Judson dos Santos Costa

Arquitetura de um sistema para automação residencial baseado em IoT

Natal – RN

Dezembro de 2022

Judson dos Santos Costa

Arquitetura de um sistema para automação residencial baseado em IoT

Trabalho de Conclusão de Curso de Engenharia de Computação da Universidade Federal do Rio Grande do Norte, apresentado como requisito para a obtenção do grau de Bacharel em Engenharia de Computação

Orientador: Prof. Dr. Diego R. C. Silva

Universidade Federal do Rio Grande do Norte – UFRN

Departamento de Engenharia de Computação e Automação – DCA

Curso de Engenharia de Computação

Natal – RN

Dezembro de 2022

Universidade Federal do Rio Grande do Norte - UFRN
Sistema de Bibliotecas - SISBI
Catalogação de Publicação na Fonte. UFRN - Biblioteca Central Zila Mamede

Costa, Judson Dos Santos.
Arquitetura de um sistema para automação residencial baseado em IoT / Judson Dos Santos Costa. - Natal, 2022.
34f.: il.

Monografia (graduação) - Universidade Federal do Rio Grande do Norte, Centro de Tecnologia, Engenharia de Computação. Natal, RN, 2022.
Orientador: Prof. Dr. Diego R. C. Silva.

1. Automação residencial - Monografia. 2. Domótica - Monografia. 3. Arquitetura para Internet das coisas - Monografia. I. Silva, Prof. Dr. Diego R. C. II. Título.

RN/UF/BCZM CDU 004

Elaborado por MARJORIE ROSIELLE SILVA DO AMARAL - CRB-15/353

AGRADECIMENTOS

Aos meus pais, por sempre acreditarem no meu sucesso e me incentivarem a ser resiliente na vida.

Ao meu orientador, Diego Silva, pelo acompanhamento, contribuições e por ser prestativo e auxiliar na elaboração desse projeto.

Aos professores e alunos que fazem parte do LII (Laboratório de Informática Industrial) por compartilharem comigo todo o conhecimento e experiência aprendido até aqui.

Por fim, aos amigos do curso que diretamente ou indiretamente estiveram comigo, oferecendo apoio em cada momento de estudo e compartilhando vitórias em cada desafio alcançado durante todos esses anos.

"Quem não programa, será programado."
(Silvio Meira)

RESUMO

O presente trabalho descreve, implementa e valida a arquitetura de um sistema para automação residencial baseado em IoT (Internet das Coisas). A arquitetura faz uso de tecnologias que permitem a coleta e o armazenamento de dados medidos através de sensores dos dispositivos que estão no ambiente residencial, além de possibilitar o gerenciamento e acionamento desses dispositivos inteligentes. A partir da visualização dos dados coletados e do gerenciamento dos dispositivos, é possível obter estratégias para melhorar a eficiência dos recursos energéticos do domicílio. Ao longo do trabalho são apresentadas algumas soluções populares presentes no mercado acessíveis ao publico assim como as tecnologias utilizadas e funcionalidades disponíveis. Em seguida é descrito o modelo de divisão em camadas da arquitetura e todas as tecnologias utilizadas em cada uma delas. Por fim são mostrados os resultados obtidos com a implementação e utilização da arquitetura proposta em diversos ambientes para validar alguns recursos, como a possibilidade de utilização em lugares que não possuem acesso à internet.

Palavras-chaves: Automação residencial. Domótica. Arquitetura para Internet das coisas.

ABSTRACT

The present work describes, implements, and validates the architecture of a system for IoT home automation. The architecture makes use of technologies that allow the collection and storage of measured data through sensors of devices in the residential environment, in addition to enabling the management and activation of these smart devices. By processing the data collected by a device, it is possible to obtain strategies to improve the efficiency of the energy resources of the house. Throughout the work, some popular solutions present in the market accessible to the public are presented, as well as the technologies used and available functionalities. Next, the model of division into layers of the architecture and all the technologies used are described. Finally, the results obtained with the implementation and use of the proposed architecture in different environments are shown to validate some features, such as the possibility of using it in places that do not have internet access.

Keywords: Home automation. Domotics. IoT Architecture.

LISTA DE ILUSTRAÇÕES

LISTA DE ABREVIATURAS E SIGLAS

API	*Application Programming Interface*
CSS	*Cascading Style Sheets*
DOM	*Document Object Model*
HTML	*HyperText Markup Language*
HTTP	*HyperText Transfer Protocol*
HTTPS	*HyperText Transfer Protocol Secure*
IoT	*Internet of Things*
I/O	*Input/Output*
JSON	*JavaScript Object Notation*
JWT	*JSON Web Token*
LII	*Laboratório de Informática Industrial*
MCU	*Microcontrolador*
MQTT	*Message Queuing Telemetry Transport*
RAM	*Random-Acess Memory*
REST	*Representational State Transfer*
SGBD	*Sistema de Gerenciamento de Banco de Dados*
SSL	*Secure Sockets Layer*
TCP	*Transmition Control Protocol*
TLS	*Transport Layer Security*
UFRN	*Universidade Federal do Rio Grande do Norte*
UI	*User Interface*
VDOM	*Virtual DOM*
WWW	*World-Wide Web*

SUMÁRIO

1 INTRODUÇÃO

Com o advento da Internet das Coisas ou IoT (do inglês *Internet of Things*) observa-se uma tendência da miniaturização e popularização de componentes eletrônicos providos de interfaces de comunicação sem fio que, por sua vez, torna viável uma grande variedade de aplicações sem precedentes no cotidiano da sociedade. Aplicações que revolucionam a comunicação entre pessoas, máquinas, objetos e o ambiente em que estão inseridos, além de permitir a criação de novas possibilidades de gerenciamento dos ambientes sociais como cidades, hospitais, indústrias e residências (HERMANN; PENTEK; OTTO, 2016).

A automação é uma peça chave na modernização de prédios e na melhoria da eficiência na utilização dos recursos naturais. Além do âmbito ambiental, a automação também é muito importante na área de segurança pública ou privada. Vale destacar que, neste caso, a segurança se refere a segurança por meio de monitoramento, alarme e proteção física (AMARANTE; VASCONCELOS; NETO, 2021).

Em um ambiente automatizado, podem existir sensores e atuadores capazes de medir e controlar indicadores de conforto e consumo de recursos. Cada dispositivo (sensor ou atuador) contém um hardware eletrônico capaz de processar, enviar e receber informações. De maneira geral, esses dispositivos inteligentes são chamados de ativos e são fundamentais na composição de um sistema de automação.

Portanto a automação possibilita a integração de sistemas com os mais variados parâmetros do local que podem ser obtidos por sensores, por exemplo: sensores de presença, de temperatura, de abertura de portas e janelas, etc. Juntamente com o sensoriamento do ambiente, também possibilita o controle remoto de todos os dispositivos disponíveis a fim de realizar alguma ação que aumente a segurança física do local, como ativar um alarme, fechamento de portas, acionamento de lâmpadas, entre outros.

