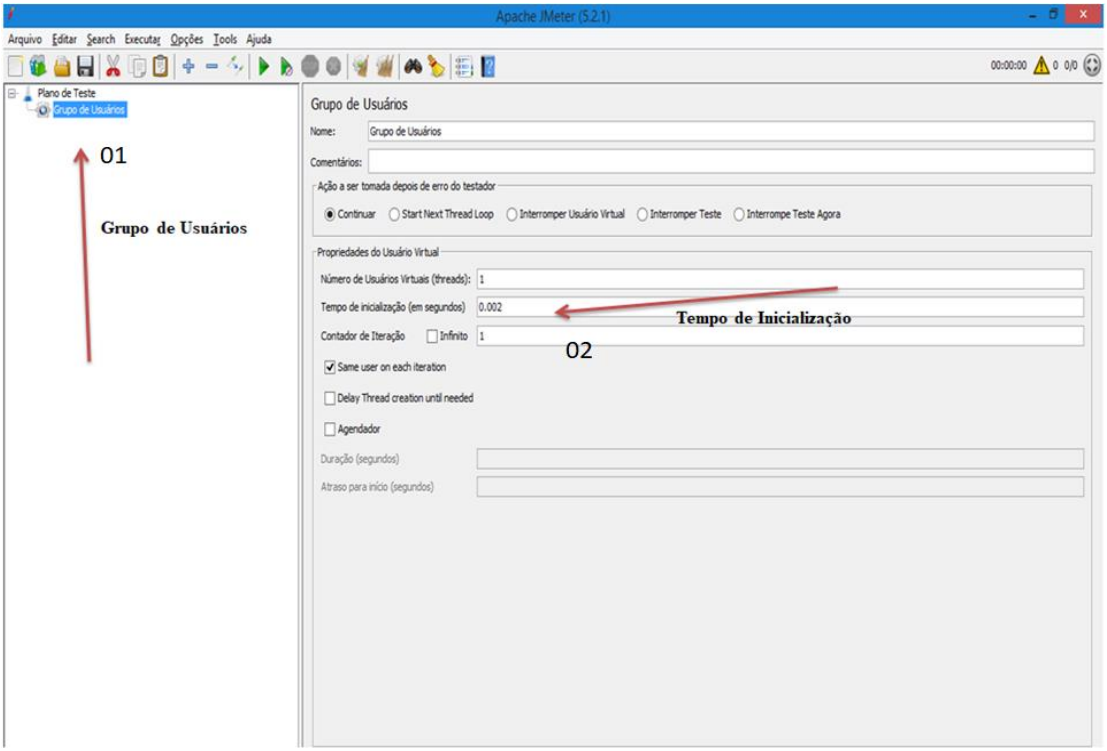
Dado que o JMeter permite realizar a criação de teste utilizando diversos tipos de protocolos (ex.:HTTP, JDBC e FTP), dentro de cada grupo de usuário foi utilizado o gerenciador de cabeçalhos HTTP, que tem por objetivo armazenar as configurações gerais de requisições HTTP. Dentro deste gerenciador HTTP, os componentes de acesso ao sistema proposto foram inseridos, como: o endereço do servidor onde está hospedada arquitetura proposta, a porta de acesso ao servidor e quais os tipos de métodos de envio, POST ou GET, conforme a Figura 5.10 no Quadro C(seta 03).

Figura 5. 10. JMeter – Plano de Teste.