1.1 Trabalhos Relacionados

Foram realizadas pesquisas bibliográficas em busca de artigos que tivessem contextos e objetivos em comum com a proposta deste trabalho. A partir disso, pode-se analisar as tecnologias e funcionalidades implementadas nos trabalhos encontrados na literatura.

Nas pesquisas foram encontradas diversas arquiteturas para IoT, sendo várias utilizadas por diversas empresas. Segundo Hejazi (HEJAZI et al., 2018), observa-se que plataformas abertas (*open-source*) quando comparadas com plataformas proprietárias, podem ser mais promissoras por possibilitar a integração mais rápida de novas funcio-

nalidades e dispositivos. Também observa-se um padrão que os sistemas se comportam como *web services*, pois a maioria das plataformas analisadas disponibilizam acesso a API REST e utilizam tecnologias web. Por último, pode-se concluir que a visualização dos dados em tempo real é um recurso básico em uma plataforma IoT pelo fato da maioria das plataformas pesquisadas fornecerem esse recurso mesmo que seja de forma simples. Entretanto, nota-se que ainda possuem poucas funcionalidades disponíveis para análise de dados, como *big data*. Pode-se exemplificar as arquiteturas (GIRAU; NITTI; ATZORI, 2013) e (LEU; CHEN; HSU, 2014) que utilizam somente o protocolo HTTP, sendo que a segunda utiliza a técnica de *polling*, isto é, requisita periodicamente os dados.

1.2 Motivação

A internet das coisas possibilita a utilização de diversos recursos que podem mudar como as pessoas interagem com os objetos, como foi dito anteriormente. A seguir são mostrados alguns itens que representam o impacto na vida da sociedade.

- Permitir um estilo de vida com mais comodidade;
- Maior controle dos gastos e recursos;
- Monitoramento e controle dos dispositivos remotamente;

No entanto, ao analisar o cenário nacional, esses benefícios não estão muito acessíveis para a maioria da população por diversos motivos, como custo de aquisição de dispositivos inteligentes, falta de infraestrutura de acesso à internet, entre outros. Portanto, pode-se observar a necessidade da criação ou adaptação das soluções de automação residencial para serem utilizadas no contexto da população em geral, ou seja, tornar essas soluções e dispositivos mais acessíveis e resilientes a problemas de infraestrutura.

1.3 Objetivos

O objetivo deste trabalho é apresentar, implementar e validar uma arquitetura IoT para automação residencial baseada em tecnologias abertas que possibilite a comunicação entre dispositivos e usuários de tal forma que possa ser utilizada em ambientes residenciais ou em contextos maiores, como universidades ou cidades. A seguir pode-se observar algumas características que a arquitetura deve apresentar.

- Utilizar apenas tecnologias abertas e resultar em uma proposta aberta;
- Reaproveitar a infraestrutura de rede já disponível;

- Permitir facilmente a integração de novos tipos de dispositivos;
- Multiplataforma;
- Fluidez entre a interface gráfica e os dispositivos;
- Segurança das informações;
- Armazenamento de grandes quantidades de informações;
- Funcionamento local não dependente da nuvem;
- Permitir a automatização de procedimentos;
- Baixa latência entre os componentes (melhor experiência do usuário);
- Permitir notificações do servidor para os clientes;
- Estrutura escalável para suportar um possível aumento na demanda.

1.4 Estrutura do Trabalho

Este trabalho apresenta uma introdução sobre Internet das coisas e arquiteturas para automação residencial no Capítulo 1. Em sequência, o Capítulo 2 apresenta uma fundamentação teórica dos principais conceitos e tecnologias utilizadas para uma melhor contextualização do estudo apresentado. O Capítulo 3, por sua vez, aborda uma ideia superficial sobre a arquitetura proposta, explicando as principais funções de cada componente e seus relacionamentos dentro da arquitetura, enquanto o Capítulo 4 trata dos resultados obtidos a partir da implementação baseada na arquitetura proposta, apresentando todas as tecnologias utilizadas. E por fim, o Capítulo 5 mostra as principais conclusões deste trabalho, no qual são retomadas as ideias principais apresentadas e os objetivos alcançados.

2 REFERENCIAL TEÓRICO

Esse capítulo é dedicado a apresentar os principais conceitos e tecnologias no qual se baseia o projeto. Todo o processo de desenvolvimento da arquitetura proposta é fundamentada na utilização de um conjunto de tecnologias e técnicas que tornam o sistema mais robusto e versátil como um todo. Para facilitar a compreensão, primeiramente serão abordados as tecnologias da camada de apresentação (*frontend*) na seção 2.1, em seguida, os tópicos da camada de acesso (*backend*) na seção 2.2.

2.1 *Camada de apresentação*

A camada de apresentação também pode ser chamada de *frontend* ou *client-side*. Esta camada é composta pela *User Interface* (UI) que permite a interação do usuário diretamente com a interface visual da aplicação, ou seja, visualizar os dados disponíveis e clicar nos elementos visuais para realizar alguma ação no sistema. As interfaces de usuário são construídas utilizando HTML, CSS e *JavaScript*. Mais detalhes dessas tecnologias são descritos a seguir.

2.1.1 HTML e CSS

HyperText Markup Language (HTML) é uma linguagem de marcação utilizada para estruturar o conteúdo das páginas web. Ela consiste de uma série de elementos que determinam como incluir e/ou agrupar diferentes partes do conteúdo e definem a maneira como eles são exibidos na visualização de uma interface. A interface é um conjunto de componentes interativos, por exemplo, menus, botões, barras de pesquisa, tabelas, imagens, entre outros.

Cascading Style Sheets (CSS) é uma linguagem declarativa utilizada no controle da aparência de páginas web nos navegadores. As descrições de estilo feitas no CSS definem as propriedades e valores que determinam o comportamento de um componente quando é exibido pelo navegador. Isso faz com que o navegador exiba os componentes da interface corretamente nas posições desejadas.

2.1.2 JavaScript

JavaScript é uma linguagem de programação criada inicialmente para executar programas nos navegadores (*browsers*), mas também é possível utilizá-la em servidores através do *Node.js*. Por ter um fluxo de execução assíncrono que permite a execução de

operações paralelamente, tornou possível o desenvolvimento de aplicações web dinâmicas. Além disso, a grande quantidade de bibliotecas e pacotes de código aberto disponíveis impacta na velocidade da criação de aplicações web, como acesso às funções dos dispositivos através do navegador, por exemplo, câmera, geolocalização, notificações, etc.