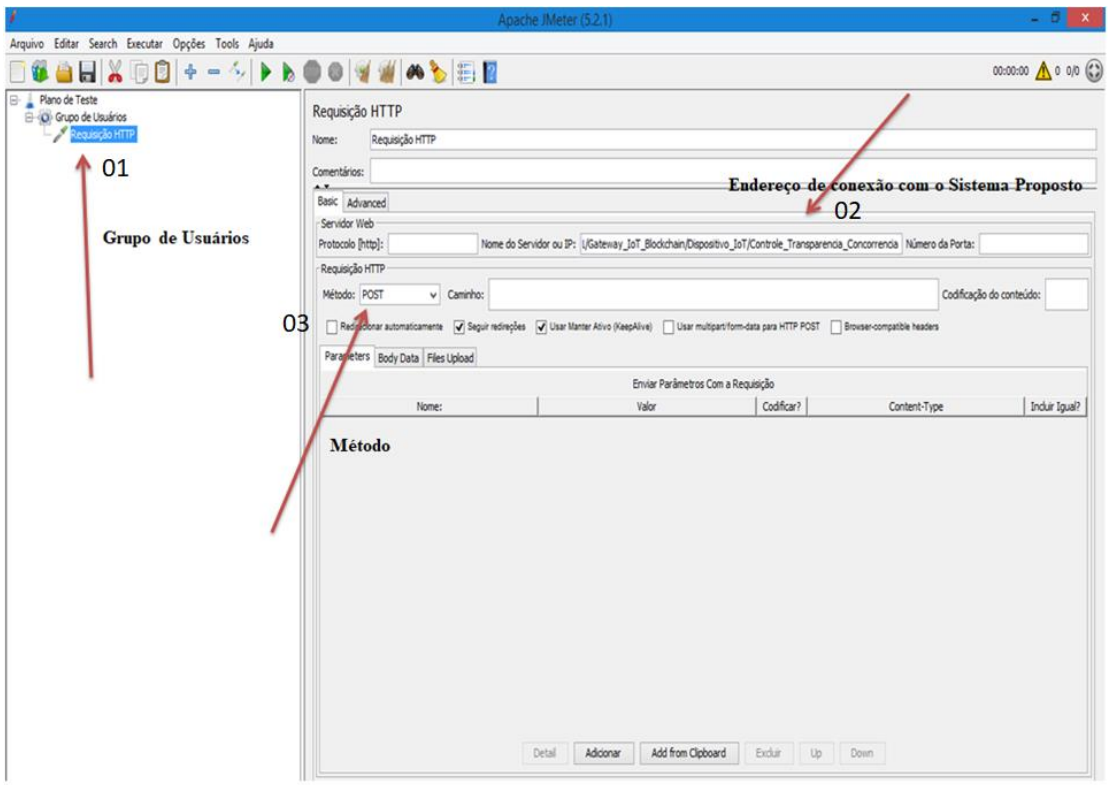
A



B



C

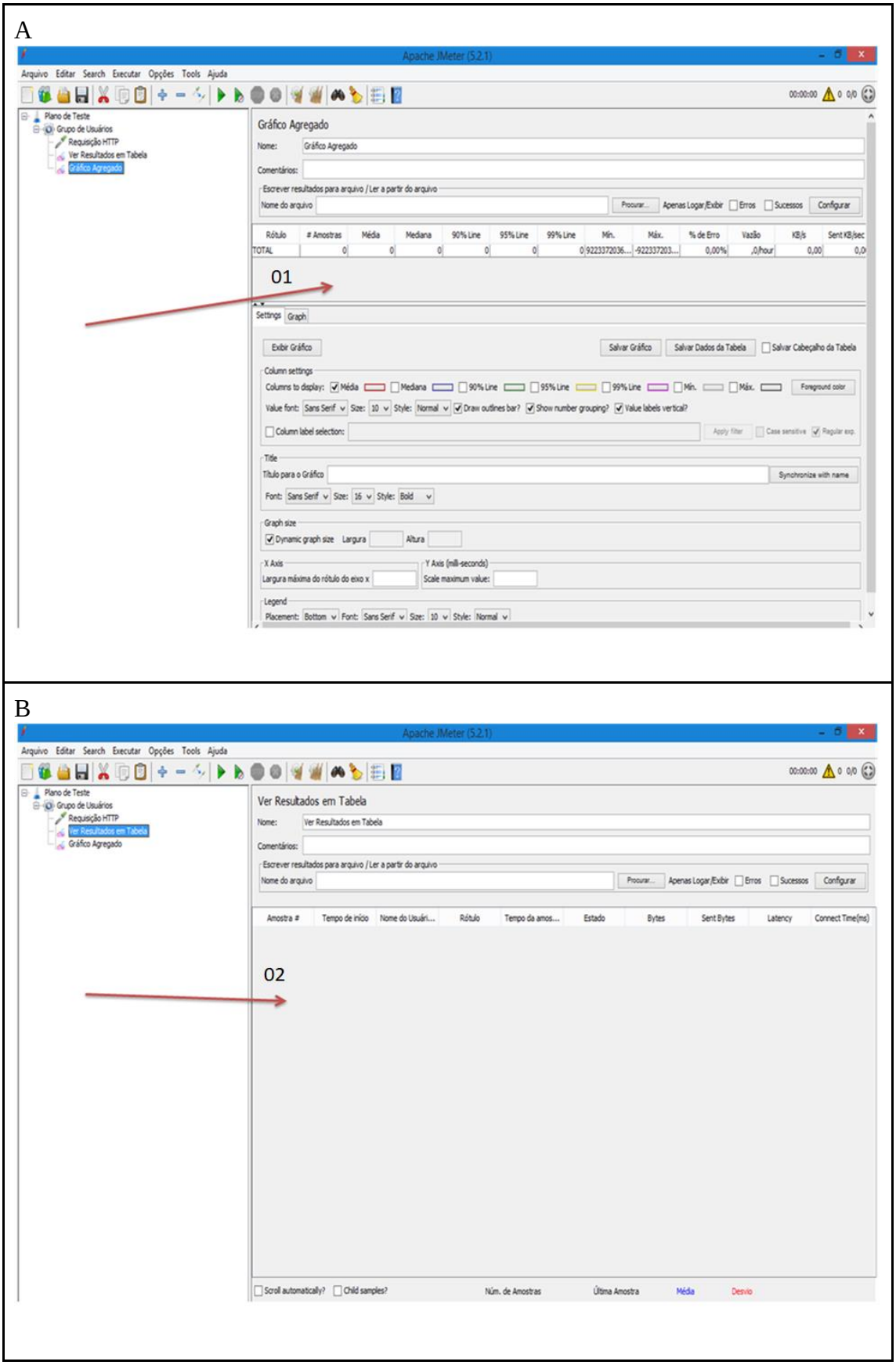


Fonte: Autor deste Trabalho.

Para todos os testes de simulação de carga, foram adicionados os ouvintes. No JMeter os ouvintes são responsáveis por imprimir relatórios para os usuários de forma automática, logo após a execução dos testes. Foram adicionados dois tipos de ouvintes para a obtenção dos resultados dos testes de carga: o primeiro ouvinte é denominado “Ver Resultados em Tabela”. Este ouvinte disponibiliza os seguintes atributos em seu relatório: tempo de início da requisição, tempo da amostra em escala de tempo em milissegundos, latência, estado das requisições, dentre outros. O quadro A(seta 01) da Figura 5.11, ilustra o exposto.

O segundo ouvinte chama-se “Gráfico Agregado”, que é responsável por disponibilizar a média, a mediana, o valor mínimo e o valor máximo da amostra, como também, o valor da vazão da requisição realizada dos testes, conforme o quadro B(seta 02) da Figura 5.11.

Figura 5. 11. JMeter - Ouvintes.



Fonte: Autor deste Trabalho.

Para iniciar a execução do experimento para o primeiro cenário, foi definido o endereço eletrônico de rede para o acesso do JMeter: “127.0.0.1/Gateway_IoT_Blockchain/Dispositivo_IoT/encaminha_blockchain.php”, sendo adicionado a pasta e o arquivo para o acesso direto através da URL.

No entanto, para o segundo cenário, o endereço eletrônico para acesso do JMeter ao sistema proposto foi: “127.0.0.1/Gateway_IoT_Blockchain/Dispositivo_IoT/encaminha_blockchain”, sendo que neste experimento é disponibilizado o nome da classe e do método para o acesso.