2.1.3 React.js

O *React.js* é uma biblioteca *JavaScript* utilizada no desenvolvimento de interfaces de usuário em sites e aplicações, sejam elas simples ou complexas. Ele renderiza apenas os componentes certos que tiveram alguma alteração nos seus dados, assim, torna eficiente a dinâmica do fluxo de dados na interface de usuário (UI). Isso é possível por causa de um componente chamado *Virtual DOM* (VDOM) que é a abstração mais rápida e leve do *Document Object Model* (DOM) do *browser*. Quando não se utiliza o VDOM, o *browser* precisa renderizar todos os componentes da UI sempre que houver alguma alteração nos dados da interface, necessitando de maior poder de processamento e mais tempo de carregamento. Isso afeta diretamente a usabilidade da aplicação, principalmente se ela não for um site simples e estático.

2.2 *Camada de acesso*

A camada de acesso, também chamada de *backend* ou *server-side*, é onde pode-se encontrar toda a lógica de manipulação dos dados, regras de negócio e o armazenamento das informações em banco de dados de uma aplicação. A seguir são mostradas as várias tecnologias e conceitos utilizados na implementação da arquitetura proposta.

2.2.1 API

A sigla API é derivada da expressão em inglês *Application Programming Interface* que trata de uma série de padrões e protocolos que permite a comunicação com outras aplicações, ou seja, possibilita a interoperabilidade entre plataformas. Quando uma aplicação utiliza uma API para se comunicar com outro sistema, ela precisa apenas saber as interfaces das funcionalidades desse sistema, assim são abstraídos todos os detalhes da implementação do software. Isso pode ser visto na Figura 1 onde mostra-se um cenário com a comunicação de um *website*, um app nativo e demais APIs com a API central. Ou seja, com o uso de APIs é possível a comunicação entre diversos tipos de sistemas, desde *websites*, passando pelo ecossistema *mobile* e outros sistemas *server-side* com uma API que atende a todas essas requisições.

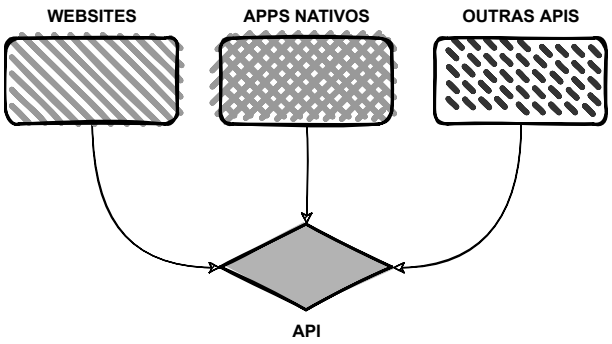

Figura 1 – Comunicação de aplicações com uma API

2.2.2 JSON

JavaScript Object Notation (JSON) é um formato de texto criado para representar estruturas de dados. Apesar de ter o *JavaScript* no nome por derivar do seu objeto literal, o JSON não pertence a nenhuma linguagem de programação e é suportado por diversas delas. O objetivo do JSON é ser uma notação simples e com pouca verbosidade com o intuito de facilitar a leitura e escrita para humanos e também para as máquinas (CROCKFORD, 2006). A Figura 2 mostra um exemplo de um objeto JSON, pode-se observar que ele é composto por uma coleção de pares chave/valor, ou seja, para cada valor, tem-se uma chave identificadora.

```
{
    "id": 120938,
    "nome": "nome do componente",
    "data": "2022-12-07T22:20:53.964Z",
    "itens": []
}
```

Figura 2 – Exemplo de um objeto em JSON.

2.2.3 Node.js

O *Node.js* é um *framework* que torna possível a execução de códigos *JavaScript* sem a necessidade de um navegador. Com isso, pode-se criar desde pequenos programas para manipulação de poucos dados até servidores web altamente escaláveis. Isso é possível porque o *Node.js* apresenta algumas características como fluxo assíncrono que permite receber múltiplas conexões e processá-las simultaneamente, além de leitura e gravação de dados de forma não bloqueante e diversos outros recursos para a construção de aplicações de alta performance (TILKOV; VINOSKI, 2010).

2.2.4 HTTP

HyperText Transfer Protocol (HTTP) foi criado em 1990 e desde então é um protocolo amplamente utilizado, sendo a base da *World-Wide Web* (WWW) (FIELDING et al., 1999). Tem como padrão de troca de mensagens o formato cliente-servidor, ou seja, o pedido é iniciado pelo cliente, geralmente por meio de um *browser*, e o servidor envia uma resposta para esse pedido. Com isso, a cada troca de mensagem entre eles, é necessário fazer uma requisição e uma resposta. Para utilizar em cenários que se tem uma grande quantidade de clientes e requisições em tempo-real, como em alguns contextos de IoT, pode resultar em sobrecarregamento da aplicação (YOKOTANI; SASAKI, 2016).

Com as evoluções tecnológicas, o protocolo também recebeu otimizações com a versão 2 (HTTP/2) para oferecer algumas melhorias como compressão de dados e maior eficiência no uso de rede (BELSHE; THOMSON; PEON, 2015). Também pode-se adicionar uma camada de segurança com o uso do SSL/TLS para obter o protocolo HTTPS (RESCORLA, 2000) que fornece criptografia na conexão cliente-servidor, dificultando a visualização por terceiros dos dados transferidos.

2.2.5 WebSocket

Assim como o *socket* possibilita que aplicações se comuniquem por meio de requisições de outros computadores ou troquem mensagens entre si mesmas, o protocolo *WebSocket* também permite essa troca de informações, mas como o nome diz, essa funcionalidade é disponibilizada no ecossistema web. Portanto, o *WebSocket* fornece um canal de comunicação bidirecional entre aplicações web e serviços através de uma conexão TCP (*Transmission Control Protocol*) (FETTE; MELNIKOV, 2011). Ou seja, as aplicações podem enviar requisições de forma assíncrona de uma para a outra, diferente do paradigma *request-response* utilizado pelo HTTP que obriga que a aplicação cliente apenas possa receber informações se explicitamente solicitar ao serviço web. Outro ponto é a possibilidade de enviar requisições através de uma mesma conexão por permanecer aberta durante os envios.

2.2.6 MQTT

O *Message Queuing Telemetry Transport* (MQTT) é um protocolo de sistema de mensagens assíncrono que utiliza um modelo de publicação e assinatura que permite a distribuição de mensagens um-para-muitos. Esse modelo funciona inicialmente com a conexão do cliente com o servidor, em seguida o cliente pode definir o tópico que queira fazer a assinatura. Após isso, toda mensagem que o servidor receber neste tópico será enviada para os clientes que assinaram esse tópico.

Em ambientes de alto desempenho com grande quantidade de requisições assíncronas, o MQTT apresenta um melhor desempenho se comparado com o HTTP (YOKOTANI; SASAKI, 2016) por apresentar características que beneficiam o seu uso nessas circunstâncias, como uso melhor da rede de dados, possuir níveis de confiabilidade que garantem o efetivo envio da mensagem e, assim como o *WebSocket*, permanecer com a conexão aberta durante o envio das requisições.

2.2.7 MySQL

MySQL é um Sistema de Gerenciamento de Banco de Dados (SGBD) relacional com armazenamento em disco. Segundo (SOUZA; SANTOS, 2015), o *MySQL* apresenta um bom desempenho na indexação de dados, ou seja, suporta a leitura de uma grande quantidade de dados. No entanto, se comparado com os bancos não-relacionais, nos cenários de aplicações com grande volume de dados e necessidade de escalar com facilidade e eficiência, o *MySQL* não possibilita isso devido não fornecer o escalonamento horizontal.

2.2.8 MongoDB

MongoDB é um banco de dados não-relacionais (NoSQL) de propósito geral voltado para aplicações web e armazenamento dos dados baseado em documentos semelhante ao JSON. A estrutura de dados pode ter uma formatação flexível, ou seja, os campos da estrutura podem ser alterados em qualquer momento e entre os documentos salvos também, bem diferente do padrão utilizado nos SGBDs relacionais. Por fim, o *MongoDB* é um banco de dados distribuído com alta disponibilidade e escalonamento horizontal adequado para aplicações com alto fluxo de dados e que precisam salvar objetos com estruturas que podem variar ao longo do tempo.

2.2.9 Redis

O *Redis* é um banco de dados não relacional que armazena os dados em memória RAM (*Random-Acess Memory*) e apresenta um alto desempenho de leitura e escrita se comparado aos SGBDs de armazenamento em disco (SOUZA; SANTOS, 2015). Os dados são armazenados em memória RAM e periodicamente são salvos em disco no padrão de chave-valor. Essas são características interessantes para serem utilizadas na persistência de objetos simples.

3 METODOLOGIA

Neste capítulo é descrita a arquitetura proposta para automação residencial e todos os detalhes dos componentes que fazem parte dela. Inicialmente é apresentada uma visão superficial sobre a arquitetura e todas as camadas que a compõem. Em seguida são abordados os detalhes de cada camada, suas principais funções e como ela se relaciona com os demais componentes.

3.1 Arquitetura

Pode-se definir a arquitetura proposta como um conjunto de camadas que se relacionam por meio dos seus componentes com o objetivo de permitir que um usuário localizado em qualquer lugar possa interagir com a residência desejada a partir de uma conexão com a internet e acesso as interfaces de usuário disponíveis para controlar os dispositivos inteligentes e/ou visualizar os dados captados pelos sensores no local escolhido.

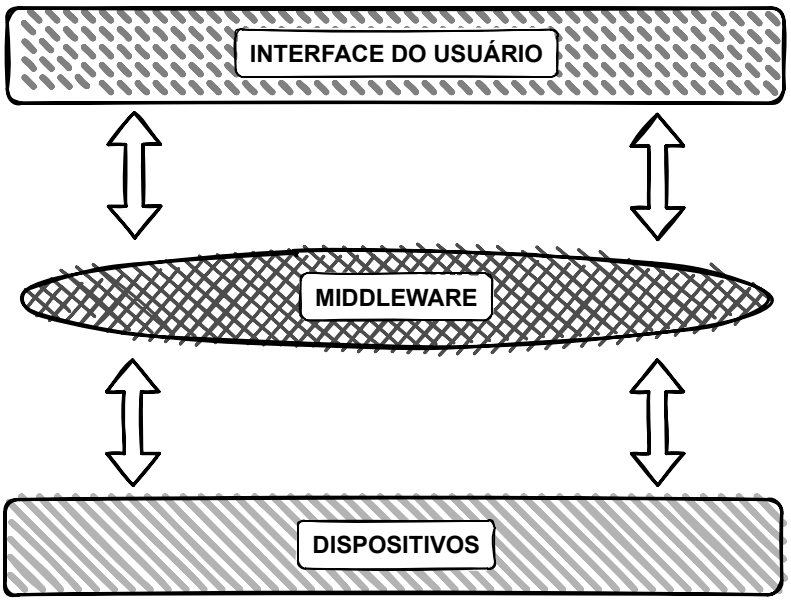

Figura 3 – Estrutura da arquitetura

Como pode ser visto na Figura 3, a estrutura da arquitetura resume-se a três camadas. Iniciando pelas camadas externas, a camada no topo constitui a interface de usuário, ou *user interface*, que permite que o usuário consiga fazer interações com as funcionalidades do sistema. Na camada inferior encontram-se os dispositivos físicos. Pode-se dividir os dispositivos nos grupos de sensores e atuadores. E por último na parte central da imagem, localiza-se a camada da unidade central, chamada de *middleware*, que concentra

o núcleo da arquitetura, ou seja, reúne grande parte das funcionalidades propostas pela arquitetura e que são descritas nas próximas seções.

3.1.1 Interface do usuário

A interface de usuário é a camada onde encontram-se todas as tecnologias que tem como objetivo prover ao usuário maneiras de utilizar as funcionalidades do sistema. Esta camada permite ter diversas aplicações que o usuário possa utilizar, por exemplo, um aplicativo nativo para *smartphone* e um *website* para ser acessado pelo computador. No contexto de automação residencial, pode-se ter várias funções como a opção de gerenciamento de todos os dispositivos inteligentes cadastrados (visualizar, cadastrar, editar, excluir), verificar o consumo medido num momento específico ou acumulado por cada aparelho, entre outras.

3.1.2 *Middleware*

O *middleware* faz a ligação das interfaces acessadas pelos usuários com os dispositivos inteligentes através de uma conexão de rede.

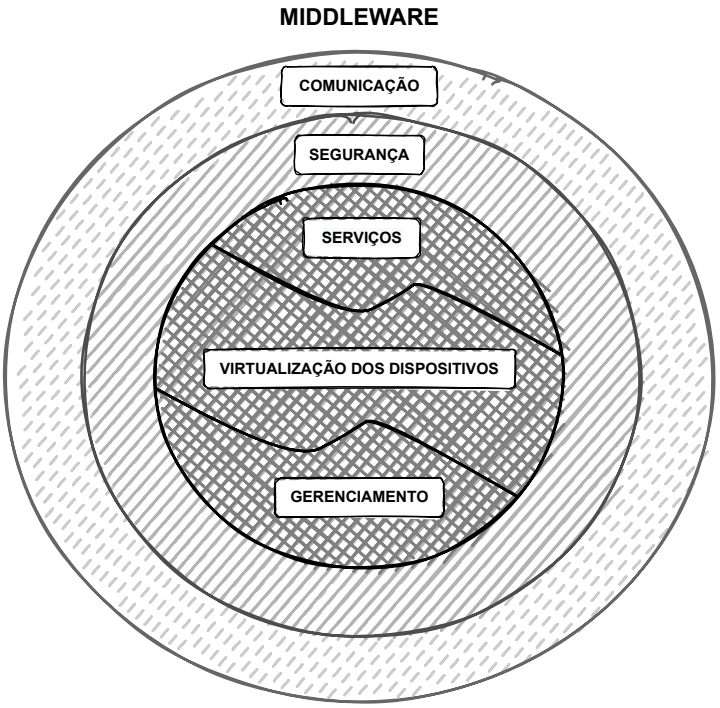

Figura 4 – Estrutura do *middleware*

Como é mostrado na Figura 4, o *middleware* é composto por cinco módulos, são eles: comunicação, segurança, serviços, virtualização de dispositivos e gerenciamento. A arquitetura permite a flexibilidade dos módulos do *middleware*, com isso, pode-se adicionar novos protocolos e funcionalidades.