No entanto, para que esse acesso direto do JMeter possa ser realizado, foi desativado o script responsável por bloquear o acesso direto às camadas da arquitetura proposta, como ilustrado na Figura 5.12.

Na linha 2 até a linha 25 da Figura 5.12 é descrito o script (Conjunto de Instrução) responsável por bloquear o acesso direto a camada de transparência e de persistência da arquitetura proposta, fazendo com que o acesso seja realizado apenas pela camada de interface.

Figura 5. 12. Script de Bloqueio no Acesso Direto.

```
1   <?php
2   $requi = $_SERVER["HTTP_REFERER"]; //http_referer = se existe o endereço da pagina
3     $requi= strtolower("/$requi/"); //coloca tudo pra letra minuscula
4
5   $server = $_SERVER['SERVER_NAME']; // O nome host do servidor onde o script atual é executado
6     $server= strtolower("/$server/"); //coloca tudo pra letra minuscula
7
8   if(preg_match($server, $requi) == 0):
9
10    header("Location: https://www.google.com.br");
11
12    die();
13
14  else:
15
16    session_start();
17
18    if($_SESSION['usuario_nome'] and $_SESSION['usuario_mat'] and $_SESSION['usuario_ms']):
19
20    else:
21        //header("location:" . URL_BASE . "Index");
22    endif;
23
24
25        date_default_timezone_set('America/Sao_Paulo');
26
27  ?>
```

Fonte: Autor deste Trabalho.

5.2.2EXECUÇÃO E AVALIAÇÃO DOS RESULTADOS

Na execução da avaliação de desempenho, foi definido o uso de diferentes grupos de clientes virtuais para prover os resultados desejados. Com isso, foi separado um servidor local isolado para ambos os experimentos utilizados na coleta dos dados. Este isolamento do servidor local foi para evitar interferências e alterações nos resultados coletados.

Os resultados referentes à análise do tempo de execução utilizando o protocolo HTTP foram medidos e avaliados de acordo com a quantidade de clientes virtuais, requisições e iterações já definidas na Tabela 5.1. A Tabela 5.2 apresenta o resultado geral dos experimentos realizados em escala crescente usando como referência os clientes virtuais para o primeiro cenário.

Tabela 5. 2. Tempo de execução medido do primeiro cenário (caso base).

CLIENTES VIRTUAIS	ITERAÇÕES	TEMPO DE RESPOSTA MÉDIA (ms)
01	30	23
25	**30**	150,1
50	**30**	309,2
100	**30**	813

Fonte: Autor deste Trabalho.

De acordo com os dados apresentados na Tabela 5.2, é possível obter o valor da média quando aplicado 30 interações com um cliente virtual (dispositivos IoT) obtendo assim o valor resultante de 23 milissegundos. Porém, quando realizado os experimentos com o valor de vinte e cinco clientes virtuais o resultado da média do tempo de execução foi de 150,1 milissegundos. Sendo o valor de clientes virtuais aumentado para cinquenta, a média obtida através das 30 iterações foi de 309,2 milissegundos.

Por fim, foi estabelecido o experimento com cem clientes virtuais, e a média resultante foi no valor de 813 milissegundos. É possível identificar que, quando analisada a quantidade com cem clientes virtuais, o valor resultante da média está dentro do aceitável em comparação com o valor de tempo de execução aplicado a sistemas Web utilizando o protocolo HTTP, que é de 5000 milissegundos (CHEN; FARLEY; YE, 2004).

Desta forma, é plausível pressupor que, no primeiro cenário, de acordo com a Tabela 5.2, a média do tempo de execução é diretamente proporcional ao aumento na quantidade de clientes virtuais, ou seja, quanto mais aumenta a quantidade de clientes virtuais, também aumenta o valor da média do tempo de execução. Contudo, com o aumento no valor da média no tempo de execução sendo proporcionalmente ao número de clientes virtuais (dispositivos IoT), os valores obtidos ainda se mantém tolerável.

Os resultados referentes à análise do tempo de execução utilizando o protocolo HTTP no segundo experimento foram medidos e avaliados de acordo com a quantidade de clientes virtuais, requisições e iterações já pré-definidas na Tabela 5.1.

A Tabela 5.3 apresenta o resultado geral dos experimentos realizados aplicado ao segundo cenário, que consiste no ciclo completo da arquitetura proposta (Módulo Gateway IoT/Blockchain) em escala crescente usando como referência os clientes virtuais.

Tabela 5. 3. Tempo de execução medido do segundo cenário.

CLIENTES VIRTUAIS	ITERAÇÕES	TEMPO DE RESPOSTA MÉDIA (ms)
01	30	45,1
25	**30**	265,2
50	**30**	475,2
100	**30**	1501,5

Fonte: Autor deste Trabalho.

Na primeira execução realizada com um cliente virtual (dispositivo IoT) utilizando 30 iterações, é possível identificar a média geral no valor de 45,1 milissegundos. Quando o valor aplicado aumentou para vinte e cinco clientes virtuais realizando 30 iterações, o valor gerado foi uma média de 265,2 milissegundos.

Pode-se afirmar que o valor obtido é tolerável para uma aplicação Web. Adicionalmente, foram realizadas outras duas simulações da mesma operação, aumentado a quantidade de clientes virtuais para o valor de cinquenta e cem clientes. Os valores resultantes para os experimentos com cinquenta e cem clientes virtuais a 30 iterações foram respectivamente 475,2 e 1501,5 milissegundos. Sendo possível analisar a relação direta entre a quantidade de clientes virtuais aplicados no segundo cenário e a média do tempo de execução obtido em cada iteração, foi identificado que a quantidade

de clientes virtuais podem ser considerados diretamente proporcionais às médias gerais obtidas em cada iteração, ou seja, aumentando o número de clientes também aumenta o valor médio no tempo de resposta.