3.1.2.1 Comunicação

O módulo de comunicação possibilita que os dispositivos utilizem qualquer protocolo suportado pelo *middleware*, com isso, eles podem se conectar por meio do protocolo que melhor se adapta ao contexto em que está inserido. Este módulo permite que dispositivos utilizando protocolos distintos possam se comunicar, assim como o usuário que acessa a interface possa controlar um dispositivo independente do protocolo que o equipamento esteja conectado.

3.1.2.2 Segurança

Este módulo tem como responsabilidade garantir a segurança entre as trocas de informações dos usuários e dispositivos. Para isso, é gerenciada a autenticação tanto de dispositivos quanto de clientes conectados à interface gráfica. Portanto, o módulo de segurança garante que as conexões entre o usuário e dispositivos sejam seguras.

3.1.2.3 Serviços

O módulo de serviços contém quaisquer serviços fornecidos pela arquitetura. As funcionalidades podem ser de visualização dos dados gerados pelos dispositivos; controlar as opções de atuadores disponíveis nos dispositivos conectados; configuração de automação de tarefas para quando algum cenário for satisfeito, a ação seja realizada; entre outros.

3.1.2.4 Virtualização dos dispositivos

O módulo de virtualização dos dispositivos tem o objetivo de criar uma abstração para cada um dos dispositivos reais cadastrados na aplicação. Por meio da abstração do dispositivo físico, é possível estabelecer uma relação bidirecional em tempo real entre o estado do dispositivo virtual e físico. A virtualização de um dispositivo pode ser modificada através de dois tipos de interação descritos a seguir.

- Dispositivo real - dispositivo virtual: As mudanças de estado que são reportadas pelos dispositivos reais para o sistema são aplicadas ao dispositivo virtual;
- Dispositivo virtual - dispositivo real: As mudanças de estado que o dispositivo virtual sofre são enviadas ao dispositivo real.

3.1.2.5 Gerenciamento

Esse módulo possibilita a visualização de dados, atualização das configurações e remoção dos dispositivos. Este módulo também é responsável por gerenciar os erros que podem ser utilizados em relatórios e pelo gerenciamento das conexões dos dispositivos que utilizam protocolos que permanecem com a conexão aberta, como o MQTT e *WebSocket*.

Com isso, quando um dispositivo é desconectado, a conexão é usada para identificar o dispositivo para notificar o usuário que esse dispositivo foi desconectado.

3.1.3 Dispositivos

Os dispositivos são parte fundamental de um sistema de automação e, por isso, recebem o nome de ativos, porque é por meio deles que é possível interagir com um ambiente sem estar presente. Um dispositivo é composto de um hardware eletrônico que tem a capacidade de fazer processamento, envio e recebimento de informações. Ele pode ser composto de sensores e/ou atuadores. Quando um dispositivo encontra-se conectado a um *middleware*, ele pode receber comandos ou enviar dados para informar algum valor captado ou reportar erros gerados por alguma interferência externa.

4 RESULTADOS

Para ser possível a validação da arquitetura proposta, foi realizada a implementação dos componentes da arquitetura utilizando algumas tecnologias web. A seção 4.1 aborda os detalhes do sistema criado para validação, assim como os dispositivos desenvolvidos para obter os resultados através de alguns cenários de aplicação inserido no contexto de internet das coisas. Em seguida pode-se observar os cenários em que os testes foram feitos.

4.1 Implementação

No desenvolvimento da implementação foram escolhidas as tecnologias de acordo com as camadas da arquitetura proposta, como pode ser visto na Figura 5. As camadas e tecnologias são abordadas em mais detalhes nas próximas seções.

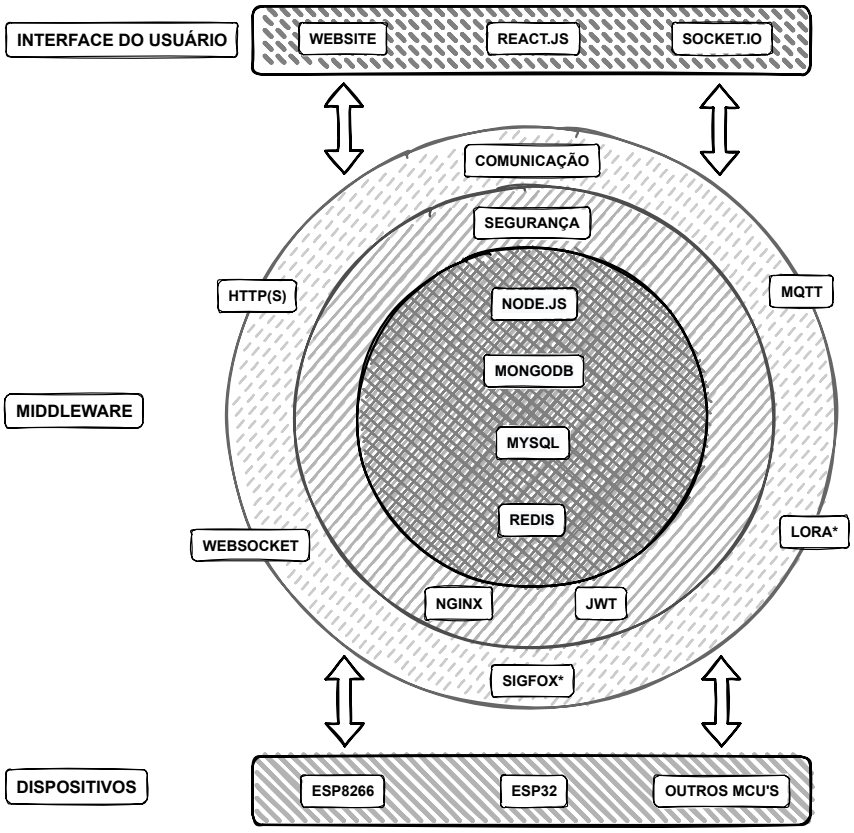

Figura 5 – Tecnologias utilizadas

A arquitetura implementada permite que o usuário que esteja no interior da residência, ou externamente, visualize os dados capturados pelos dispositivos, e controle os equipamentos inteligentes ao executar alguma ação que modifique o ambiente. As funcionalidades podem ser utilizadas pela interface web de usuário que está conectada ao

middleware. Por sua vez, o *middleware* recebe os comandos e envia para os dispositivos que estão disponíveis no local. Os dispositivos inteligentes se conectam ao *middleware* para enviar dados e para receber comandos realizados pelo usuário.