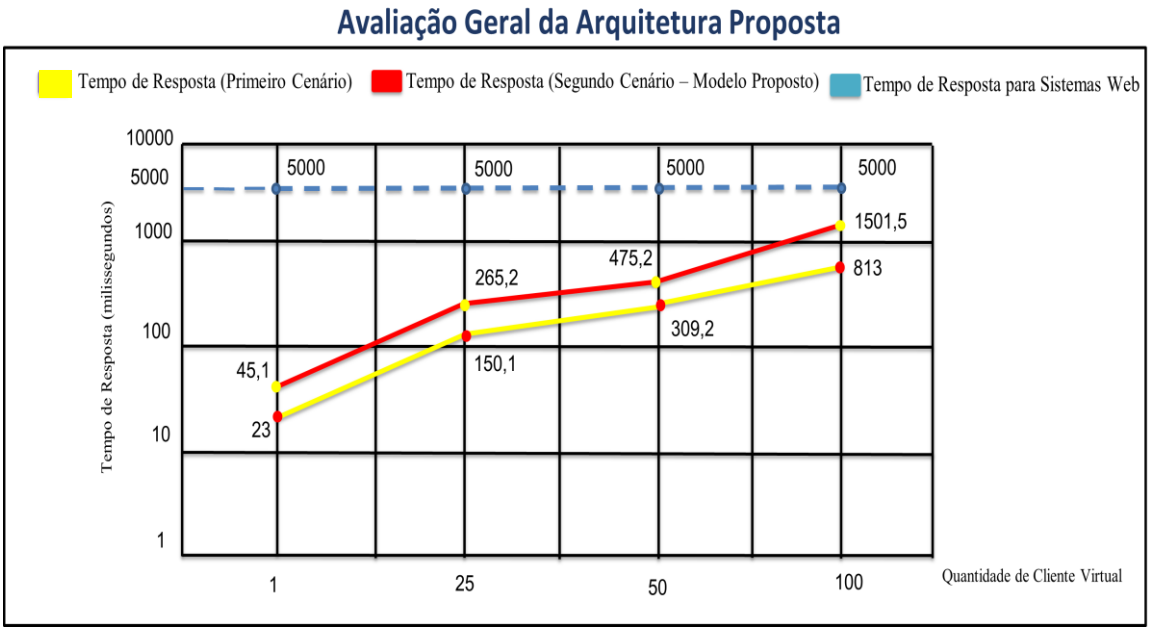
Ao se analisar o resultado geral nos dois cenários realizados, os valores obtidos dos experimentos com apenas um cliente virtual (primeiro grupo) tiveram um aumento na variação entre eles de aproximadamente 1 (um) segundo. Contudo, esse valor é compreensivo, devido a todo o processo de validação e autenticação em que os dados são submetidos dentro do sistema proposto. Já na análise geral com vinte e cinco clientes virtuais (segundo grupo) também houve um aumento na variação no valor de aproximadamente 1 (um) segundo.

Quando analisado os valores gerais obtidos com cinquenta clientes virtuais (terceiro grupo) foi percebido também um aumento aproximado na variação de 1/2 (meio) segundo, contudo, foi verificado que o valor de variação foi menor em relação ao valor obtido com vinte e cinco clientes virtuais, porém, o aumento entre as variações analisadas nos três primeiros grupos de clientes virtuais, pode indicar um padrão de comportamento com tendência de crescimento.

Por fim, o valor aproximado do aumento na variação com cem clientes virtuais (quarto grupo) foi de 1 (um) segundo, demonstrando o início de uma tendência de disparidade mais acentuada a partir desta quantidade de clientes (dispositivo IoT) e projetando um aumento no valor da variação com o terceiro grupo de cliente virtual.

É importante ressaltar que o maior valor obtido no segundo cenário (Módulo Gateway IoT/Blockchain) foi de 1501,5 milissegundos, indicando que o resultado está abaixo do valor usado como referência para aplicação em métricas de QoS para navegação Web em sistemas convencionais utilizando o protocolo HTTP, que é de 5000 milissegundos, o que pode ser considerado um aspecto positivo, como ilustra a Figura 5.13.

Figura 5. 13. Avaliação Geral.



Fonte: Autor deste Trabalho.

5.3 CONSIDERAÇÕES FINAIS DO CAPÍTULO

Neste capítulo foram apresentados os estudos de casos, que foram desenvolvidos a partir do modelo arquitetural proposto. O estudo de caso 1 foi proposto para analisar o funcionamento e o comportamento do sistema como um todo, tendo como objetivo geral enviar dados dos dispositivos IoT até o sistema proposto e, assim, encaminhar os dados para o serviço Blockchain. No estudo de caso 2 foi realizada uma avaliação de desempenho com dois cenários distintos. O primeiro cenário consiste em avaliar o tempo de execução na geração e envio dos dados dos dispositivos IoT até o Servidor Web e em seguida ao serviço Blockchain. O segundo cenário realiza a avaliação de desempenho com a solução proposta (Módulo Gateway IoT/Blockchain). Nos dois cenários, foram realizados experimentos para avaliar as métricas que influenciam na percepção do usuário na utilização do sistema. Foi possível concluir que o uso da solução proposta nesta dissertação acarretou em uma sobrecarga de desempenho tolerável.

6 CONCLUSÕES E TRABALHOS FUTUROS

Neste capítulo, serão apresentadas as conclusões, limitações e trabalhos futuros associados a esta dissertação.

6.1 CONCLUSÕES

O crescimento tecnológico vem se mostrando acelerado e a tecnologia das redes Blockchain tem se mostrado uma alternativa interessante para prover para os sistemas de Internet das Coisas é uma alternativa para melhorar a segurança e a imutabilidade dos dados gerados. No entanto, a integração entre essas duas tecnologias apresenta desafios não triviais.

Tendo em vista a complexidade desta integração, este trabalho propôs uma arquitetura para transparência de conectividade de serviços Blockchain em sistemas IoT, de forma que o usuário final possa utilizar serviços Blockchain em seus sistemas IoT de forma facilitada.