As tecnologias adotadas são do ecossistema web por apresentarem diversos benefícios como a possibilidade de enviar qualquer tipo de informação, além de ser difundida e de fácil implementação ao se comparar com a programação nativa para *desktop* ou *mobile*, por exemplo C/C++ ou Java. Além disso, pode-se dizer que uma das maiores vantagens de se utilizar tecnologias web é a possibilidade da aplicação ser multiplataforma, ou seja, ser acessada por um navegador independente se é um computador ou *smartphone*.

4.1.1 *Middleware*

O *middleware* é o principal componente desta arquitetura por ser o responsável em fazer a ligação entre o usuário e os dispositivos inteligentes. Além de permitir a criação de vários serviços que agregam o uso da automação residencial.

Por utilizar tecnologias web, o *middleware* foi implementado como um *web service* que recebe dados seguindo um formato pré-definido e retorna os dados solicitados, assim torna o sistema interoperável. Também foi escolhido o *JavaScript* (subseção 2.1.2) como linguagem de programação e o *Node.js* (subseção 2.2.3) como *framework*.

O *Node.js* é responsável por tratar todas as requisições recebidas pelo *middleware*, ou seja, toda a lógica de leitura e gravação de dados, a execução de regras, a comunicação com os bancos de dados dos serviços é feita por ele.

O *middleware* é composto de várias ferramentas e bibliotecas que funcionam integradas entre si. A seguir é descrito como foram implementados todos os módulos.

4.1.1.1 Comunicação

Para permitir a comunicação do *middleware* com a interface de usuário e os dispositivos, foi realizada a integração com alguns protocolos de comunicação que podem ser utilizados de acordo com o objetivo desejado. Os dispositivos podem se comunicar com o *middleware* por meio dos protocolos HTTP(S), *WebSocket* ou MQTT. Enquanto que a interface de usuário pode utilizar o HTTP(S) e *WebSocket*.

4.1.1.2 Segurança

Esse módulo implementa algumas formas de segurança na aplicação. Pode-se adicionar segurança por meio de criptografia na conexão cliente-servidor, como dito na subseção 2.2.4, com isso, os protocolos de comunicação disponíveis receberam essa camada para dificultar a visualização dos dados por terceiros. Também foi implementada

a autenticação de dispositivos e usuários para identificar e controlar o acesso desses componentes, como o armazenamento de senhas e gerenciamento de *tokens*.

4.1.1.3 Serviços

O módulo de serviços implementa uma variedade de funcionalidades que tentam facilitar o cotidiano do usuário. Exemplos de funcionalidades implementadas: visualização dos dados em forma de histórico (Figura 6); controle de dispositivos; gerenciamento de permissões e grupos de dispositivos; criação de regras para automatizar ações; e notificações para reportar algum acontecimento reconhecido por algum sensor de um dispositivo (Figura 7).

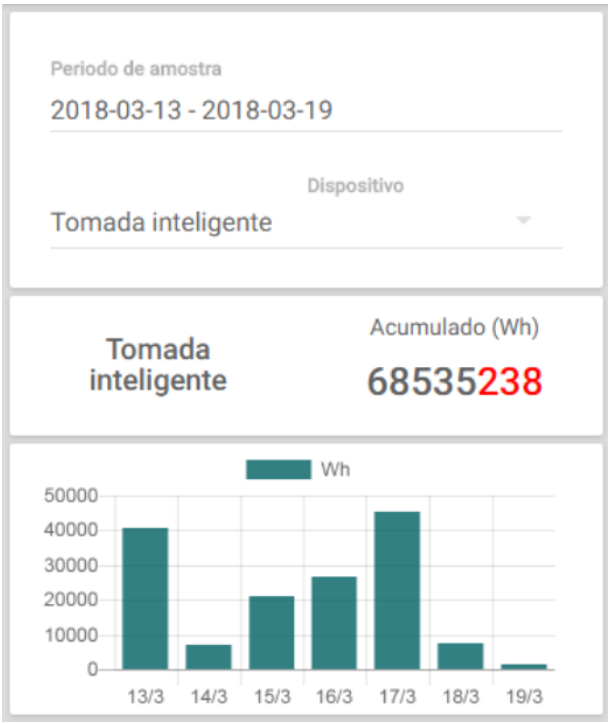

Figura 6 – Monitoramento do consumo de energia

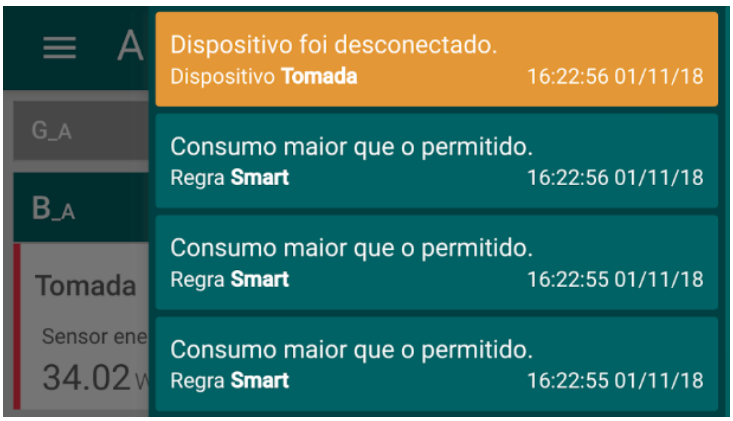

Figura 7 – Visualização de notificações

4.1.1.4 Virtualização dos dispositivos

Esse módulo gerencia a virtualização de cada dispositivo físico cadastrado na aplicação para permitir que as mudanças feitas pelo usuário no dispositivo virtual modifique o dispositivo real. A partir do cadastro de um dispositivo físico, é criado uma cópia no sistema com os estados para permitir a manipulação dessa abstração.

4.1.1.5 Gerenciamento

O gerenciamento de dispositivos foi implementado com o intuito de permitir que o usuário possa manipular os dispositivos cadastrados, como pode-se visualizar na Figura 8. Também torna possível a descoberta de novos dispositivos, ou seja, um dispositivo pode notificar ao *middleware* que ele está pronto para ser utilizado.

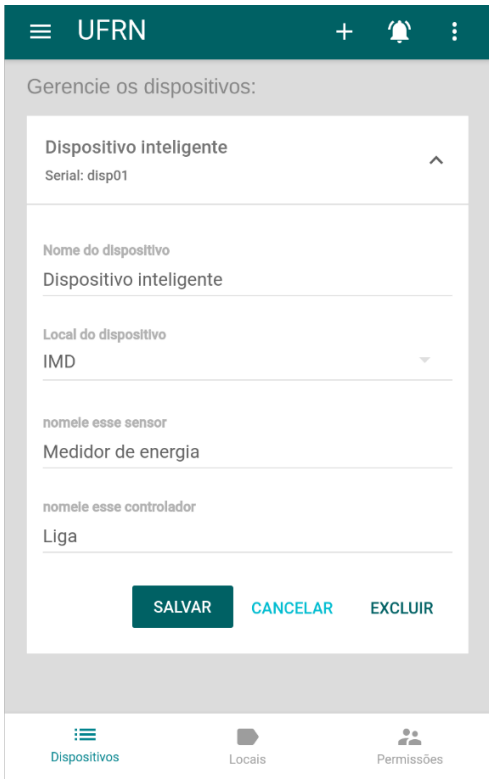

Figura 8 – Configuração de dispositivo

4.1.2 Interface do usuário

Existem várias linguagens de programação e *frameworks* utilizados no desenvolvimento de interfaces web, cada um tem pontos positivos e negativos e o melhor cenário apropriado para ser utilizado. Nesta implementação foi escolhido o *JavaScript* como linguagem de programação e o *React.js* como *framework* devido a fatores que já foram citados na subseção 2.1.2 e subseção 2.1.3 respectivamente.

Além disso, foi preciso adicionar alguns módulos para completar o desenvolvimento da aplicação. Os módulos utilizados são listados a seguir.

- React-router: controle das rotas e telas da interface;
- Redux: gerenciamento dos dados e componentes da aplicação;
- Socket.io-client: implementação do *WebSocket* para facilitar sua utilização;

A utilização de todos os módulos possibilita a interação do usuário com a interface web (Figura 9). Quando o usuário requisita uma alteração nos dispositivos, o respectivo componente pode solicitar ao servidor ou buscar nos dados já armazenados na aplicação. Caso seja solicitado ao servidor, é disparado um evento na conexão *WebSocket* que retorna os novos dados.

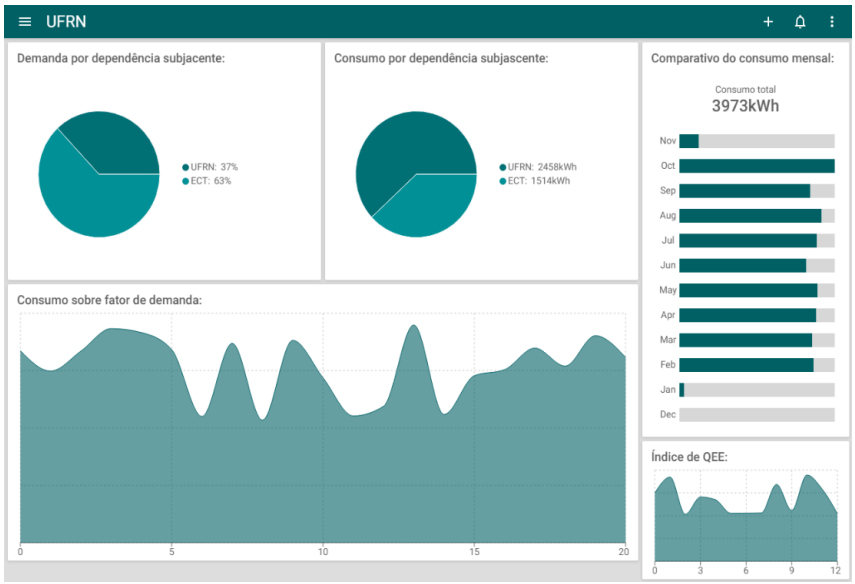

Figura 9 – Interface mostrando *dashboard*

4.1.3 Dispositivos

Foram realizadas pesquisas para encontrar um componente eletrônico que apresenta as propriedades compatíveis com as necessidades desta arquitetura. Por ser baseada em protocolos de comunicação web, utiliza-se microcontrolador (MCU) ESP8266 como base para o desenvolvimento dos dispositivos inteligentes utilizados na implementação. O ESP8266 suporta nativamente conexão *Wi-Fi* e os protocolos web, além dos padrões de segurança como WEP, WPA-PSK, WPA2-PSK e WPS.

4.2 Validação da arquitetura

Inicialmente o grupo de pesquisa do LII (Laboratório de Informática Industrial) da UFRN desenvolveu vários dispositivos inteligentes (TAVARES et al., 2018; NETO et al., 2018) para serem utilizados em diversos cenários definidos previamente que simulam o cotidiano de um usuário, e assim possibilitar a validação da arquitetura proposta.

Os dispositivos inteligentes desenvolvidos permitem checar a comunicação com o *middleware* e a interface de usuário, para assim, possibilitar a avaliação dos tempos de resposta e análise da usabilidade e performance dos componentes da arquitetura.

Para ser possível a comunicação entre os dispositivos distintos, o *middleware* e a interface, foi criado um protocolo de padronização de comunicação para dispositivos IoT que permite o envio e recebimento de dados dos dispositivos pelo *middleware* (SILVEIRA et al., 2019).

No primeiro cenário foi desenvolvido um hidrômetro inteligente que realiza a medição do consumo de água e envia para o *middleware*. Com isso é possível, além da medição, detectar vazamentos e comparar, dia a dia, a quantidade de água que abastece a caixa d' água em uma escala de tempo. A Figura 10 mostra a UI com o gráfico do monitoramento do consumo de água. Ao analisar os dados recebidos, é possível comparar com os valores medidos analogicamente pelo próprio hidrômetro, e assim validar se a aplicação teve o comportamento esperado.