A arquitetura proposta foi estruturada baseada na hierarquia em camadas, onde cada camada da arquitetura implementa uma parte do sistema. A solução proposta permite que dispositivos IoT enviem dados coletados do ambiente em que estão para um serviço Blockchain.

A arquitetura em três camadas promoveu a separação das funcionalidades, permitindo a aplicação da transparência de acesso, possibilitando a ocultação na diferença da representação de dados e no modo de acesso ao usuário final, como também, contribuiu na inserção da transparência de localização, ocultando onde os recursos do sistema e os dados estão sendo armazenados.

A partir da execução e análise dos estudos de casos, foi possível observar que o desenvolvimento de uma arquitetura para transparência de conectividade de serviços Blockchain em sistemas IoT pode ser considerada uma contribuição interessante, pois diante dos resultados obtidos foi possível observar que, em ambiente controlado, o sistema proposto obteve valores de desempenho compatíveis com o aceitável para sistemas Web convencionais.

Adicionalmente, através da análise do estado da arte descrito no Capítulo 3, pode ser observado que nenhum dos trabalhos referenciados nesta pesquisa tem foco na

proposição de uma arquitetura para transparência de conectividade de serviços Blockchain em sistemas IoT. Este fato evidencia a importância da contribuição científica desta dissertação.

6.2 LIMITAÇÕES

Mesmo considerando que este trabalho buscou ao máximo variar as quantidades de clientes virtuais para a realização dos experimentos de avaliação de desempenho do sistema proposto, uma das limitações é que foi utilizado um valor máximo de cem clientes virtuais nos dois experimentos realizados, o que limita uma avaliação de desempenho mais profunda da estrutura da arquitetura proposta. Outra limitação a ser considerada neste trabalho reside na aplicação apenas da transparência de acesso e a transparência de localização na arquitetura proposta, dado que existem outros tipos de transparências utilizadas em sistemas distribuídos e que poderiam ser aplicadas.

Outra limitação relevante é que para a realização da análise da estrutura e do comportamento da arquitetura proposta, foi desenvolvido um Blockchain privado para a conexão com o sistema proposto. No entanto, a utilização deste Blockchain limita a análise sobre a arquitetura proposta apresentada, dado que o sistema poderia ser conectado a outros serviços de Blockchain disponíveis no mercado atual, como por exemplo, a Ethereum.

6.3 TRABALHOS FUTUROS

Diante das melhorias que podem ser realizadas na pesquisa desenvolvida nesta dissertação, se vislumbram os seguintes trabalhos futuros:

- Ampliar os estudos para análise e introdução das demais transparências aplicadas em sistemas distribuídos;
- Realizar experimentos para conectividade de serviços de Blockchain utilizadas no mercado, como a Ethereum que é uma plataforma on-line que permite a implementação de aplicações descentralizadas (dapps) e de contratos inteligentes (smart contracts);

- Avaliar uma forma de automatizar a conexão dos dispositivos IoT no sistema proposto, possibilitando uma melhor experiência de utilização para o usuário final;
- Realizar estudos com o objetivo de implantar o sistema proposto diretamente dentro do Gateway IoT, para abstrair do usuário a sensação de descentralização do sistema, dado que, o Módulo Gateway IoT/Blockchain poderá ser aplicado dentro do próprio gateway e o banco de dados em um servidor qualquer;
- Promover avaliação da arquitetura proposta com vários cenários de cargas, com o objetivo de tentar extrapolar os limites julgados aceitáveis de tolerância ao atraso da entrega.

Referências

3-Tier Architecture: A Complete Overview. Logi REPORT, 2020. Disponível em: <https://www.jinfonet.com/resources/bi-defined/3-tier-architecture-complete-overview/> Acesso em: 11 de setembro de 2020.

AHRAM, Tareqet al. Blockchain technologyinnovations. In: 2017 IEEE technology&engineering management conference (TEMSCON). IEEE, 2017. p. 137-141.

ALGHAMDI, Waleed; SCHUKAT, Michael. SlaveClock Responses to Precision Time ProtocolAttacks: A Case Study. In: 2020 International Conference on Cyber Security and Protection of Digital Services (Cyber Security). IEEE, 2020. p. 1-4.

ALI, Muhammad Salek; DOLUI, Koustabh; ANTONELLI, Fabio. IoT data privacy via blockchains and IPFS. In: Proceedings of the Seventh International Conference on the Internet of Things. ACM, 2017. p. 14.

ALI, NabeelSalih; SHIBGHATULLAH, A. S. Protection web applicationsusing real-time technique to detectstructured query languageinjectionattacks. International Journal of Computer Applications, v. 149, n. 6, p. 26-32, 2016.

ALMEIDA, Guilherme Augusto Machado de. Fatores de escolha entre metodologias de desenvolvimento de software tradicionais e ágeis. 2017. Tese de Doutorado. Universidade de São Paulo.

ALVES, William Pereira. Construindo uma Aplicação Web Completa com PHP e MySQL. Novatec Editora, 2018.

BANZI, Massimo; SHILOH, Michael. Primeiros passos com o Arduino. São Paulo: Novatec, p. p1, 2011.

BENAVÍDEZ GÓMEZ, Javer Andrés et al. Arquitectura REST para la plataforma UAOIoT. 2019. Trabalho de Conclusão de Curso. Universidad Autónoma de Occidente.

BENAVÍDEZ GÓMEZ, Javer Andrés et al. Arquitectura REST para la plataforma UAOIoT. 2019. Trabalho de Conclusão de Curso. Universidad Autónoma de Occidente.

BERNAL BLAY, Miguel Angel. Blockchain, administración y contratación pública. 2018.

BOMBARDELLI, Felipe Gustavo. URF (Framework Unificado de Robótica): proposta de interface para sistemas distribuídos.

BRITO, George Lauro Ribeiro; DETTENBORN, Renê; BARBOSA, Gentil Veloso. Ambiente Expresso Para Processamento Distribuído com Matlab. In: Prooceddings Of XxxiiCilamce: IberianLatin American Congress on ComputationalMethods in Engineering, Ouro Preto-MG. 2011.

CÂNDIDO, Aubani Júnio Teixeira. Protocolo para segurança da informação em gateways IoT. 2019.

CÂNDIDO, AubaniJúnio Teixeira. Protocolo para segurança da informação em gateways IoT. 2019.

CANGIANELLI, Daniel Negreiros. Protocolo de comunicação GPS-Servidor utilizando MQTT. 2019.

CARLOZO, Lou. What is blockchain?.Journal of Accountancy, v. 224, n. 1, p. 29, 2017
CARVALHO, Rogério Atem de. Carteira de Cursos Baseada em Tecnologia Blockchain. 2019.

CHEN, Yan; FARLEY, Toni; YE, Nong. QoS requirements of network applications on the Internet. InformationKnowledge Systems Management, v. 4, n. 1, p. 55-76, 2004.
CHERNICK, Michael R. et al. Bootstrap methods. 2011.

CHICARINO, V. R. et al. Uso de blockchain para privacidade e segurança em internet das coisas. Livro de Minicursos do VII Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais. Brasília: SBC, v. 28, 2017.

COELHO, Flávia Estélia Silva et al. Avaliação de disponibilidade de redes de computadores baseadas na arquitetura cliente/servidor em n-camadas. 2000.

CORRELL, Kendall; BARENDT, Nick; BRANICKY, Michael. Design considerations for software onlyimplementations of the IEEE 1588 precision time protocol. In: Conference on IEEE. 2005. p. 11-15.

COSMI, Arthur Brito; MOTA, Vinicius FS. Uma Análise dos Protocolos de Comunicação para Internet das Coisas. In: Anais do III Workshop de Computação Urbana. SBC, 2019. p. 153-166.

COSMI, Arthur Brito; MOTA, Vinicius FS. Uma Análise dos Protocolos de Comunicação para Internet das Coisas. In: Anais do III Workshop de Computação Urbana. SBC, 2019. p. 153-166.

COSTA, Heitor Henrique de Paula Moraes. Controle de acesso na plataforma de nuvem federada BioNimbuZ. 2015.

COULOURIS, George et al. Sistemas Distribuídos-: Conceitos e Projeto. Bookman Editora, 2013.

CREEL, Kathleen A. Transparency in complexcomputational systems. 2020.

DA HORA, Diego Neves et al. Narrowing the gap between QoS metrics and Web QoE usingAbove-the-foldmetrics. In: International Conference on Passive and Active Network Measurement. Springer, Cham, 2018. p. 31-43.

DA SILVA, Jonas Ayres et al. Autenticação de dispositivos em redes IoT com gateways distribuídos. In: VII Mostra Científica do IFRS-Campus Restinga. 2017.

DALL'OGLIO, Pablo. PHP Programando com orientação a Objetos. Novatec Editora, 2018.

DE LEMOS, Maxmilian Ferreira et al. Aplicabilidade da arquitetura MVC em uma aplicação web (WebApps). RE3C-Revista Eletrônica Científica de Ciência da Computação, v. 8, n. 1, 2013.

DE LIMA, Victor Almeida Cabral; MONTEIRO, Marcel Stanlei. Blockchain: uma abordagem bibliográfica. REVISTA DE CIÊNCIAS SOCIAIS APLICADA, v. 1, n. 1, 2020.

DE OLIVEIRA, Sérgio. Internet das coisas com ESP8266, Arduino e Raspberry PI. Novatec Editora, 2017.

DEDEOGLU, Volkanet al. Blockchain technologies for iot. In: AdvancedApplications of Blockchain Technology. Springer, Singapore, 2020. p. 55-89.

DIAS, Rui Pedro das Neves. Análise de plataformas Blockchain. 2019. Tese de Doutorado. Universidade de Coimbra.

DO ESPÍRITO SANTO, Walter; ORDOÑEZ, Edward; RIBEIRO, Admilson. Uma revisão sistemática sobre a Segurança nos Protocolos de Comunicação para Internet das Coisas. Journal on Advances in Theoretical and Applied Informatics, v. 4, n. 1, p. 1-9, 2018.

DO ESPÍRITO SANTO, Walter; ORDOÑEZ, Edward; RIBEIRO, Admilson. Uma revisão sistemática sobre a Segurança nos Protocolos de Comunicação para Internet das Coisas. Journal on Advances in Theoretical and Applied Informatics, v. 4, n. 1, p. 1-9, 2018.

DORRI, Ali et al. Blockchain for IoT security and privacy: The case study of a smart home. In: PervasiveComputing and Communications Workshops (PerCom Workshops), 2017 IEEE International Conference on. IEEE, 2017. p. 618-623. Acesso em: 10 deSetembro de 2018.

DORRI, Ali; KANHERE, SalilS .; JURDAK, Raja. Para uma blockchain otimizada para IoT. In: Anais da Segunda Conferência Internacional sobre Design e Implementação da Internet das Coisas . ACM, 2017. p. 173-178.

DRESCHER, Daniel. Blockchain basics. Berkeley, CA: Apress, 2017.

ESTEVÃO, Bruno de Souza; SANDMANN, André; SANTOS, Izaias Batista dos. Aplicações ricas para Internet: proposta de Arquitetura de software na nuvem para atender ao Agronegócio. 2018.

EVARISTO, Bruno et al. Editora Científica Autônoma e Distribuída sobre Blockchain Privada. In: Anais do II Workshop em Blockchain: Teoria, Tecnologia e Aplicações. SBC, 2019.

FAHEEM, Muhammad; TUNA, Gurkan; GUNGOR, VehbiCagri. QERP: Quality-of-service (QoS) awareevolutionaryroutingprotocol for underwater wireless sensor networks. IEEE Systems Journal, v. 12, n. 3, p. 2066-2073, 2017.