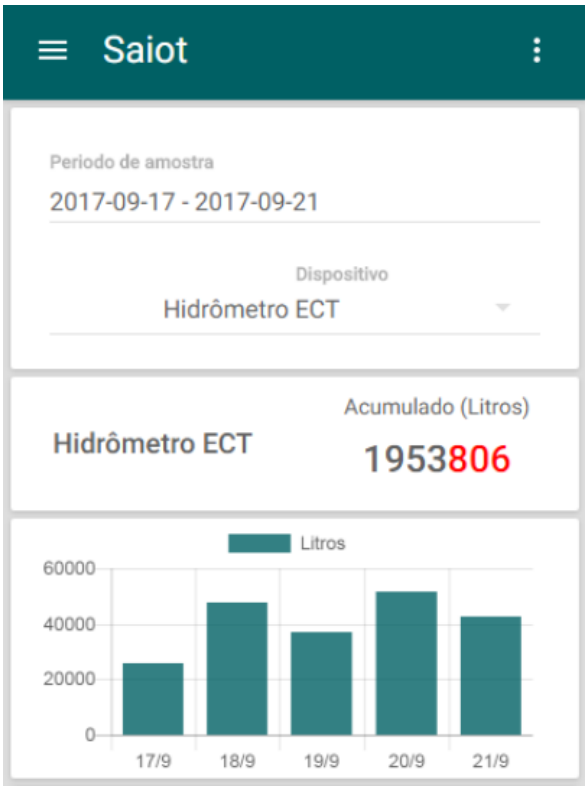

Figura 10 – Monitoramento do consumo de água

No segundo cenário foi implementado um medidor inteligente que monitora o consumo de energia elétrica. Como pode ser visto na Figura 11, através do dispositivo é possível saber o consumo e variáveis como tensão, corrente e potência da energia do local.

Figura 11 – Monitoramento do medidor de energia

O terceiro cenário é composto de um conjunto de lâmpadas para validar a fluidez da comunicação entre a interface e os dispositivos através da internet e também na rede local da residência. Na Figura 12 pode-se visualizar uma lista de dispositivos, mas neste cenário em específico foram desenvolvidos três tipos de lâmpadas, são elas: uma lâmpada comum de liga/desliga; uma lâmpada com variação de intensidade; e uma lâmpada com coloração RGB. Ao analisar a fluidez no comportamento das lâmpadas e da interface, pode-se concluir que o tempo de resposta é aceitável por ficar em torno de 1 milissegundo (OLIVEIRA et al., 2018) e permite a usabilidade da interface pelo usuário e interação com as lâmpadas.

Figura 12 – Controle de lâmpadas

Ao fim da análise, verificou-se que o *middleware* tem um comportamento estável, não perde nenhum dado recebido dos dispositivos, permite fluidez na interação da interface com os dispositivos e disponibiliza os dados de forma consistente para a interface do usuário.

5 CONCLUSÃO

Este trabalho foi realizado com o objetivo de apresentar uma proposta de arquitetura para automação residencial utilizando tecnologias abertas que possibilite a comunicação entre dispositivos de tal forma que possa ser utilizada em ambientes residenciais ou em contextos maiores, como universidades ou cidades.

Para validar a arquitetura proposta foi realizada a implementação de um *web service* utilizando tecnologias web e uso de rede *Wi-Fi*. Após isso, foi possível realizar testes em vários cenários, condições de infraestrutura de rede e dispositivos diferentes com o objetivo de analisar os seguintes pontos: confiabilidade do *middleware*; compatibilidade com protocolos e tecnologias existentes; e uso sem a necessidade de conexão com internet.

Com os resultados obtidos na realização dos testes, é possível concluir que o *middleware* opera de forma confiável por não apresentar perda de dados no processamento das requisições. A arquitetura é compatível com os roteadores residenciais por usar comunicação por rede *Wi-Fi* e torna-se universalmente acessível por utilizar protocolos como HTTP, *WebSocket* e MQTT. E por último, a aplicação pode ser utilizada com uma rede local ou com conexões instáveis, pois oferece a possibilidade de ter um *middleware* local que se comunica com os dispositivos e permite que o usuário aproveite os benefícios da automação.

REFERÊNCIAS

AMARANTE, A. P. M.; VASCONCELOS, C. D. F. L. D.; NETO, V. P. D. S. Simulação de cenários de iot focada na segurança residencial e da informação: Revisão da literatura. 2021.

BELSHE, M.; THOMSON, M.; PEON, R. Hypertext transfer protocol version 2 (http/2). 2015.

CROCKFORD, D. *The application/json media type for javascript object notation (json)*. [S.l.], 2006.

FETTE, I.; MELNIKOV, A. *The websocket protocol*. [S.l.], 2011.

FIELDING, R. et al. *Hypertext transfer protocol–HTTP/1.1*. [S.l.], 1999.

GIRAU, R.; NITTI, M.; ATZORI, L. Implementation of an experimental platform for the social internet of things. In: *2013 Seventh International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*. [S.l.: s.n.], 2013. p. 500–505.

HEJAZI, H. et al. Survey of platforms for massive iot. In: IEEE. *2018 IEEE international conference on future IoT technologies (future IoT)*. [S.l.], 2018. p. 1–8.

HERMANN, M.; PENTEK, T.; OTTO, B. Design principles for industrie 4.0 scenarios. In: IEEE. *2016 49th Hawaii international conference on system sciences (HICSS)*. [S.l.], 2016. p. 3928–3937.

LEU, J.-S.; CHEN, C.-F.; HSU, K.-C. Improving heterogeneous soa-based iot message stability by shortest processing time scheduling. *IEEE Transactions on Services Computing*, IEEE Computer Society, Los Alamitos, CA, USA, v. 7, n. 4, p. 575–585, 2014. ISSN 1939-1374.

NETO, I. M. B. et al. Thdi measurement system of home energy signal based on iot. In: IEEE. *2018 Workshop on Metrology for Industry 4.0 and IoT*. [S.l.], 2018. p. 180–185.

OLIVEIRA, G. M. et al. Comparison between mqtt and websocket protocols for iot applications using esp8266. In: IEEE. *2018 Workshop on Metrology for Industry 4.0 and IoT*. [S.l.], 2018. p. 236–241.

RESCORLA, E. Http over tls. 2000.

SILVEIRA, D. V. et al. Proposta de padronização de comunicação para dispositivos iot. In: *Congresso Brasileiro de Automática-CBA*. [S.l.: s.n.], 2019. v. 1, n. 1.

SOUZA, V. C. O. de; SANTOS, M. V. C. dos. Maturing, consolidation and performance of nosql databases: Comparative study. In: *Proceedings of the Annual Conference on Brazilian Symposium on Information Systems: Information Systems: A Computer Socio-Technical Perspective*. [S.l.: s.n.], 2015. v. 1, p. 32.

TAVARES, S. A. et al. Telemetry for domestic water consumption based on iot and open standards. In: IEEE. *2018 Workshop on Metrology for Industry 4.0 and IoT*. [S.l.], 2018. p. 1–6.

TILKOV, S.; VINOSKI, S. Node. js: Using javascript to build high-performance network programs. *IEEE Internet Computing*, IEEE, v. 14, n. 6, p. 80–83, 2010.

YOKOTANI, T.; SASAKI, Y. Comparison with http and mqtt on required network resources for iot. In: IEEE. *2016 international conference on control, electronics, renewable energy and communications (ICCEREC)*. [S.l.], 2016. p. 1–6.