FONTALBA, Paulo Lima E. Matheus. Uma Experiência Blockchain. Clube de Autores (managed), 2019.

FRAMINGHAM, Massachusetts, WorldwideSpending on the Internet of Things Will Slow in 2020 ThenReturn to Double-DigitGrowth, According to a New IDC SpendingGuide, 18 de junho de 2020. Disponível em: <https://www.idc.com/getdoc.jsp?containerId=prUS46609320 />, Acesso em: 23 Agosto 2020.

FURTADO, Fabricio Reis. L7SP: serviços para otimizar o gerenciamento e o desempenho de Blockchain Privados. 2019.

GARG, Ayushet al. Wireless precision time protocol. IEEE Communications Letters, v. 22, n. 4, p. 812-815, 2017.

GOMES, João Tomé Carvalho. Riscos e vulnerabilidades dos equipamentos IoT em unidades de saúde. 2019. Tese de Doutorado.

GREVE, Fabíola Greve et al. Blockchain e a Revolução do Consenso sob Demanda. Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC)-Minicursos, 2018.

GUINGO, Bruno Clemente et al. Blockchain para registro, certificação e rastreabilidade de bens infungíveis. Revista Cereus, v. 12, n. 3, p. 116-126, 2020.

GUPTA, Sourav Sen. Blockchain. John Wiley& Sons, Inc, 2017.

HESTERBERG, Tim. Bootstrap. WileyInterdisciplinaryReviews: ComputationalStatistics, v. 3, n. 6, p. 497-526, 2011.

HUREL, Louise Marie; LOBATO, Luisa Cruz. Segurança e privacidade para a Internet das Coisas. 2018.

KANG, Do-Hun et al. Room temperature control and fire alarm/suppression IoT service using MQTT on AWS. In: 2017 International Conference on Platform Technology and Service (PlatCon). IEEE, 2017. p. 1-5.

KANG, Do-Hun et al. Roomtemperaturecontrol and firealarm/suppression IoT serviceusing MQTT on AWS. In: 2017 International Conference on Platform Technology and Service (PlatCon). IEEE, 2017. p. 1-5.

KANNISTO, Juhaet al. Precision time protocolprototype on wireless LAN. In: International Conference on Telecommunications. Springer, Berlin, Heidelberg, 2004. p. 1236-1245.

KARAKUS, Murat; DURRESI, Arjan. Quality of service (QoS) in software defined networking (SDN): A survey. Journal of Network and Computer Applications, v. 80, p. 200-218, 2017.

KASHYAP, Monika; SHARMA, Vidushi; GUPTA, Neeti. Taking MQTT and NodeMcu to IOT: Communication in Internet of Things. Procedia Computer Science, v. 132, p. 1611-1618, 2018.

KHAN, Minhaj Ahmad; SALAH, Khaled. IoT security: Review, blockchain solutions, and open challenges. Future Generation Computer Systems, v. 82, p. 395-411, 2018.

KIM, Tai-hoon; RAMOS, Carlos; MOHAMMED, Sabah. Smartcity and IoT. 2017.

KOŠŤÁL, Kristiánet al. Management and Monitoring of IoT Devices Using Blockchain. Sensors, v. 19, n. 4, p. 856, 2019.

KSHETRI, Nir; VOAS, Jeffrey. Blockchain-enabled e-voting. IEEE Software, v. 35, n. 4, p. 95-99, 2018.

LAGHARI, Asif Ali et al. Assessment of quality of experience (QoE) of imagecompression in social cloudcomputing. Multiagent and Grid Systems, v. 14, n. 2, p. 125-143, 2018.

LAPPALAINEN, Sami; KOBAYASHI, Takashi. A PatternLanguage for MVC Derivatives. https://goo. gl/dqGTmg>. Acessado em, v. 10, p. 1-5, 2017.

LAVINA, Maria Eduarda. Validação do uso da tecnologia Blockchain para o tráfego seguro de dados na área da saúde. Gestão da Segurança da Informação-Unisul Virtual, 2018.

LAVINA, Maria Eduarda. Validação do uso da tecnologia Blockchain para o tráfego seguro de dados na área da saúde. Gestão da Segurança da Informação-Unisul Virtual, 2018.

LEAL, Rodrigo Lima Verde. Aplicações e iniciativas.

LINARES-BARBERO, Manuela. Trazabilidadcon Blockchain. 2019.

LOPER, Adriane Aparecida. Conceitos e características dos sistemas distribuídos. Projetos de redes e sistemas distribuídos, p. 9.

LYCETT, Mark; RADWAN, Omar. Developing a quality of experience (QoE) model for web applications. Information Systems Journal, v. 29, n. 1, p. 175-199, 2019.

MACHADO, Caciano; FRÖHLICH, Antônio Augusto Medeiros. IoT data integrity verification for cyber-physical systems using blockchain. In: 2018 IEEE 21st International Symposium on Real-Time Distributed Computing (ISORC). IEEE, 2018. p. 83-90.

MADAKAM, Somayyaet al. Internet of Things (IoT): A literaturereview. Journal of Computer and Communications, v. 3, n. 05, p. 164, 2015.

MALISUWAN, Settapong; MILINDAVANIJ, Dithdanai; KAEWPHANUEKRUNGSI, Wassana. Quality of service (QoS) and quality of experience (QoE) of the 4G LTE perspective. International Journal of Future Computer and Communication, v. 5, n. 3, p. 158, 2016.

MCROBERTS, Michael. Arduino básico. Novatec Editora, 2018.

MENDES, Renann. © Profissional de E-commerce, 2016. Disponível em: <https://www.profissionaldeecommerce.com.br/2-segundos-de-carregamento-dos-sites/> Acesso 1/09/2020.

MIERS, Charles C. et al. Análise de mecanismos para consenso distribuído aplicados a Blockchain. 2019.

MORAIS, A. M. de; CALLOU, G. R. de A.; LINS, F. A. A. Simulação e avaliaçãode desempenho de uma rede blockchain utilizando containers docker. Cadernos doIME-Série Informática, v. 44, p. 73–87, 2020.

MOUGAYAR, William. Blockchain para negócios: promessa, prática e aplicação da nova tecnologia da internet. Alta Books Editora, 2018.

NETO, Valdemar V. Graciano; SANTOS, Rodrigo P.; ARAUJO, Renata. Sistemas de sistemas de informação e ecossistemas de software: Conceitos e aplicações. Tópicos em Sistemas de Informação: Minicursos SBSI, p. 22-41, 2017.

NOFER, Michael et al. Blockchain. Business &Information Systems Engineering, v. 59, n. 3, p. 183-187, 2017.

OLIVEIRA, Marcela T. et al. Uma Avaliação de Desempenho de Cadeias de Blocos Privadas Permissionadas através de Cargas de Trabalho Realísticas∗. In: Anais Principais do XVIII Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais. SBC, 2018. p. 309-322.

PANARELLO, Alfonso et al. Blockchain and iot integration: A systematicsurvey. Sensors, v. 18, n. 8, p. 2575, 2018.

PAUFERRO, Gabriel BrognoAlcantara; DE PAIVA, SeilaVasti Faria; LESSA, Nayari Marie. IoT: conceitos de segurança de dados e criptografia. Cogitare, v. 2, n. 3, p. 40-52, 2020.

PEREIRA, Eliseu Moura. Plataforma Industrial de Internet das Coisas. 2018.

PINNO, Otto JulioAhlert; GRÉGIO, André Ricardo Abed; DE BONA, Luis CE. Controlchain: Blockchain as a central enabler for accesscontrolauthorizations in the iot. In: GLOBECOM 2017-2017 IEEE Global Communications Conference. IEEE, 2017. p. 1-6.

PRATES, N. et al. Ameaças de segurança, defesas e análise de dados em iot baseada em sdn. Minicursos SBSeg, p. 1-50, 2018.

RAMOS, Ricardo Argentonet al. Reuso da implementaçao orientada a aspectos do padrão de projeto camada de persistência. In: The FourthLatin American Conference on PatternLanguages of Programming-SugarLoafPLoP, Fortaleza-CE. 2004.

RAY, ParthaPratim. A survey on Internet of Things architectures. Journal of King SaudUniversity-Computer and InformationSciences, v. 30, n. 3, p. 291-319, 2018.

RICHARDSON, Matt; WALLACE, Shawn. Gettingstarted with raspberry PI. " O'Reilly Media, Inc.", 2012.

RODRIGUES, Carlo Kleber da Silva. Uma análise simples de eficiência e segurança da tecnologia Blockchain. 2018.

RODRIGUEZ, Luis Gustavo Araujo. Arquitetura baseada em camadas e métricas de desempenho para sistemas móveis: uma proposta. 2017. Dissertação de Mestrado. Universidade Estadual de Maringá.

ROTERMEL, Felipe; SOMMARIVA, Leonardo W. Inovações advindas na nova versão da linguagem de programação web PHP 7.0. Revista Interdisciplinar Científica Aplicada, v. 10, n. 4, p. 1-20, 2016.

SARZOSA CHOQUE, Ana Isabel. PERFORMANCE TESTING A WEB SERVICES CON JMETER. 2019. Tese de Doutorado.

SHIELD, Arduino Ethernet. Arduino Ethernet Shield. Recuperado el, 2016.

SILVA, Matheus Passos. A segurança da democracia e a blockchain. Projeção, Direito e Sociedade, v. 9, n. 1, p. 119-138, 2018.

SIMIĆ, Miloš; SLADIĆ, Goran; MILOSAVLJEVIĆ, Branko. A Case Study IoT and Blockchain poweredHealthcare. Acesso em: 10 deSetembro de 2018.

SINGH, Manisha; BARANWAL, Gaurav. Quality of service (qos) in internet of things. In: 2018 3rd International Conference on Internet of Things: SmartInnovation and Usages (IoT-SIU). IEEE, 2018. p. 1-6.

ŠKRABA, Andrejet al. Prototype of groupheart rate monitoring with NODEMCU ESP8266. In: 2017 6th MediterraneanConference on EmbeddedComputing (MECO). IEEE, 2017. p. 1-4.

SONI, Dipa; MAKWANA, Ashwin. A survey on mqtt: a protocol of internet of things (iot). In: International Conference On Telecommunication, Power Analysis And ComputingTechniques (ICTPACT-2017). 2017.

SONI, Dipa; MAKWANA, Ashwin. A survey on mqtt: a protocol of internet of things (iot). In: International Conference On Telecommunication, Power Analysis And Computing Techniques (ICTPACT-2017). 2017.

TEIXEIRA, Márcio Andrey; CATANDUVA, S. P. Servidor WEB Apache. 2018.

TORRES, Andrei BB; ROCHA, Atslands R.; DE SOUZA, José Neuman. Análise de desempenho de brokers mqtt em sistema de baixo custo. In: Anais do XV Workshop em Desempenho de Sistemas Computacionais e de Comunicação. SBC, 2016. p. 47-58.

VUJOVIĆ, Vladimir; MAKSIMOVIĆ, Mirjana. Raspberry Pi as a Sensor Web node for home automation. Computers&ElectricalEngineering, v. 44, p. 153-171, 2015.

WU, Longfeiet al. Um esquema de autenticação out-of-band para internet de coisas usando a tecnologia blockchain. In: 2018 Conferência Internacional sobre Computação, Redes e Comunicações (ICNC) . IEEE, 2018. p. 769-773.

YU, Shitanget al. A high performance blockchain platform for intelligent devices. In: 2018 1st IEEE International Conference on Hot Information-Centric Networking (HotICN). IEEE, 2018. p. 260-261.